Abordagens para Problemas de Roteamento

Marco Alves Ganhoto

Trabalho Final de Mestrado Profissional

Instituto de Computação
Universidade Estadual de Campinas

ABORDAGENS PARA PROBLEMAS DE ROTEAMENTO

Marco Alves Ganhoto

dezembro de 2004

Banca Examinadora:

- Prof. Dr. Flávio Keidi Miyazawa
  Instituto de Computação, UNICAMP (Orientador)
- Prof. Dr. Carlos Eduardo Ferreira
  Instituto de Matemática e Estatística, USP
- Prof. Dr. Orlando Lee
  Instituto de Computação, UNICAMP
- Prof. Dr. Ricardo Dahab
  Instituto de Computação, UNICAMP (Suplente)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Ganhoto, Marco Alves

G155a Abordagens para problemas de roteamento / Marco Alves Ganhoto -- Campinas, [S.P. :s.n.], 2004.

Orientador : Flávio Keidi Miyazawa

Trabalho final (mestrado profissional) - Universidade Estadual de Campinas, Instituto de Computação.

1. Otimização combinatória. 2. Heurística. 3. Algoritmos. 4. Pesquisa operacional. I. Miyazawa, Flávio Keidi. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Abordagens para Problemas de Roteamento

Este exemplar corresponde à redação final do Trabalho Final devidamente corrigida e defendida por Marco Alves Ganhoto e aprovada pela Banca Examinadora.

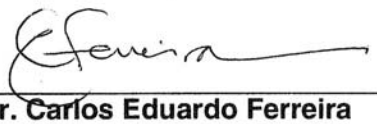
Campinas, 15 de dezembro de 2004.

Prof. Dr. Flávio Keidi Miyazawa
Instituto de Computação, UNICAMP (Orientador)

Trabalho Final apresentado ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Computação na Área de Engenharia de Software.

TERMO DE APROVAÇÃO

Trabalho Final Escrito defendido e aprovado em 15 de dezembro de 2004, pela Banca Examinadora composta pelos Professores Doutores:



Prof. Dr. Carlos Eduardo Ferreira
IME – Universidade de São Paulo

Prof. Dr. Orlando Lee
IC - UNICAMP

Prof. Dr. Flavio Keidi Miyazawa
IC - UNICAMP

Resumo

Neste trabalho, investigamos abordagens para problemas de roteamento, que têm como finalidade encontrar um melhor conjunto de rotas para que veículos possam transportar mercadorias a clientes geograficamente dispersos, respeitando certas restrições, como por exemplo, a de capacidade de carga dos veículos. Para isto, além de pesquisas em diversas fontes de informações, desenvolvemos um aplicativo para auxiliar no entendimento dos algoritmos, na ilustração do texto e na realização de experimentos. A partir de observações feitas durante as execuções do aplicativo, experimentamos combinações de critérios de seleção de localidades, utilizando tais combinações durante a realização dos movimentos de intercâmbio de vértices entre rotas de uma conhecida estratégia, a Metaheurística Busca Tabu. Foram combinados critérios baseados em distâncias com critérios baseados em ângulos, para compor algoritmos que foram testados com instâncias clássicas utilizadas por diversos pesquisadores. Os resultados obtidos foram apresentados juntamente com os de outras estratégias, fornecendo valores iguais ao melhor valor conhecido para duas instâncias, e valores intermediários para as outras cinco instâncias utilizadas nos testes.

Abstract

In this work, we examine some approaches for vehicle routing problems, to find a best set of routes to enable companies for delivery goods or commodities to customers, respecting some constraints, such as vehicles loading capacity. To this purpose, besides researching available information sources, we have developed a software to help us to understand the algorithms issues, for enriching the text with illustrations, and for effectiving some experiments concerning to previously selected approaches. From the analysis made during running software process, we decided to arrange chosen vertices criterias, using these arrangements in the vertices interchanging movements between routes of an already know method, the Tabu Search Metaheuristic. More precisely, we have combined distances and angles criteria, to implement algorithms on which it were tested using some classical instances considered by several researchers. The obtained results are presented with those selected approaches, and it provided us two equals values to the best known solution, and five intermediate values amoung to the others used on these experiments.

Agradecimentos

Dentre os materiais consultados para a elaboração deste trabalho, tive a felicidade de ter em mãos a dissertação de um dos alunos do Instituto de Computação da Unicamp. Nela, o autor encara a seção de agradecimentos como a vitrine das dissertações e teses, mencionando que essa é a primeira parte que ele procura ler quando manuseia um trabalho. Achei muito interessante a colocação dele e de certa forma vou tentar imitá-lo, procurando deixar essa parte agradável para que o leitor possa se interessar pela leitura do restante do trabalho.

Bem, inicialmente gostaria de agradecer ao amigo Taylor, por ter me falado sobre o Mestrado Profissional na Unicamp. Agradeço também aos senhores Adilson e Allan por colaborarem no meu processo de inscrição, com suas cartas de recomendação; na ocasião eles eram meus gerentes na AGF Brasil Seguros.

Agradeço ao Instituto de Computação pelo oferecimento do Mestrado Profissional, e aos professores do curso pelo conteúdo ministrado. Em especial, agradeço ao meu orientador, Prof. Flávio, pela sua paciência durante a orientação, pelas "aulas extras" de Otimização, pelas "sabatinas" sobre os tópicos do trabalho, pelas correções e idéias para melhorar esse texto, enfim, pela excelente orientação.

Agradeço também aos colegas do curso, em especial ao Adão, Josko, Wennder, Dinailton, Marcus Valerio e Luis Gustavo; "êta caboclada valente sô !!". Em alguns momentos, eu me pegava resmungando por ter que viajar de Jundiaí até Campinas para assistir as aulas. Aí eu me lembrava desses caras... Dois deles vinham de Brasília, outros dois de Uberlândia, um de Goiânia e outro de São Paulo, (semanalmente)... Isso é que é determinação... Agradeço a eles pelo companheirismo, pelas caminhadas para o almoço, pelas discussões e estudo em grupo.

Agradeço ao pessoal da Biblioteca Central da Unicamp pelos valiosos cursos do Programa de Capacitação de Usuários em Informação Científica. Agradeço também aos profissionais das outras bibliotecas da Unicamp, e das bibliotecas do IME e da POLI (USP), pela atenção e orientação durante a etapa de pesquisa. Confesso que passei a admirar ainda mais o trabalho dessas pessoas.

Agradeço aos amigos da SISCORP - Sistemas Corporativos, por cederem recursos e tempo para a realização dos testes. Agradeço ao meu coordenador Odair, por me apoiar e permitir que eu me ausentasse em alguns dias para me dedicar a este trabalho. Agradeço à turma do almoço, em especial aos professores Taveira e Paulo Marcio, pelos conselhos e pela camaradagem durante esses anos.

Agradeço também às meninas Hideko e Milinha, por tolerarem minha chatice durante o período de escrita do trabalho, e à tininha, uma chinchila da dona Milinha que insiste em conversar em assembler, e que é dona de movimentos acrobáticos que podem até servir de inspiração para uma heurística; se as formigas têm uma, por que a chinchila não pode ter?

Agradeço, de maneira muito especial, ao meu pai Salvador e à minha mãe Marilene, por me criarem num lar Cristão e por me apoiarem em todos os momentos da minha vida. Agradeço também à minha irmã Martha pela paciência e incentivo. Finalmente, agradeço acima de tudo a Deus, por tudo que Ele tem realizado na minha vida.

Sumário

1 Introdução

Quando viajamos por nossas estradas ou passamos por nossas ruas, é comum encontrarmos alguns veículos com logotipos de empresas de logística, distribuição e transporte. No dia-a-dia, tais veículos, enfrentam diversos tipos de problemas para levarem os pedidos aos seus respectivos clientes.

De imediato, podemos imaginar que para a empresa, bastaria escolher o conjunto de rotas de menor distância, ou o caminho mais curto, para realizar as suas entregas, pois fazendo assim, aparentemente economizaria variáveis como combustível e tempo, reduzindo com isso os gastos com o processo.

Contudo, a estratégia de tentar encontrar o conjunto de rotas de menor distância nem sempre é suficiente para a resolução de problemas reais. É comum encontrarmos estradas em péssimo estado de conservação, fazendo com que trechos mais curtos sejam percorridos em tempo maior e às vezes com maior consumo de combustível, desgaste do condutor e do veículo.

Outras variáveis como custos com pedágios também participam de problemas do cotidiano. Em alguns casos os condutores preferem passar por trechos um pouco mais longos, ou menos conservados, para poderem driblar os pedágios. Em outros casos, a rota mais curta acaba passando por ruas de uma grande cidade, fazendo com o que o condutor corra o risco de cair em um engarrafamento, atrasando com isso a entrega.

Enfim, existem variados tipos de problemas observados no dia-a-dia, que contribuem para deixar ainda mais complexa a tarefa da empresa de transportar e distribuir os seus produtos. Entretanto, antes de tentarmos propor melhorias para esses cenários, é necessário estudar estratégias e modelos apresentados pelos pesquisadores, para posteriormente tentar adequá-los às nossas necessidades.

A motivação para este trabalho consiste então em adquirir conhecimento e habilidades para tentar, em trabalhos futuros, apresentar soluções ou melhorias para cenários de transporte e distribuição vividos por nossas empresas.

1.1 Objetivos do Trabalho

O objetivo principal deste trabalho é estudar e apresentar alguns algoritmos utilizados na busca de soluções para os problemas de roteamento. Além disso, efetuamos alguns ensaios a partir de combinações de elementos observados nesses algoritmos. Para tanto, realizamos pesquisas em diversas fontes de informação, e introduzimos algumas das estratégias conhecidas em um aplicativo que apresenta o resultado da execução dos algoritmos em uma interface amigável. Com o auxílio deste aplicativo, observamos comportamentos e elaboramos experimentos que podem ser traduzidos como variações das estratégias estudadas.

1.2 Organização do Texto

Os próximos capítulos estão organizados da seguinte forma:

No capítulo 2, apresentamos uma breve introdução sobre problemas de roteamento, abordando suas aplicações e complexidade.

No capítulo 3, apresentamos alguns dos métodos publicados para o Problema do Caixeiro Viajante. Dedicamos uma parte do trabalho a este problema devido a sua importância histórica, e porque algumas estratégias para resolvê-lo podem também ser utilizadas no processo de resolução do Problema de Roteamento de Veículos.

No capítulo 4, apresentamos alguns dos métodos publicados para o Problema de Roteamento de Veículos. Tanto neste capítulo quanto no capítulo 3, procuramos exemplificar os passos dos algoritmos e ilustrar o resultado da aplicação dos mesmos.

No capítulo 5, apresentamos alguns ensaios que realizamos a partir de observações feitas durante o estudo e elaboração do trabalho. Os resultados dos experimentos foram colocados em tabelas juntamente com os resultados de outras estratégias, permitindo com isso, eventuais comparações.

No capítulo 6, apresentamos as conclusões deste trabalho.

O trabalho conta ainda com quatro apêndices e um anexo que fornecem informações diversas que auxiliam no entendimento dos recursos utilizados para ilustrar o texto.

2 Problemas de Roteamento

Um vendedor possui uma lista de clientes e periodicamente costuma visitá-los para oferecer suas mercadorias, ou coletar pedidos deles para entregas posteriores. Muitas vezes, tais clientes estão estabelecidos em cidades diferentes, fazendo com que o vendedor se desloque até elas, retornando depois à cidade origem onde reside. A ordem das cidades a serem visitadas pode variar, fazendo com que ele percorra distâncias maiores ou menores. Dentre as diversas possibilidades, o vendedor deve escolher a seqüência de visitas em que a distância a ser percorrida seja a menor possível.

Esse é o Problema do Caixeiro Viajante, um dos problemas mais estudados da área conhecida como Otimização Combinatória, onde os problemas apresentam a característica comum de que a solução ótima é procurada em um conjunto finito de possibilidades. Esse problema, assim como outros dessa área, são considerados problemas de difícil solução, ou seja, problemas que ainda não possuem algoritmos capazes de resolvê-los para qualquer quantidade de cidades. Apesar disso, existem algoritmos capazes de resolver instâncias de tamanho médio e até instâncias consideradas grandes de maneira exata. Algumas dessas estratégias, são baseadas na utilização do Método Simplex com estratégias de enumeração e planos de corte (Loparic [1]).

O Método Simplex possui uma história interessante. Caixeta-Filho [2] relata que no início da década de 40 uma equipe de cientistas, liderada pelo americano George Dantzig, foi convocada pelos aliados da segunda guerra para oferecer subsídios técnicos para as tomadas de decisão que envolvessem a distribuição ótima de tropas entre as diferentes frentes de batalha. O resultado desse esforço de pesquisa, finalizado em 1947 e publicado posteriormente, recebeu o nome de Método Simplex, que é um algoritmo incorporado na maioria dos *softwares* para resolução de problemas de programação linear.

Num cenário parecido com o anterior, uma empresa possui uma lista de clientes que periodicamente efetuam pedidos de compra de mercadorias. Tais pedidos são encaminhados ao depósito da empresa, e de lá, veículos são carregados com os produtos a serem entregues aos clientes. Devido a limitações, como a capacidade dos veículos por exemplo, os clientes serão atendidos em viagens diferentes, sendo que existem diversas possibilidades de organizar tais

viagens. Neste caso, temos que tentar identificar o conjunto de rotas em que o custo com as viagens seja o menor possível.

Esse é o Problema de Roteamento de Veículos, um problema de grande aplicação prática, que apesar da estreita semelhança com o anterior, é alvo de estudos mais recentes. Devido a algumas características como demanda dos clientes, capacidade dos veículos, possíveis restrições de tempo, o Problema de Roteamento de Veículos é também considerado um problema muito difícil de ser resolvido, sendo que os algoritmos existentes para resolvê-lo de maneira ótima conseguem atacar apenas instâncias consideradas pequenas, quando comparadas ao tamanho das instâncias do Problema do Caixeiro Viajante resolvidas de maneira exata.

2.1 Aplicações Práticas

Goldbarg e Luna [3] relatam que podemos encontrar diversas versões do Problema do Caixeiro Viajante sendo aplicadas em inúmeros problemas práticos. Os autores destacam as seguintes aplicações:

- Programação de operações de máquinas em manufatura
- Programação de transporte entre células de manufatura
- Otimização do movimento de ferramentas de corte
- Otimização de perfurações e soldas de circuitos impressos
- Na maioria dos problemas de roteamento de veículos
- Na solução de problemas de seqüenciamento de DNA (Genoma)
- Na solução de problemas de programação e distribuição de tarefas em plantas
- Trabalhos administrativos

Para o Problema de Roteamento de Veículos, os referidos autores mencionam que existe um grande número de aplicações práticas que afetam diversos setores como a indústria, o comércio, o setor de serviços, a segurança, a saúde pública e o lazer. Tais aplicações foram estudadas por diversos pesquisadores, e podem ser condensadas da seguinte forma:

- Distribuição de jornais, manufaturados, produtos diversos, bebidas, valores, produtos químicos, pão, gás, derivados de petróleo, alimentos, material fotográfico, vagões ferroviários, leite, concreto, etc.
- Transporte de pedras, escolar, coletivo (táxi, ônibus, trens de metrô), bens pessoais (mudanças), etc.

- Recolhimento de lixo, borracha, garrafas de leite, cana de açúcar , etc.
- Entrega de correspondências, pizzas, congelados, correspondências bancárias, etc.
- Serviços de emergência, patrulhamento policial, proteção contra incêndio, medidores elétricos, limpeza de rua com veículos vassoura, limpeza de gelo em ruas e estradas, etc.
- Roteamento de helicópteros, linhas aéreas, navios de curso longo (petroleiros) e cabotagem, auditores bancários, entre células de manufatura flexível, de robôs em manufatura, de pacotes em redes de computadores, de fluxo de comunicações em redes de telecomunicações, etc.
- Projeto de redes de telecomunicações em anéis.

2.2 Complexidade Computacional

O termo Complexidade Computacional está relacionado com o estudo dos problemas e dos algoritmos capazes de resolvê-los. Segundo Ziviani [4], um algoritmo pode ser definido como uma seqüência de ações executáveis para a obtenção de uma solução para um determinado tipo de problema. Podem existir vários algoritmos capazes de resolver um mesmo problema, sendo necessário portanto, realizar a escolha do mais apropriado.

Para identificar o algoritmo mais apropriado, é necessário medir o custo de execução de cada candidato. Para isso, é comum definir uma função de custo, ou função de complexidade *f*, onde *f(n)* pode ser a medida de espaço de memória ou a medida de tempo, necessário para executar um algoritmo para uma instância de tamanho *n*. No primeiro caso, *f* é chamada de função de complexidade de espaço do algoritmo, enquanto que no segundo, *f* é chamada de função de complexidade de tempo do algoritmo. É importante destacar que a complexidade de tempo nem sempre representa tempo diretamente. Ela pode representar o número de execuções de determinadas operações consideradas relevantes, como por exemplo, comparações, operações aritméticas, etc. (Ziviani [4]).

O custo para obter uma solução para um dado problema aumenta de acordo com o tamanho *n* do problema. Em outras palavras, o tempo necessário para resolver o problema cresce quando *n* cresce. Para visualizar este fato, consideremos o Problema do Caixeiro Viajante com *n* municípios. Podemos tentar resolvê-lo com o método de enumeração completa, que simplesmente lista todas as possíveis soluções e escolhe a melhor delas. Como podemos partir de qualquer município, existem $(n - 1)! / 2$ possíveis soluções para o problema (simétrico – versão

onde as distâncias entre os pares de municípios são as mesmas, independente do sentido viajado). Supondo que um computador possa listar todas as possíveis soluções de um problema de 10 municípios em aproximadamente 1 segundo, observe na tabela seguinte o que ocorreria para resolvermos instâncias maiores com o mesmo computador.

(n) municípios	**$f(n) = (n - 1)! / 2$**	**Tempo estimado para solução**
10	181440	~ 1 segundo
11	1814400	~ 10 segundos
12	19958400	~ 2 minutos
13	239500800	~ 22 minutos
14	3113510400	~ 5 horas
15	43589145600	~ 3 dias
16	6,53837E+11	~ 42 dias
17	1,04614E+13	~ 2 anos
18	1,77844E+14	~ 31 anos
19	3,20119E+15	~ 5 séculos
20	6,08226E+16	~ 10 milênios

Observando os dados da tabela, podemos notar que para uma pequena variação do número de municípios ocorre uma variação gritante no tempo de resolução do problema. Existem algoritmos melhores que o método de enumeração completa para a obtenção da solução ótima do Problema do Caixeiro Viajante e de outros problemas.

Ziviani [4] relata que para valores pequenos de n, qualquer algoritmo custa pouco para ser executado, em outras palavras, a escolha do algoritmo não é tarefa crítica para problemas de tamanho pequeno. Por isso, a análise de algoritmos é realizada para valores grandes de n, estudando o comportamento de suas funções de custo para tais valores. Em outras palavras, estuda-se o comportamento assintótico das funções de custo, que representa o limite do comportamento do custo quando n cresce.

Para o estudo do comportamento assintótico utiliza-se a notação O que pode ser formalizada da seguinte maneira: Dizemos que $g(n) = O(f(n))$ se existirem duas constantes positivas c e uma inteira m tais que $g(n) \leq c \bullet f(n)$ para $n > m$. Isto é, para valores de n suficientemente grandes, $g(n)$ não cresce mais rapidamente que $f(n)$. Podemos ler também, $g(n)$ é da ordem no máximo $f(n)$.

Os algoritmos podem então, ser avaliados a partir da comparação dos seus comportamentos assintóticos. Um algoritmo com tempo de execução $O(n)$ é melhor que um

algoritmo com tempo de execução $O(n^2)$. Contudo, constantes de proporcionalidade podem gerar algumas exceções. Por exemplo, um algoritmo com tempo de execução $2n^2$ pode ser mais eficiente do que um algoritmo com tempo de execução $100n$; isso ocorre para instâncias de problemas com tamanho $n < 50$. Entretanto, para valores maiores de n o algoritmo de complexidade $O(n^2)$ é muito mais lento que o algoritmo de complexidade $O(n)$.

As principais funções de complexidade que definem o comportamento assintótico dos algoritmos são apresentadas por Ziviani [4] da seguinte maneira:

- $f(n) = O(1)$ **[complexidade constante]**. O uso do algoritmo não depende do tamanho de n; suas instruções são executadas um número fixo de vezes.

- $f(n) = O(\log n)$ **[complexidade logarítmica]**. Este tipo de função ocorre em algoritmos que resolvem um problema dividindo-o em problemas menores.

- $f(n) = O(n)$ **[complexidade linear]**. Esse tipo de comportamento aparece em algoritmos que realizam um pequeno trabalho em cada elemento de entrada.

- $f(n) = O(n \log n)$ **[complexidade $n \log n$]**. Esse tipo de comportamento ocorre em algoritmos que atacam um problema dividindo-o em problemas menores, resolvendo cada um deles independentemente, ajuntando em seguida as soluções.

- $f(n) = O(n^2)$ **[complexidade quadrática]**. Esse tipo de função ocorre em algoritmos em que os itens de dados são processados aos pares, muitas vezes em um laço dentro do outro.

- $f(n) = O(n^3)$ **[complexidade cúbica]**. Algoritmos desta ordem de complexidade são utilizados apenas para resolver pequenos problemas.

- $f(n) = O(2^n)$ **[complexidade exponencial]**. Esse tipo de função ocorre em algoritmos que utilizam força bruta para resolver o problema.

- $f(n) = O(n!)$ **[complexidade fatorial]**. Esse tipo de comportamento aparece em algoritmos que listam todas as possíveis combinações para escolher a melhor, como o método de enumeração completa utilizado no exemplo anterior do Problema do Caixeiro Viajante.

Observando a tabela seguinte, podemos visualizar a diferença de tempo entre algumas funções de complexidade típicas para diferentes tamanhos de *n*, onde cada função expressa o tempo de execução em microsegundos.

***f*(*n*)**	***n* = 10**	***n* = 20**	***n* = 30**	***n* = 40**	***n* = 50**	***n* = 60**
n	0,00001 segundos	0,00002 segundos	0,00003 segundos	0,00004 segundos	0,00005 segundos	0,00006 segundos
n^2	0,0001 segundos	0,0004 segundos	0,0009 segundos	0,0016 segundos	0,0025 segundos	0,0036 segundos
n^3	0,001 segundos	0,008 segundos	0,027 segundos	0,064 segundos	0,125 segundos	0,216 segundos
n^5	0,1 segundos	3,2 segundos	24,3 segundos	1,7 minutos	5,2 minutos	13,0 minutos
2^n	0,001 segundos	1,0 segundos	17,9 minutos	12,7 dias	35,7 anos	366 séculos
3^n	0,059 segundos	58 minutos	6,5 anos	3855 séculos	2 x 10^8 séculos	1,3 x 10^13 séculos

Fonte: Pelizaro [5] p. 30 APUB Garey e Johnson [6]

Um algoritmo polinomial é aquele que possui função de complexidade $O(p(n))$, onde $p(n)$ é um polinômio. Já um algoritmo exponencial, é aquele que possui função de complexidade $O(c^n)$, com $c > 1$. Pelizaro [5] menciona que a definição de algoritmo exponencial inclui também outras funções de complexidade de tempo não polinomiais, como por exemplo n^log n, que normalmente não é considerada uma função exponencial.

A partir da tabela anterior, podemos perceber um dos motivos que torna um algoritmo polinomial mais desejável que um algoritmo exponencial. Os algoritmos polinomiais são tidos como bons algoritmos, ou algoritmos eficientes, enquanto que, os algoritmos exponenciais são classificados como ineficientes.

Entretanto, existem algumas exceções para tal classificação, que ocorrem geralmente para instâncias de tamanho limitado. Por exemplo, o algoritmo de complexidade 2^n é mais rápido que o algoritmo n^5 para $n \leq 20$ (ver tabela anterior). De qualquer forma, a distinção entre algoritmos eficientes e ineficientes torna-se mais significativa com o aumento do tamanho de *n*.

Os algoritmos exponenciais são geralmente variações dos métodos exaustivos, enquanto que os algoritmos polinomiais são concebidos através de um entendimento maior das características do problema, fazendo com que eles sejam mais úteis na prática do que os exponenciais. Apesar disso, existem algoritmos exponenciais que são muito úteis na prática,

como por exemplo, o Simplex, que possui complexidade exponencial para o pior caso, mas que em média executa muito rápido (Ziviani [4]).

Da mesma forma que um algoritmo pode ser classificado como bom ou ruim, dependendo se ele possui ou não função de complexidade polinomial, os problemas podem ser classificados como fáceis ou difíceis, dependendo se eles possuem ou não um algoritmo polinomial capaz de resolvê-los.

Para o estudo da classificação dos problemas, a teoria da complexidade computacional se restringe à **problemas de decisão**, ou seja, problemas cujas soluções requerem apenas um SIM ou NÃO como resposta. Contudo, muitos **problemas de otimização** podem ser naturalmente expressos como problemas de decisão, tal que se existir um algoritmo de tempo polinomial para o problema de decisão, então existirá um também para a sua versão em problema de otimização (Pelizaro [5]).

Como exemplo, podemos gerar uma versão de problema de decisão para o Problema do Caixeiro Viajante acrescentando um limite numérico como parâmetro adicional. Assim fazendo, ao invés de procurarmos pela rota que passa por todas as cidades percorrendo a menor distância, temos que responder a seguinte questão: existe uma rota que passa por todas as cidades com distância total não maior que D? Note que a questão exige um SIM ou NÃO como resposta, onde D é o nosso parâmetro adicional.

Embora a teoria tenha sido desenvolvida para os problemas de decisão, suas implicações podem ser estendidas para os problemas de otimização. Em Lewis e Papadimitriou [7] podemos encontrar um estudo detalhado sobre a classificação dos problemas. Os referidos autores apresentam também o conceito de redução polinomial, necessário para o entendimento de tal classificação.

Segundo Lewis e Papadimitriou [7], o conceito de redução polinomial é importante por revelar interessantes afinidades entre problemas computacionais. Podemos dizer que T é uma redução polinomial do problema V para o problema X quando T transforma, em tempo polinomial, instâncias do problema V em instâncias do problema X, de maneira que Vi é uma instância do problema V com resposta SIM, se e somente se, $T(Vi)$ for uma instância do problema X que fornece a resposta SIM.

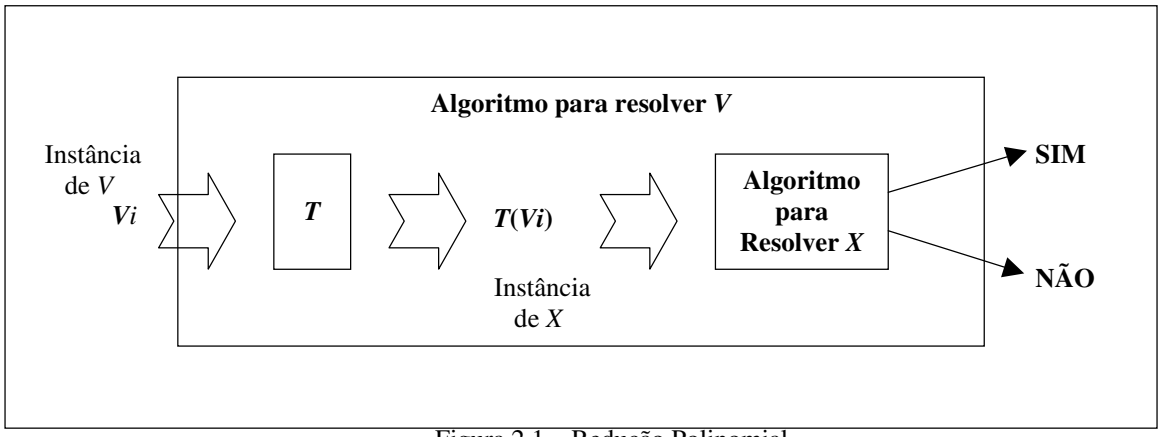


Figura 2.1 – Redução Polinomial
Fonte: Lewis e Papadimitriou [7] p. 292

Quando existe uma redução Polinomial de *V* para *X*, podemos dizer que *X* é no mínimo mais complexo que *V* . Se existir um algoritmo polinomial para *X*, esse método também resolverá *V* em tempo polinomial. Se *V* exigir um tempo exponencial, então *X* exigirá um tempo pelo menos exponencial.

A maioria dos problemas de decisão associados a problemas de interesse prático, derivados ou não de problemas de otimização, pertencem às seguintes classes:

- **Classe P (*Polynomial time*)**. Um problema *X* pertence à classe P se ele pode ser resolvido por um algoritmo de tempo polinomial.

- **Classe NP**. Um problema *X* pertence à classe NP quando ele admite, ou apresenta, um certificado. Um certificado deve ser polinomialmente sucinto, ou seja, seu comprimento deve ser dado no máximo, pelo valor de um polinômio sobre o comprimento da cadeia de entrada; ele também deve ser verificável em tempo polinomial.

- **Classe NP-Completo**. Um problema *X* pertence à classe NP-Completo se ele for pertencente à classe NP e se qualquer outro problema *V* também pertencente à classe NP puder ser reduzido polinomialmente a *X*. Em outras palavras, *X* é NP-Completo quando está em NP, e para qualquer *V* em NP, um algoritmo para resolver *X* pode ser adaptado em tempo polinomial para resolver *V* (ver ilustração anterior).

- **Classe NP-Difícil (NP-Árduo ou NP-*Hard*)**. Um problema *X* pertence à classe NP-Difícil se qualquer outro problema *V* pertencente à classe NP puder ser reduzido polinomialmente a *X*. Note que aqui, *X* não necessariamente pertence à classe NP. Um problema NP-Difícil é pelo menos tão difícil quanto qualquer problema em NP.

A classe P está contida em NP. No entanto, existem muitos problemas em NP para os quais não são conhecidos algoritmos que os resolvam em tempo polinomial. Por isso não se sabe se P = NP; existe uma forte conjectura de que P ≠ NP, embora isso não tenha ainda sido provado. Na figura seguinte, temos uma provável configuração das classes.

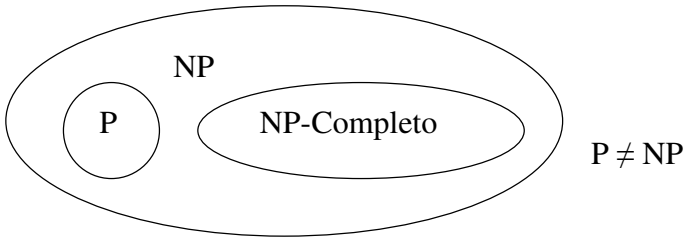


Figura 2.2 – Provável configuração das classes
Fonte: Pelizaro [5] p. 35 APUB Garey e Johnson [6]

Nos capítulos seguintes, serão apresentadas estratégias para o Problema do Caixeiro Viajante, pertencente à classe NP-Completo, e para o Problema de Roteamento de Veículos, pertencente à classe NP-Difícil. Quando encontramos classificações como essas em artigos e livros, devemos ter em mente que os autores estão se referindo à versão de decisão dos problemas que estão sendo classificados, já que a teoria da complexidade computacional trabalha com problemas de decisão.

3 O Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV) é um dos problemas mais estudados em Otimização Combinatória. O objetivo do PCV é encontrar em um grafo $G=(V,A)$, um circuito hamiltoniano de menor custo.

Um grafo pode ser definido como um conjunto de vértices e arestas. Os vértices, ou nós, são pontos que podem representar cidades, depósitos, postos de trabalho ou atendimento. Já as arestas, ou arcos, são linhas que conectam os vértices, podendo representar ruas ou estradas. O custo ou tamanho de um circuito hamiltoniano é obtido pela soma dos custos ou tamanhos das arestas a serem percorridas.

Um circuito hamiltoniano é um passeio que percorre todos os vértices de um grafo e retorna ao vértice origem (início do passeio), passando por cada vértice apenas uma vez. Tal passeio recebe esse nome devido a Willian Rowan Hamilton que, em 1857, elaborou um jogo sobre um dodecaedro em que cada vértice estava associado a uma importante cidade da época. O objetivo era encontrar uma rota através dos vértices do dodecaedro que iniciasse e terminasse numa mesma cidade sem nunca repetir uma visita (Goldbarg e Luna [3]).

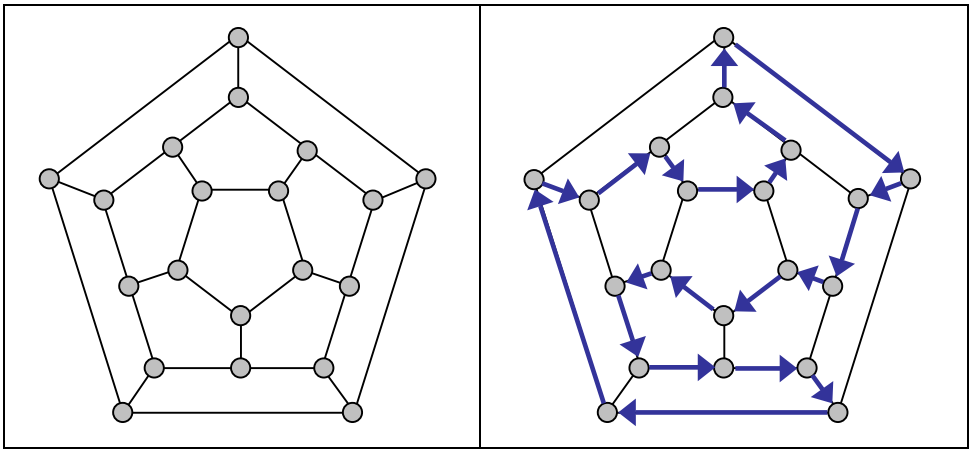


Figura 3.1 – Exemplo de solução do jogo de Hamilton

Goldbarg e Luna [3] relatam que Hamilton não foi o primeiro a apresentar o problema, mas o seu jogo ajudou a divulgá-lo. Os autores mencionam ainda que modernamente, a primeira

menção conhecida do PCV é devida a Hassler Whitney em 1934, em um trabalho na Princeton University. Com o passar do tempo, a quantidade de versões do problema aumentou, sendo que algumas delas apresentam particularidades que tornam mais fáceis as tentativas de solução.

Quando os custos de viagem de uma cidade para a outra não dependem do sentido percorrido ($Cij = Cji$, $\forall\ i, j \in V$), o problema é denominado **PCV Simétrico**. Quando todos os nós estão contidos num mesmo plano, e os custos entre eles corresponderem às distâncias Euclidianas, o problema é denominado **PCV Euclidiano** (Bellmore e Nemhauser [8]).

Neste capítulo, apresentamos algumas estratégias para o PCV, sendo que nosso interesse se restringe a métodos que fornecem resultados para o PCV Simétrico. Eventualmente, uma ou outra estratégia pode ser utilizada em outras versões do PCV, mas o nosso foco está em resolver problemas de roteamento, onde os custos entre os vértices estarão relacionados com distâncias geográficas.

Além das descrições dos passos dessas estratégias, desenvolvemos um programa para ilustrar a aplicação dos algoritmos num cenário próximo do leitor (mapa do estado de São Paulo). Como exemplo, a figura seguinte nos mostra o resultado da aplicação da heurística das economias seguida da heurística de melhoria 2-Opt (que serão abordadas neste trabalho), para um cenário onde o caixeiro viajante visita a capital e todos os municípios iniciados pela letra P.

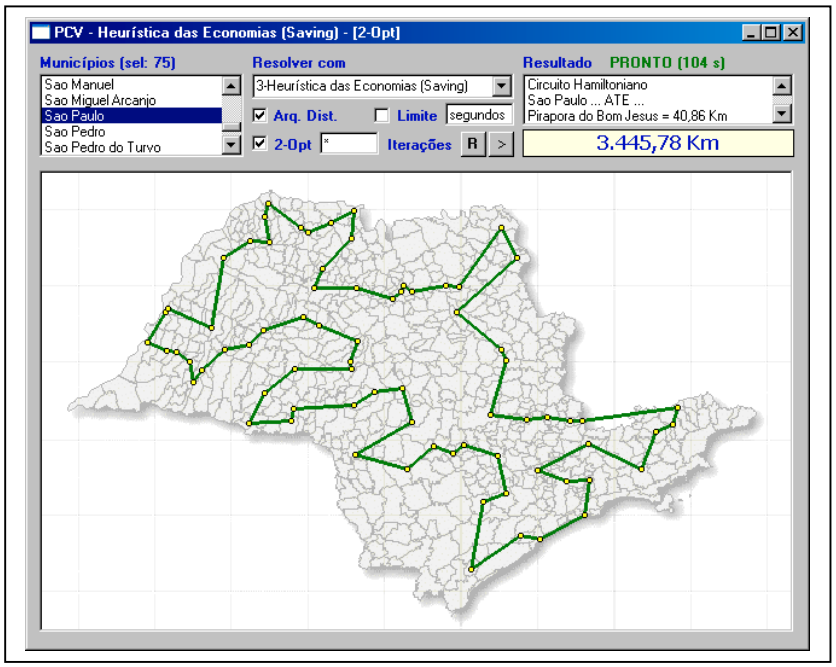


Figura 3.2 - Simulação da Heurística das Economias + 2-Opt
(capital e municípios do estado de SP iniciados pela letra P)

A figura 3.2 nos mostra a tela do aplicativo, onde a caixa de texto localizada no canto direito, logo acima do mapa, armazena o tamanho ou custo do circuito hamiltoniano gerado para a instância em questão. No aplicativo, as distâncias entre os municípios são calculadas a partir das coordenadas geográficas dos mesmos. O leitor pode estranhar tais distâncias se compará-las com as distâncias obtidas em guias rodoviários. No apêndice C apresentamos um exemplo desse cálculo e realizamos algumas comparações para ilustrar a diferença existente entre esses valores. Informalmente as distâncias percorridas pelas rodovias são aproximadamente 20% maiores que as calculadas através das coordenadas geográficas.

3.1 Algoritmos Heurísticos

O PCV é um problema de difícil tratamento onde soluções para um grande número de vértices consomem um longo tempo de processamento. Os algoritmos que fornecem uma solução exata são mais apropriados para a resolução de instâncias menores do problema. Diante disso algoritmos heurísticos ou aproximados, são desenvolvidos para encontrar soluções próximas da ótima com um tempo de processamento aceitável.

Chong [9] menciona que as heurísticas para o PCV podem ser divididas em três grupos:

- Heurísticas construtivas – buscam construir o circuito hamiltoniano de maneira gradativa, através de adições seqüenciais de vértices ou municípios.
- Heurísticas de melhoria – a partir de um circuito inicial, buscam melhorar o custo do mesmo através de trocas de posições dos municípios.
- Heurísticas compostas – combinam elementos dos dois procedimentos anteriores.

Existem vários exemplos de heurísticas para cada um desses grupos, sendo que no grupo de heurísticas compostas, estratégias bem elaboradas já foram propostas com o objetivo de encontrar soluções cada vez mais próximas da ótima. Para esta parte do trabalho, nos concentramos em apresentar algumas abordagens clássicas dos dois primeiros grupos que estão relacionadas a seguir.

3.1.1 Heurística de Bellmore e Nemhauser

No ano de 1968 foi publicado um estudo de Bellmore e Nemhauser sobre o Problema do Caixeiro Viajante. Dentre os teoremas, métodos e técnicas de solução, os autores apresentaram

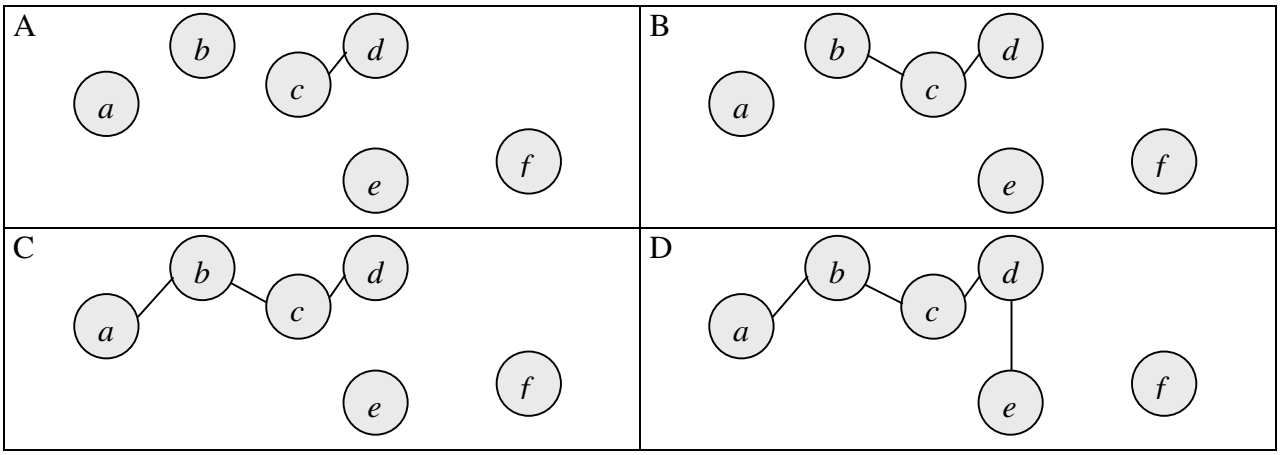
uma heurística construtiva que elege um vértice inicial de maneira arbitrária e na seqüência inclui outros vértices no circuito em formação, seguindo o critério de escolha do vértice ou vizinho mais próximo (Bellmore e Nemhauser [8]).

A complexidade dessa heurística é $O(n^2)$. Goldbarg e Luna [3] mencionam que existe uma variação elaborada para minimizar o efeito da influência da escolha do vértice inicial. Nela, o algoritmo é repetido para todos os possíveis vértices candidatos ao início do processo. Nesse caso, a heurística passa a ter complexidade $O(n^3)$. Apesar de existirem algumas variações da regra de escolha do vizinho mais próximo, podemos escrever o algoritmo da seguinte forma:

Heurística de Bellmore e Nemhauser

1. Escolha um vértice inicial, encontre o vértice mais próximo dele e inicie a construção do circuito através da ligação desses dois extremos.
2. Encontre o vértice mais próximo de um dos vértices extremos incluídos na solução; insira este vértice na rota em construção ligando-o com o seu vizinho mais próximo.
3. Se todos os vértices foram incluídos, complete com a aresta que liga os vértices extremos e pare.
4. Caso contrário retorne ao passo 2.

Podemos exemplificar a execução dos passos dessa heurística fazendo uso do cenário incluído na próxima figura, onde as distâncias entre os vértices são as seguintes: *ab*=15*, ac*=25, *ad*=36, *ae*=37, *af*=56, *bc*=13, *bd*=23, *be*=29, *bf*=45, *cd*=11, *ce*=17, *cf*=32, *de*=18, *df*=25, *ef*=20. Os valores apresentados correspondem à distância aproximada, em milímetros, entre os centros dos vértices que formam os pares relacionados.



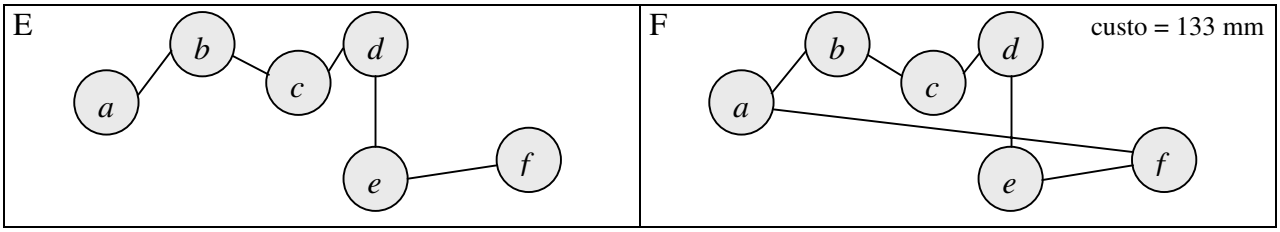


Figura 3.3 - Heurística de Bellmore e Nemhauser passo a passo.

Observando a figura anterior, notamos que o primeiro passo da heurística é executado com a escolha do vértice inicial *c*, seguido da ligação deste com o seu vizinho mais próximo *d*. A rota passa então a ser expandida a partir da execução do passo 2, onde o vizinho mais próximo de um dos vértices extremos da solução vai sendo localizado e incluído no circuito em formação. Os passos 3 e 4 da heurística estão relacionados com a verificação da formação de um circuito hamiltoniano para prosseguir ou parar com a execução do algoritmo. No final do processo, encontramos na figura 3.3 um circuito hamiltoniano de custo igual a 133 mm.

Podemos também exemplificar o uso dessa heurística num cenário prático, utilizando o módulo de simulação mencionado anteriormente. Neste exemplo, bem como nos outros apresentados neste capítulo, simulamos a geração do circuito hamiltoniano em que o caixeiro viajante visita a capital São Paulo e os municípios do estado iniciados pela letra J.

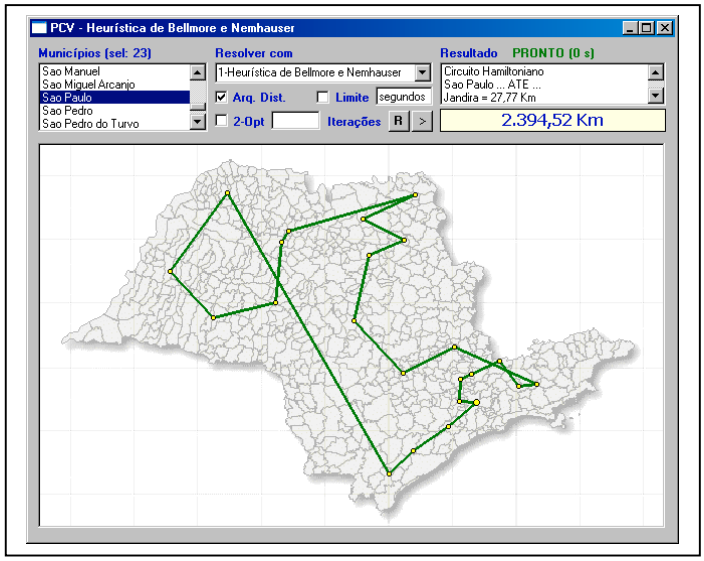


Figura 3.4 - Simulação da Heurística de Bellmore e Nemhauser
(capital e municípios do estado de SP iniciados pela letra J)

Tomando São Paulo como o município de partida, o percurso apresentado na figura 3.4 foi montado a partir dos seguintes trechos:

São Paulo → (27,77 Km) → Jandira → (38,61 Km) → Jundiaí → (18,99 Km) → Jarinu → (50,30 Km) → Joanópolis → (51,38 Km) → Jacareí → (28,91 Km) → Jambeiro → (146,38 Km) → Jaguariúna → (91,80 Km) → Jumirim → (118,78 Km) → Jaú → (118,21 Km) → Jaboticabal → (63,45 Km) → Jardinópolis → (76,78 Km) → Jaborandi → (95,61 Km) → Jeriquara → (216,01 Km) → Jaci → (22,45 Km) → José Bonifácio → (107,18 Km) → Júlio Mesquita → (103,86 Km) → João Ramalho → (106,86 Km) → Junqueirópolis → (165,98 Km) → Jales → (556,57 Km) → Jacupiranga → (55,88 Km) → Juquiá → (71,19 Km) → Juquitiba → (61,55 Km) → **São Paulo**

Lembrando que as distâncias apresentadas foram calculadas a partir do uso das coordenadas geográficas dos municípios. Observando a solução podemos notar que, devido a natureza seqüencial da heurística, a última inserção é forçada e inclui no circuito um grande trecho entre os municípios de Jales e Jacupiranga.

3.1.2 Heurísticas de Inserção

As heurísticas de inserção são procedimentos construtivos que ainda possuem um processo de decisão míope, porém, um pouco mais elaborado do que o de algumas heurísticas simplesmente gulosas. Goldbarg e Luna [3] relatam que tais heurísticas partem de um circuito inicial (normalmente utilizando três vértices) e vão selecionando e inserindo vértices, ainda não pertencentes a solução, até completar o circuito hamiltoniano. Na execução deste procedimento, são tomados três tipos de decisão:

- a decisão de um circuito inicial.
- a escolha do vértice a ser inserido na solução.
- a posição de inserção do novo vértice.

Quanto a decisão de escolha do vértice a ser incluído no circuito, existem alguns critérios para a seleção do candidato, dentre eles:

- inserção do vértice mais próximo de um dos vértices do circuito.
- inserção do vértice mais distante de um dos vértices do circuito.
- inserção do vértice que conduz ao circuito mais barato.
- inserção aleatória.

No caso das inserções de vértices mais próximos ou mais distantes de qualquer vértice do circuito em construção, escolhido o candidato resta ainda decidir como ele será adicionado ao circuito. Como o novo vértice ainda não pertence ao circuito serão necessárias duas arestas para incluí-lo. Se a inclusão do candidato for realizada entre os vértices Vi e $Vi+1$ também será necessário remover a aresta que já existia entre esses dois vértices.

O critério de decisão sobre que arestas deverão ser utilizadas é então expresso pelo efeito do balanço entrada x saída de arestas, ou seja, temos que minimizar a soma dos custos das arestas incluídas subtraindo o custo da aresta removida. Supondo que o candidato a inclusão é o vértice *Vj*, escolhido por estar mais próximo do vértice *Vi* já pertencente ao circuito, para encontrarmos a posição para adicioná-lo teríamos que satisfazer o critério: **Minimizar {$C_{i,j} + C_{j,i+1} - C_{i,i+1}$}**.

No caso da inserção que conduz ao circuito mais barato, o próprio candidato passa a ser escolhido pelo critério de mínimo balanço, considerando todas as possíveis inserções para todos os vértices ainda não pertencentes ao circuito em formação (Goldbarg e Luna [3]).

Para este trabalho, implementamos a variação da heurística que utiliza o critério de inserção do vértice mais próximo de um dos vértices pertencentes ao circuito em formação. O algoritmo pode ser descrito pelos seguintes passos:

Heurística de Inserção

1. Inicie por um circuito com três vértices escolhidos por algum critério.
2. Encontre o vértice *Vj*, não pertencente ao circuito, mais próximo de qualquer um dos vértices pertencentes ao circuito em formação.
3. Encontre a posição de inserção entre o vértice mais próximo *Vi* e um de seus vizinhos Vi_{+1} que minimize $\{Ci,j + Cj,i_{+1} - Ci,i_{+1}\}$.
4. Insira o candidato entre os vértices *Vi* e Vi_{+1}.
5. Se com a adição do vértice candidato ocorrer a formação de um circuito hamiltoniano então pare, senão retorne ao passo 2.

Para exemplificar os passos do algoritmo, usamos o cenário proposto para este capítulo.

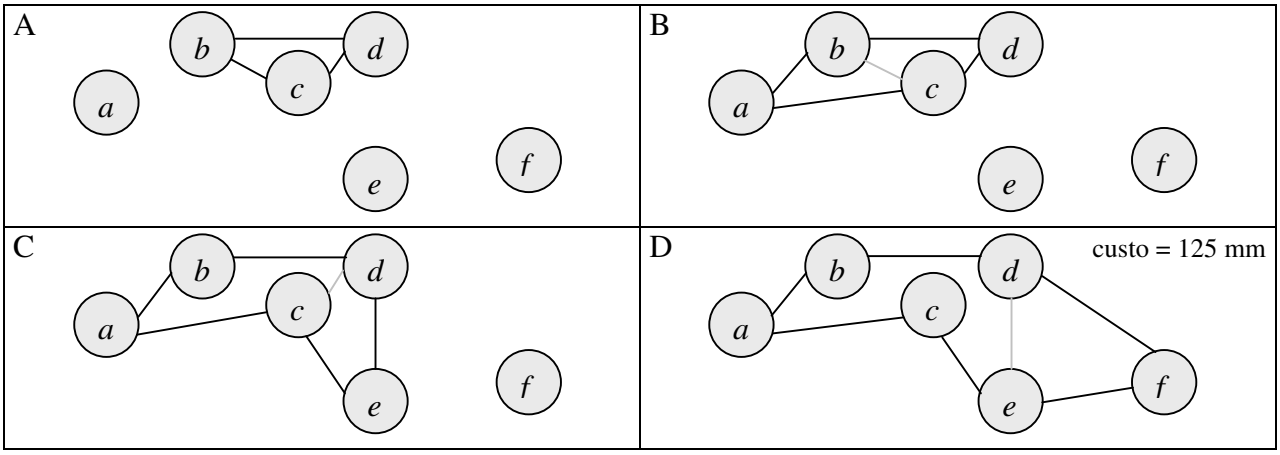


Figura 3.5 - Heurística de Inserção passo a passo.

Observando a figura 3.5, notamos que o algoritmo parte de um circuito inicial escolhido por algum critério, que na nossa implementação foi o de adotar os três vértices mais próximos. Em seguida o algoritmo procura um vértice ainda não incluído na solução que esteja mais próximo de um dos vértices do circuito. Encontrado este vértice, cabe ainda decidir em que posição ele será incluído. Como exemplo, vamos observar a transição entre as células B e C. Na célula B, a rota em formação contava com os vértices *a*, *b*, *c*, *d*. Executando o segundo passo da heurística, encontramos o vértice *e* como o próximo candidato a inserção, por estar mais próximo de um dos vértices do circuito, no caso *c*. Em seguida é necessário decidir se o candidato *e* será inserido entre os vértices *c* e *d* ou entre *c* e *a*. Para isso o terceiro passo calcula:

inserindo entre *c* e *d*: custo = $ce + de - cd = 24$

inserindo entre *c* e *a*: custo = $ce + ae - ac = 29$

Após o cálculo o algoritmo decide que a inclusão ocorrerá entre os vértices *c* e *d*. Na figura, as linhas claras indicam as arestas que foram sendo removidas. A cada inclusão realizada o algoritmo verifica se ocorreu a formação de um circuito hamiltoniano para parar ou continuar com novas inserções. No final do procedimento, encontramos um circuito hamiltoniano de custo igual a 125 mm, um valor melhor que o obtido através da estratégia anterior.

A partir do aplicativo desenvolvido, podemos observar a solução gerada com o uso dessa heurística para o mesmo cenário prático apresentado anteriormente.

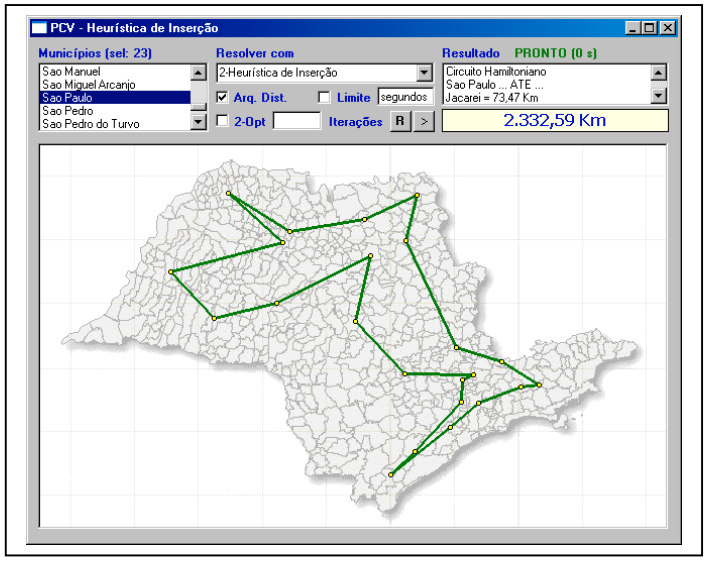


Figura 3.6 - Simulação da Heurística de Inserção
(capital e municípios do estado de SP iniciados pela letra J)

Comparando as figuras 3.6 e 3.4, notamos que o resultado apresentado pela heurística de inserção, para esse cenário, é melhor que o resultado obtido através da execução da heurística do vizinho mais próximo. Consultado o apêndice A, podemos observar os trechos do circuito hamiltoniano apresentado.

3.1.3 Heurística das Economias

A Heurística das Economias é um procedimento construtivo elaborado por Clarke e Wright [10] originalmente para o Problema de Roteamento de Veículos, onde uma economia pode ser traduzida como a diminuição da distância (custo) conseguida a partir da união de duas rotas. Uma economia pode ter sinal negativo, ou seja, ao mesclar duas rotas, a distância total a ser percorrida aumenta ao invés de diminuir.

Essa heurística pode ser utilizada para gerar soluções para o Problema do Caixeiro Viajante. Para facilitar o entendimento, procuramos associar a descrição do algoritmo com a figura seguinte, usada para exemplificá-lo. Os passos da heurística são os seguintes:

Heurística das Economias

1. Inicie pelo vértice *c*, selecionado por algum critério ou aleatoriamente;
2. Para cada um dos demais vértices do grafo, crie rotas que partem de *c*, passam por um diferente vértice, e retornam à origem *c*. Após esse passo teremos (*n* vértices - 1) rotas (ver célula A da figura 3.7).
3. Obtenha a lista de economias relacionando o vértice *c* e todos os possíveis pares de vértices, diferentes de *c*, restantes do grafo. Por exemplo, para o par de vértices *e* e *f* a economia é calculada da seguinte forma: $E = Cec + Ccf - Cef$ onde *E* é a economia feita se o vértice *e* for ligado ao vértice *f* sem passar pelo vértice inicial *c*.
4. Ordene a lista de economias em ordem decrescente (da maior para a menor economia).
5. Percorra a lista de economias iniciando pela primeira posição e tente unir as rotas, por exemplo, através da inserção da aresta (e, f) e da remoção da aresta (e, c) e (c, f).
6. Se não for possível unir as rotas expandindo o circuito em formação, tente com a próxima economia da lista.
7. Repita o procedimento até a formação do circuito hamiltoniano.

Segundo Goldbarg e Luna [3], a aplicação eficiente dessa heurística para o Problema do Caixeiro Viajante sugere a necessidade de um grafo completo, (onde cada vértice possui ligação com todos os demais vértices existentes no grafo), sendo portanto uma abordagem bastante razoável para o PCV simétrico e euclidiano.

Podemos exemplificar os passos do algoritmo com o mesmo cenário utilizado anteriormente. Para o cálculo das economias são utilizados os seguintes custos: C_{ab}=15, C_{ac}=25, C_{ad}=36, C_{ae}=37, C_{af}=56, C_{bc}=13, C_{bd}=23, C_{be}=29, C_{bf}=45, C_{cd}=11, C_{ce}=17, C_{cf}=32, C_{de}=18, C_{df}=25, C_{ef}=20. Os valores apresentados correspondem à distância aproximada, em milímetros, entre os centros dos vértices do grafo.

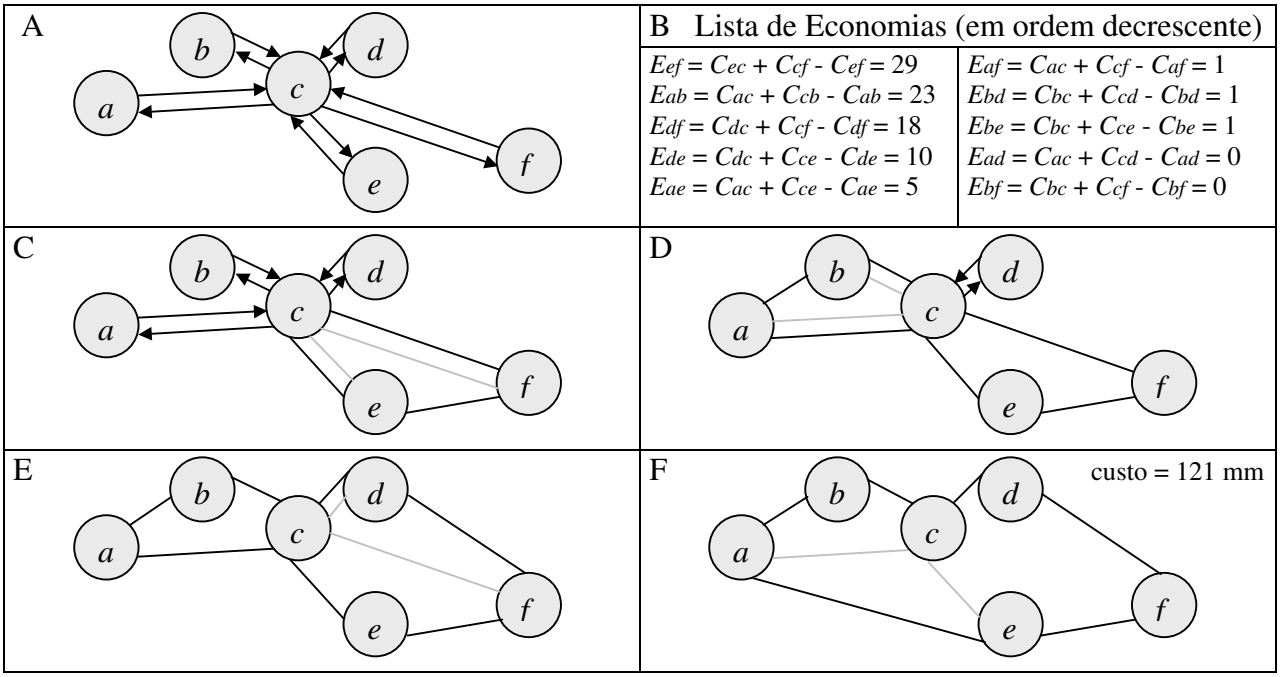


Figura 3.7 - Heurística das Economias passo a passo.

Observando a figura 3.7 podemos verificar a evolução do algoritmo. Após o cálculo e ordenação da lista de economias, a heurística inicia a formação da rota a partir da utilização da maior economia, como mostrado na célula C. No desenho, as linhas claras correspondem às arestas que foram removidas. O algoritmo prossegue percorrendo a lista de economias e incluindo partes do circuito a ser montado. Após o uso da economia *Edf*, mostrado na célula E, a seqüência natural indica a utilização da *Ede*, contudo, não é possível utilizar tal economia devido ao fato do vértice *f* já estar incluído no circuito em formação. Esse tipo de verificação é realizado pelo passo 6 que impede a utilização desta economia, selecionando a próxima da lista para

análise. No final da execução, o algoritmo apresenta um circuito hamiltoniano de custo igual a 121 mm, valor este superior em qualidade aos das duas heurísticas apresentadas anteriormente.

Também implementamos os passos dessa heurística no módulo responsável pelas simulações. Utilizando o mesmo cenário proposto para as heurísticas anteriores, podemos observar o resultado da heurística das economias através da figura mostrada a seguir. No apêndice A, podemos observar a ordem em que os municípios devem ser visitados.

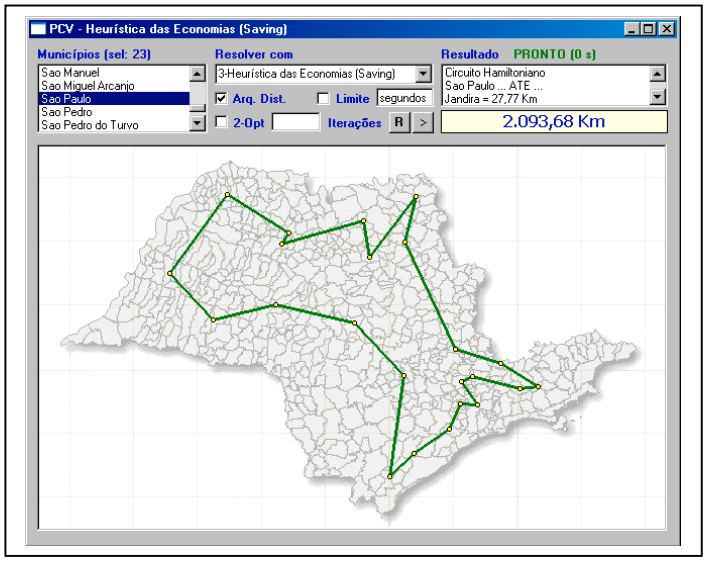


Figura 3.8 - Simulação da Heurística das Economias
(capital e municípios do estado de SP iniciados pela letra J)

Comparando o resultado dessa simulação com o das anteriores, notamos que a distância total a ser percorrida é menor na solução apresentada pela heurística das economias.

3.1.4 Heurística λ-Opt

A heurística λ-Opt faz parte do grupo das heurísticas de melhoria, que são estratégias que partem de um circuito hamiltoniano inicial e tentam melhorá-lo a partir de trocas na ordem em que os vértices serão visitados pelo caixeiro viajante. Para realizar tais trocas, Lin [11] propôs que fossem procuradas arestas no circuito inicial para serem retiradas e substituídas por outras que gerassem um circuito hamiltoniano de menor custo.

Lin [11] encontrou soluções que passou a chamar de λ ótimas, para λ = 2, 3, ..., *n* arestas ou arcos do grafo. Por exemplo, uma solução é considerada "2 ótima" (2-Opt) quando não conseguimos mais melhorá-la substituindo dois arcos do circuito por outros dois arcos. Já uma

solução é considerada 3-Opt quando não conseguimos melhorá-la através da substituição de três ou duas arestas do circuito. Generalizando, uma solução é considerada λ-Opt quando não conseguimos melhorá-la substituindo λ arestas do circuito; uma solução λ-Opt é ainda uma solução λ'-Opt, onde λ' < λ. Podemos descrever os passos desse algoritmo da seguinte forma:

Heurística λ-Opt

1. Obtenha um circuito hamiltoniano a partir de uma heurística qualquer;
2. Remova λ arestas do circuito inicial;
3. Teste todas as possíveis combinações de λ arestas para a reconstrução do circuito hamiltoniano;
4. Se for encontrado um conjunto de λ arestas que remontam o circuito com tamanho menor que o anterior, efetive a troca;
5. Retorne ao passo 2 até que nenhuma melhoria possa ser obtida.

Para casos em que λ é maior do que 3, são verificadas todas as possíveis substituições de λ arestas até não existir mais nenhuma troca que melhore a solução. Entretanto, o número de operações necessárias para esse procedimento cresce rapidamente com o aumento do número de vértices do grafo. Diante disso, valores de λ=2 e λ=3 são mais utilizados, sendo que soluções obtidas com λ=3 são geralmente melhores do que as obtidas com λ=2. Na figura seguinte podemos observar exemplos de movimentos de troca utilizados nas heurísticas 2-Opt e 3-Opt.

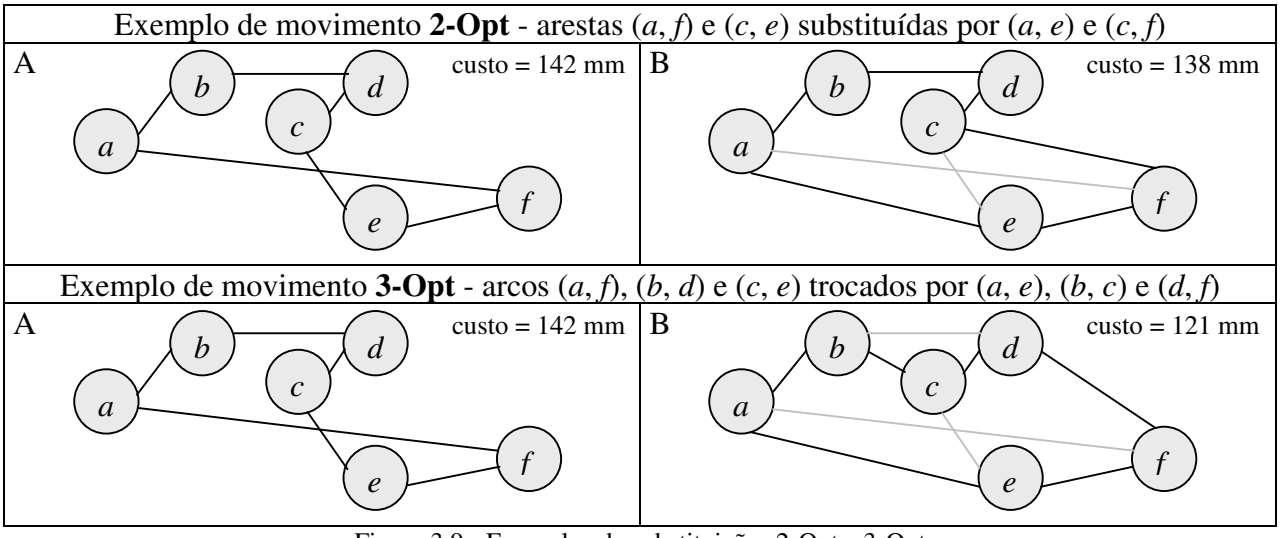


Figura 3.9 - Exemplos de substituições 2-Opt e 3-Opt

As linhas claras correspondem às arestas que foram substituídas durante o procedimento de troca. Para ilustrar o efeito de uma heurística de melhoria, implementamos a 2-Opt e realizamos a simulação, sendo que o circuito inicial foi gerado a partir do uso da heurística de Bellmore e Nemhauser mostrado na figura 3.4.

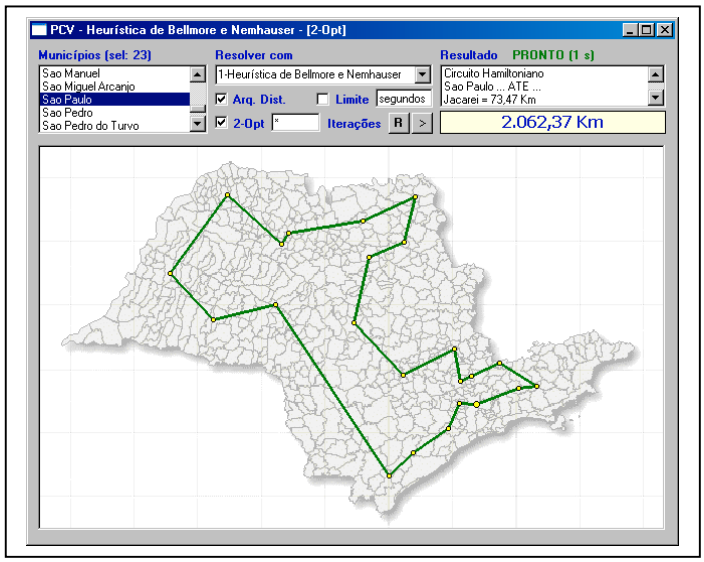


Figura 3.10 - Simulação da Heurística 2-Opt
(capital e municípios do estado de SP iniciados pela letra J)

Comparando o circuito inicial (figura 3.4) com o circuito apresentado após a aplicação da heurística de melhoria (figura 3.10), é possível verificar a eficiência deste procedimento. Podemos aplicar a heurística 2-Opt utilizando como circuitos iniciais os resultados obtidos através das outras heurísticas apresentadas. Com isso podemos encontrar soluções diferentes da apresentada na figura 3.10, isso porque estamos partindo de circuitos iniciais diferentes para o processo de melhoria.

3.2 Método de Planos de Corte

Os principais algoritmos exatos para resolução do PCV são baseados em formulações do problema usando o modelo de Programação Linear Inteira e/ou no método *Branch and Bound* (Chong [9]). Em Goldbarg e Luna [3] encontramos uma formulação apresentada por Dantzig, Fulkerson e Johnson em 1954, descrita da seguinte maneira:

$$\text{Minimizar} \quad z = \sum_{j=1}^{n}\sum_{i=1}^{n} CijXij$$

$$\text{sujeito a} \begin{cases} \sum_{i=1}^{n} Xij = 1 & \forall j \in V, \quad (1) \\ \sum_{j=1}^{n} Xij = 1 & \forall i \in V, \quad (2) \\ \sum_{i,j \in S} Xij \leq |S| - 1 & \forall S \subset V, \quad (3) \\ Xij \in \{0,1\} & \forall \mathrm{i},j \in V. \quad (4) \end{cases}$$

A variável binária *Xij* assume valor igual a 1 se a aresta (*i*,*j*), pertencente às arestas do grafo *G*=(*V*,*A*), for escolhida para integrar a solução, ou 0 em caso contrário. *S* é um subgrafo de *G*, onde |*S*| representa o número de vértices desse subgrafo. As restrições (1) e (2) indicam que exatamente uma aresta (*i*,*j*) emana de cada vértice, e exatamente uma aresta (*i*,*j*) é direcionada para cada vértice. O conjunto de restrições (3) é responsável pela eliminação de subcircuitos, ou seja, é responsável por apresentar uma solução conexa. Já o grupo (4) define o problema de Programação Linear Inteira. Para o PCV simétrico, podemos utilizar a formulação encontrada em Balas e Toth [12] que pode ser descrita da seguinte forma:

$$\text{Minimizar} \quad z = \sum_{a \in A} CaXa$$

$$\text{sujeito a} \begin{cases} \sum_{a \in \delta(v)} Xa = 2 & \forall v \in V, \quad (5) \\ \sum_{a \in \delta(S)} Xa \geq 2 & \forall S \subset V,\ S \neq 0, \quad (6) \\ Xa \in \{0,1\} & \forall a \in A. \quad (7) \end{cases}$$

A variável binária *Xa* assume valor igual a 1 se a aresta (*a*) pertence ao circuito da solução, ou 0 em caso contrário. A restrição (5) indica que existem duas arestas da solução que incidem em cada vértice (uma para entrar e outra para sair). O conjunto de restrições (6) indica que a solução deve ser conexa, enquanto que, as restrições (7) são as de integralidade.

Loparic [1] menciona que, na ocasião, os três pesquisadores descreveram um método de resolução do PCV que utilizava o algoritmo Simplex de resolução de problemas de programação linear, idealizado por um deles (Dantzig) em 1947. Para poder utilizar o Simplex, os pesquisadores tiveram que elaborar uma estratégia pois não seria possível utilizá-lo em conjunto com a formulação apresentada. Isso porque o Simplex não aceita as restrições de integralidade (7), e também porque o número de restrições do tipo (6) é muito elevado, e isso inviabiliza que se monte um programa linear com todas elas, para ser resolvido pelo Simplex.

Diante disso, Dantzig, Fulkerson e Johnson elaboraram uma estratégia que é considerada uma das versões pioneiras do que passou a ser chamado de Método de Planos de Corte. Tal estratégia pode ser descrita pelos seguintes passos:

1. Monte o programa linear apenas com as restrições do tipo (5), incluindo também restrições do tipo: $0 \leq Xa \leq 1, \forall a \in A$;
2. Resolva o programa linear com o uso do Simplex;
3. Se for notada alguma violação de uma restrição de eliminação de subcircuitos, adicione tal restrição ao programa linear e retorne ao passo 2.

Loparic [1] relata que o cenário escolhido para testar a estratégia, era composto de 49 cidades (a capital dos Estados Unidos, Washington D.C, e as capitais dos 48 estados da época). Ele menciona que os autores obtiveram 24 restrições de eliminação de subcircuitos, adicionadas ao programa linear conforme a estratégia descrita, mas que em seguida o Simplex devolveu uma solução fracionária para a qual eles não conseguiram encontrar nehuma restrição de eliminação de subcircuitos violada. Loparic [1] menciona ainda que depois disso, eles conseguiram prosseguir graças à ajuda de Glickberg da Rand Corporation, que encontrou restrições que estavam violadas pela solução obtida; com a inclusão de duas dessas restrições, os pesquisadores obtiveram a solução exata do PCV de 49 cidades.

A estratégia de descrever parcialmente o problema, como a que foi utilizada, é chamada de relaxação do problema, justamente por utilizar um conjunto de restrições mais fracas, ou seja, restrições que são satisfeitas não apenas por pontos do problema, mas eventualmente por pontos inviáveis. Existem muitos pontos que satisfazem as restrições do problema. O conjunto desses pontos define um poliedro. Quando relaxamos o problema, obtemos um poliedro Q que contém em seu interior o poliedro P do PCV. A inserção de uma restrição violada, é como um plano que corta, reduz, o poliedro Q que está sendo trabalhado pelo Simplex.

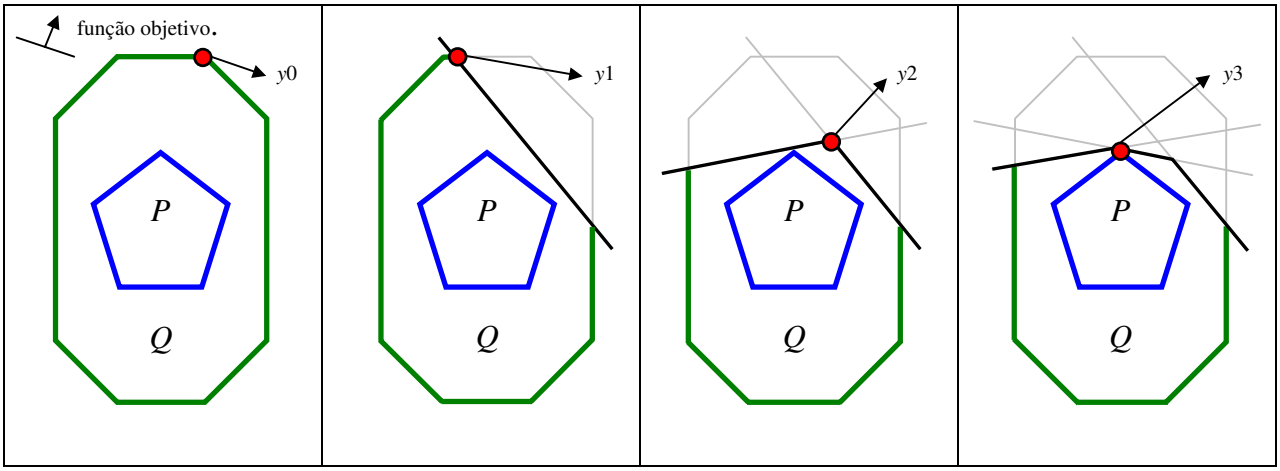


Figura 3.11 – Exemplo de Planos de Corte

Após a inclusão de um plano de corte, o programa linear é resolvido novamente, gerando uma solução ótima yx de Q. Se a solução yx pertencer também ao poliedro P, encontramos a solução ótima do PCV. De uma maneira geral, podemos escrever os passos do Método de Planos de Corte da seguinte forma:

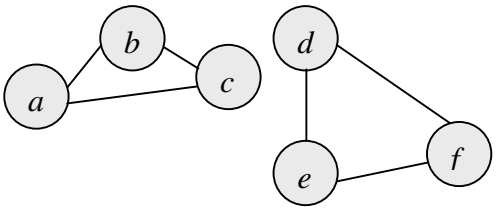
Método de Planos de Corte

1. Inicie por um poliedro Q (descrito por poucas desigualdades) que contém o poliedro P;
2. Encontre a solução ótima yx de Q;
3. Se $yx \notin P$, adicione a Q uma desigualdade válida (plano de corte) que separa (corta) yx de P sem perder nenhuma solução de P, e retorne ao passo 2.

Na figura seguinte utilizamos o cenário das exemplificações anteriores para ilustrar a execução dos passos desse método. As arestas são as seguintes: $\boldsymbol{a}=(a,b)$, $\boldsymbol{b}=(a,c)$, $\boldsymbol{c}=(a,d)$, $\boldsymbol{d}=(a,e)$, $\boldsymbol{e}=(a,f)$, $\boldsymbol{f}=(b,c)$, $\boldsymbol{g}=(b,d)$, $\boldsymbol{h}=(b,e)$, $\boldsymbol{i}=(b,f)$, $\boldsymbol{j}=(c,d)$, $\boldsymbol{k}=(c,e)$, $\boldsymbol{l}=(c,f)$, $\boldsymbol{m}=(d,e)$, $\boldsymbol{n}=(d,f)$, $\boldsymbol{o}=(e,f)$. Inicialmente, inserimos as restrições do tipo (5) e as restrições $0 \le X\boldsymbol{a} \le 1$, $\forall \boldsymbol{a} \in A$.

$$\min \quad C\boldsymbol{a}X\boldsymbol{a} + C\boldsymbol{b}X\boldsymbol{b} + \ldots + C\boldsymbol{o}X\boldsymbol{o}$$

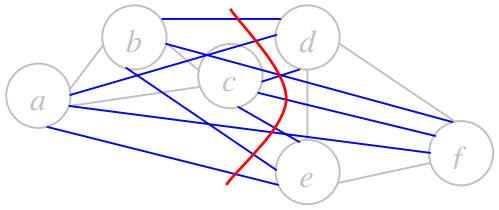
$$\text{s.a.} \begin{cases} X\boldsymbol{a} + X\boldsymbol{b} + X\boldsymbol{c} + X\boldsymbol{d} + X\boldsymbol{e} = 2 & \text{vértice } a \\ X\boldsymbol{a} + X\boldsymbol{f} + X\boldsymbol{g} + X\boldsymbol{h} + X\boldsymbol{i} = 2 & \text{vértice } b \\ X\boldsymbol{b} + X\boldsymbol{f} + X\boldsymbol{j} + X\boldsymbol{k} + X\boldsymbol{l} = 2 & \text{vértice } c \\ X\boldsymbol{c} + X\boldsymbol{g} + X\boldsymbol{j} + X\boldsymbol{m} + X\boldsymbol{n} = 2 & \text{vértice } d \\ X\boldsymbol{d} + X\boldsymbol{h} + X\boldsymbol{k} + X\boldsymbol{m} + X\boldsymbol{o} = 2 & \text{vértice } e \\ X\boldsymbol{e} + X\boldsymbol{i} + X\boldsymbol{l} + X\boldsymbol{n} + X\boldsymbol{o} = 2 & \text{vértice } f \\ 0 \le X\boldsymbol{a} \le 1, \ldots, 0 \le X\boldsymbol{o} \le 1 \end{cases}$$



solução inicial que viola restrição do tipo (6)

Figura 3.12a – Método de Planos de Corte – exemplo

Na figura 3.12a, observamos que a solução inicialmente calculada não se trata de um circuito hamiltoniano. O próximo passo é identificar a restrição de subcircuito violada e inserir uma desigualdade válida no programa linear para submetê-lo novamente ao Simplex.



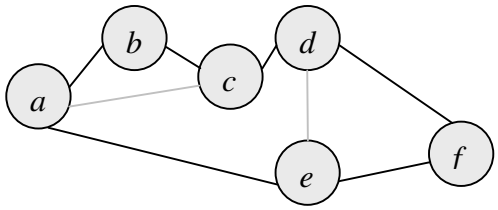
Para conectar os subcircuitos, observe que pelo menos duas arestas, em destaque, devem fazer parte da solução. Com essas arestas montamos uma nova desigualdade para o programa linear.

Figura 3.12b – Método de Planos de Corte – exemplo

Identificada a restrição violada, inserimos a desigualdade correspondente (plano de corte) e resolvemos novamente o programa linear. A desigualdade e a nova solução podem ser visualizadas na figura seguinte.

$$\min \quad CaXa + CbXb + \ldots + CoXo$$

$$s.a. \begin{cases} Xa + Xb + Xc + Xd + Xe = 2 \\ Xa + Xf + Xg + Xh + Xi = 2 \\ Xb + Xf + Xj + Xk + Xl = 2 \\ Xc + Xg + Xj + Xm + Xn = 2 \\ Xd + Xh + Xk + Xm + Xo = 2 \\ Xe + Xi + Xl + Xn + Xo = 2 \\ 0 \le Xa \le 1, \ldots, 0 \le Xo \le 1 \\ Xc + Xd + Xe + Xg + Xh + Xi + Xj + Xk + Xl \ge 2 \end{cases}$$



solução ótima
circuito hamiltoniano de custo 121 mm

Figura 3.12c – Método de Planos de Corte – exemplo

Como feito para as estratégias anteriores, implementamos o Método de Planos de Corte no programa responsável pelas simulações deste capítulo, para ilustrar o efeito da aplicação dessa estratégia em um cenário mais próximo do leitor.

Para essa parte do programa, utilizamos uma implementação do método Simplex contida na biblioteca GLPK (*GNU Linear Programming Kit*) versão 3.2.1. Essa biblioteca disponibiliza um conjunto de funções e métodos para a resolução de problemas com o modelo de Programação Linear. Versões mais recentes da GLPK podem ser obtidas acessando o endereço na *internet* (http://www.gnu.org).

Utilizando o cenário composto pela capital São Paulo e pelos municípios do estado iniciados pela letra J, encontramos uma solução melhor que as anteriormente apresentadas, (solução ótima).

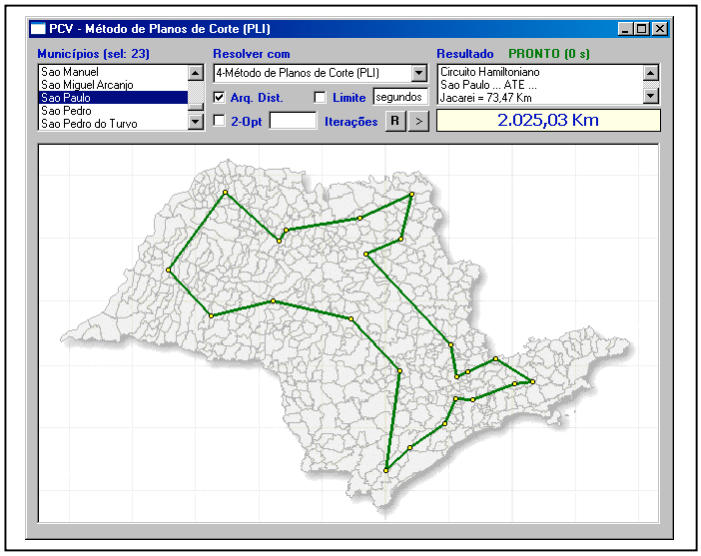


Figura 3.13 - Simulação do Método de Planos de Corte
(capital e municípios do estado de SP iniciados pela letra J)

Os trechos da solução apresentada na figura 3.13, bem como a composição dos circuitos hamiltonianos de todas as outras estratégias abordadas neste capítulo, podem ser consultados no Apêndice A. No Apêndice D, apresentamos testes com outras instâncias, envolvendo todas as estratégias apresentadas neste trabalho para o Problema do Caixeiro Viajante.

4 O Problema de Roteamento de Veículos

O Problema de Roteamento de Veículos (PRV) é um dos problemas de Otimização Combinatória que desperta interesse devido, em parte, à sua grande importância prática. O objetivo do PRV é encontrar, em um grafo completo $G=(V,A)$, rotas para veículos minimizando os custos de transporte de produtos a serem entregues a um conjunto de clientes ou consumidores.

No cenário do PRV, os vértices do grafo G podem assumir os papéis de depósitos ou de postos de entrega/coleta de produtos. As arestas, representam estradas, ruas, que conectam os vértices fornecendo as distâncias entre os mesmos. As demandas, que podem estar tanto nos vértices como nas arestas, devem ser supridas por veículos que possuem capacidade própria. Em alguns casos, as entregas devem ser realizadas em tempos estipulados, respeitando também horários de chegada ao local. Em algumas variações, são computados o tempo total de viagem e tempo para eventuais pausas.

Este cenário pode se tornar complexo com a inclusão de diversas restrições, o que torna o problema difícil de ser resolvido. Quando eliminamos tais restrições e reduzimos a frota para um veículo que deve atender a todos os postos a partir de um único depósito, estamos tratando do Problema do Caixeiro Viajante. Apesar dessa estreita relação, o PRV é alvo de estudos mais recentes.

Chong [9] cita que o PRV foi inicialmente apresentado por Garvin et al. [13] em 1957 para auxiliar no processo de distribuição de gasolina para postos de serviço, utilizando uma frota com veículos de capacidades distintas. Na ocasião, Garvin et al. [13] utilizaram o modelo de Programação Linear para resolução de uma variedade de problemas nas áreas de perfuração e produção, manufatura, marketing e distribuição da indústria petroquímica.

Existem diversas variações do PRV, umas com múltiplos depósitos, outras com restrições de tempo, algumas utilizando frotas com veículos de capacidades diferentes, outras em que as demandas se encontram nas arestas, como por exemplo, os problemas de coleta de lixo e distribuição de jornais. Neste trabalho nos concentramos no estudo do PRV com um único depósito, demandas nos vértices, frota homogênea (veículos com mesma capacidade) e sem

restrições de tempo, onde nosso objetivo é encontrar um conjunto de rotas minimizando a distância total a ser percorrida obedecendo as seguintes restrições:

- cada rota começa e termina no depósito.
- todo município, ou posto de entrega, é visitado apenas uma vez e somente por um veículo.
- a demanda total de qualquer rota não deve superar a capacidade de um veículo.

Como no capítulo 3, para auxiliar na ilustração dos métodos e algoritmos deste capítulo, implementamos um programa que realiza simulações do PRV, utilizando como vértices os municípios do estado de São Paulo.

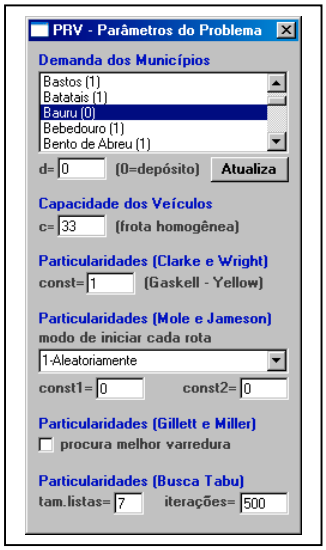


Figura 4.1 – Tela de Parâmetros

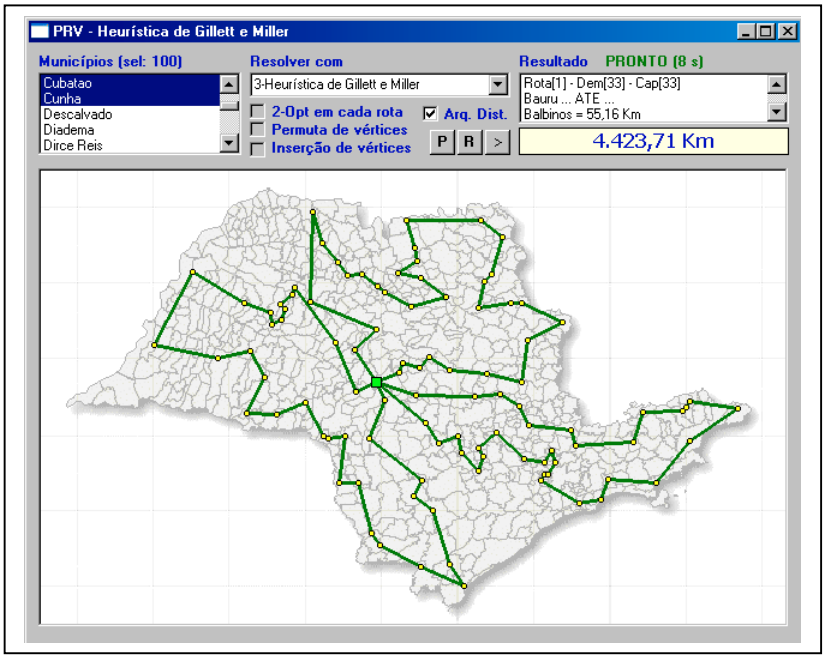


Figura 4.2 – Exemplo de Simulação do PRV
(municípios do estado de SP iniciados pelas letras B e C. Depósito=Bauru)

Na figura 4.1, temos a ilustração da tela de parâmetros do problema. A partir dessa tela atualizamos as demandas dos municípios, a capacidade dos veículos da frota e informamos particularidades das estratégias de resolução do problema. Na figura 4.2, temos um exemplo da execução do programa, onde foi indicado que a demanda de cada vértice é igual a 1 e que a capacidade dos veículos é igual a 33. O município de Bauru foi selecionado como depósito e a heurística de Gillett e Miller foi escolhida para resolver o problema. A caixa de texto localizada no canto direito, sobre o mapa, nos mostra a distância total a ser percorrida e acima dela, uma lista apresenta informações sobre as rotas que fazem parte da solução.

4.1 Algoritmos Heurísticos

Existem muitas heurísticas propostas para o Problema de Roteamento de Veículos. Essas heurísticas podem ser classificadas em duas classes principais: a das heurísticas clássicas, sendo a maioria desenvolvida entre os anos de 1960 e 1990, e a das metaheurísticas cuja evolução ocorreu a partir da década de 90. A classe das heurísticas clássicas conta com procedimentos que realizam uma exploração limitada no espaço das soluções e fornecem bons resultados com pouco tempo de processamento. Tais procedimentos são amplamente utilizados em *softwares* comerciais e a maior parte deles podem ser facilmente alterados para atender as restrições encontradas em cenários reais. Já a classe das metaheurísticas conta com procedimentos que realizam uma exploração profunda nas regiões mais promissoras do espaço das soluções. Tal exploração consome tempo de processamento, mas permite que esses procedimentos forneçam resultados melhores que os das heurísticas clássicas (Laporte e Semet [14]).

As heurísticas clássicas podem ser divididas em três categorias:

- Heurísticas construtivas – são procedimentos que constróem uma solução a partir de inserções gradativas de vértices nas rotas ou a partir da união de rotas existentes usando o critério de economias.
- Heurísticas de duas fases – são procedimentos que dividem o problema em duas etapas, uma visando reunir ou agrupar os clientes, outra para gerar as rotas para os veículos. A ordem das etapas pode ser agrupar-rotear e rotear-agrupar.
- Heurísticas de melhoria – são procedimentos que partem de uma solução inicial e procuram melhorá-la a partir de trocas de arestas ou vértices entre rotas.

No início deste capítulo apresentamos alguns dos procedimentos relacionados com a classe das heurísticas clássicas. Em seguida destacamos a classe das metaheurísticas demonstrando um de seus procedimentos, a Busca Tabu.

4.1.1 Heurística de Clarke e Wright

A heurística de Clarke e Wright [10] é uma das heurísticas mais conhecidas para o PRV. Ela parte de uma solução inicial, contendo várias rotas onde cada cliente é atendido individualmente por um veículo, e em seguida procura unir essas rotas a partir do conceito das economias apresentado no capítulo 3.

Ela aplica-se naturalmente a problemas em que o número de veículos não é estabelecido, e pode ser utilizada tanto na versão seqüencial quanto na versão paralela (Laporte e Semet [14]). À medida em que as rotas vão sendo expandidas rumo a uma solução de melhor qualidade, é possível realizar os testes relativos às restrições do problema (Goldbarg e Luna [3]). Os passos do algoritmo na sua versão seqüencial podem ser descritos da seguinte forma:

Heurística de Clarke e Wright (versão seqüencial)

1. Inicie pelo vértice $V0$, que corresponde ao depósito;
2. Para todos os demais vértices do grafo, crie pequenas rotas saindo de $V0$, passando por exemplo por Va e retornando à origem $V0$.
3. Obtenha a lista de economias relacionando o vértice $V0$ e todos os possíveis pares de vértices restantes do grafo. Por exemplo, para o par de vértices Va e Vu a economia é calculada da seguinte forma: $E = Ca0 + C0u - Cau$ onde E é a economia feita se o vértice Va for ligado ao vértice Vu sem passar pelo vértice inicial $V0$.
4. Ordene a lista de economias em ordem decrescente (da maior para a menor economia).
5. Eleja uma rota que possa ser expandida.
6. Percorra a lista de economias, iniciando pela primeira da lista, e selecione a primeira economia que pode ser usada para expandir a rota eleita no passo anterior, respeitando as restrições do problema.
7. Efetue a união das rotas, por exemplo, através da inserção da aresta (Va, Vu) e da remoção da aresta (Va, $V0$) e ($V0$, Vu), expandindo com isso a rota eleita no passo 5; remova a economia utilizada da lista de economias.
8. Se a rota em questão puder ainda ser expandida volte ao passo 6.
9. Se existirem rotas que possam ser expandidas volte ao passo 5.

Enquanto a versão seqüencial percorre várias vezes a lista de economias na tentativa de expandir uma única rota antes de iniciar a expansão da próxima, a versão paralela percorre a lista de economias apenas uma vez, tentando realizar a expansão de uma ou mais rotas até que a lista seja esgotada. Na prática, resultados obtidos através da versão paralela são superiores em qualidade aos obtidos pelo uso da versão seqüencial. Os passos do algoritmo da versão paralela podem ser descritos da seguinte forma:

Heurística de Clarke e Wright (versão paralela)

1. Idêntico ao da versão seqüencial;
2. Idêntico ao da versão seqüencial;
3. Idêntico ao da versão seqüencial;
4. Idêntico ao da versão seqüencial;
5. Percorra a lista de economias iniciando pela primeira posição e tente unir as rotas, por exemplo, através da inserção da aresta (*Va*, *Vu*) e da remoção da aresta (*Va*, *V*0) e (*V*0, *Vu*); lembrando que as restrições do problema devem ser respeitadas.
6. Se não for possível usar a economia selecionada para unir as rotas, tente com a próxima economia da lista;
7. Repita o procedimento até que a lista seja percorrida até o final.

Para ilustrar uma solução, implementamos a versão paralela desta heurística no simulador. No exemplo seguinte, bem como nos outros apresentados neste capítulo, utilizamos os 35 primeiros municípios de SP iniciados pela letra A; o depósito escolhido está localizado no município de Agudos e as demandas dos municípios consumidores são de 130 unidades de um determinado produto. Os veículos da frota possuem capacidade para transportar 1000 unidades do produto.

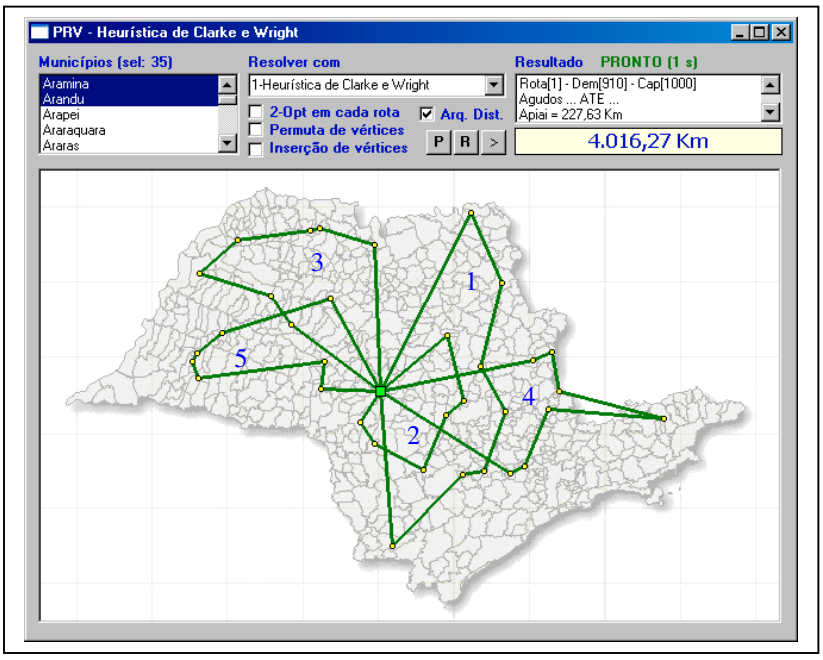


Figura 4.3 – Simulação da Heurística de Clarke e Wright
(35 primeiros municípios de SP, em ordem alfabética, iniciados pela letra A. Depósito=Agudos)

Observando a figura anterior, podemos notar a formação de cinco rotas que totalizam a distância de 4.016,27 Km a serem percorridos pelos veículos da frota. No Apêndice B podemos consultar os detalhes da composição de cada rota.

Para o exemplo apresentado, o algoritmo gerou 34 rotas através do seu segundo passo. A partir do momento em que as rotas vão sendo unidas, a distância total a ser percorrida pela frota vai diminuindo. Quando uma rota vai se expandindo, ou seja, vai sendo mesclada com outras, alguns vértices passam a ficar desligados do depósito; tais vértices são chamados de vértices internos e não são mais candidatos a melhoria na rota, uma vez que para união das rotas só são testados os vértices externos, ou seja, aqueles que possuem ligação com o depósito. Em alguns momentos, isso pode ser encarado como uma fragilidade do algoritmo (Goldbarg e Luna [3]).

Observando a figura 4.3, notamos que a rota de número 1 parece não fazer parte do conjunto de boas rotas a serem adotadas pela solução. Laporte e Semet [14] mencionam que um dos pontos fracos do algoritmo original de Clarke e Wright é que ele tende a produzir boas rotas de início mas ao se aproximar do final, realiza uniões que podem produzir rotas menos interessantes. Além disso, para algumas instâncias o algoritmo pode consumir um perceptível tempo de processamento para calcular, ordenar e armazenar a lista de economias.

Diante disso, vários autores trabalharam em avanços para essa estratégia. Gaskell [15] e Yellow [16], por exemplo, apresentaram trabalhos relacionados com uma constante θ introduzida no cálculo das economias da seguinte maneira: $Eij = Ci0 + C0j - \theta Cij$. Alterando o valor dessa constante é possível gerar rotas de diferentes formatos.

Outros avanços foram propostos, tendo em vista a agilidade da solução. Golden, Magnanti e Nguyen [17] apresentaram um trabalho que, além de um estudo sobre o PRV, contém a proposta de um método que armazena as economias numa estrutura de dados denominada *heap* (árvore binária balanceada, na qual o valor do nó pai é sempre maior ou igual ao valor dos nós filhos). Quando duas rotas são unidas, a *heap* é reconstruída de maneira eficiente, eliminando as economias relacionadas com a última união e todas as economias associadas a um vértice interior de uma rota. Esse procedimento dinamiza o processo de escolha das próximas economias.

Posteriormente, Nelson et al. [18] apresentaram métodos que utilizam estruturas de dados complexas baseadas em *heaps*, expandindo a idéia do trabalho citado anteriormente. Os autores mostraram formas diferentes de eliminação das arestas associadas aos vértices internos e compararam a aplicação desses procedimentos, incluindo também uma implementação direta da

heurística que armazena a lista de economias em vetores. O resultado da comparação demonstra a superioridade dos métodos propostos sobre a implementação original, e também a evolução deles próprios, sendo que o melhor procedimento encontrou a solução para uma instância de 1000 vértices em 184.54 segundos (na ocasião em um IBM 4341).

Para o nosso aplicativo, incluímos a constante θ observada nos trabalhos de Gaskell [15] e Yellow [16]. No exemplo a seguir, utilizamos o mesmo cenário da simulação anterior, só que alterando o valor da constante θ de 1.0 para 1.2 .

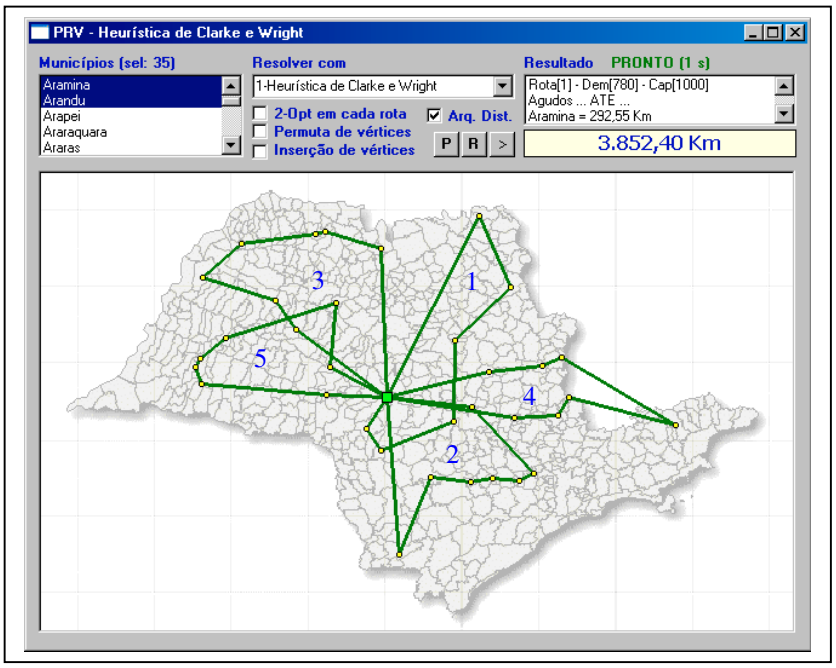


Figura 4.4 – Simulação da Heurística de Clarke e Wright com θ = 1.2
(35 primeiros municípios de SP, em ordem alfabética, iniciados pela letra A. Depósito=Agudos)

A seguir, incluímos informações detalhadas das rotas ilustradas na figura 4.4, pois elas serão comparadas com as dos procedimentos de melhoria, na seqüência do texto.

rota**[1]** . demanda**[780]** . trecho**[821,01 Km]**

Agudos → (292,55 Km) → Aramina → (112,55 Km) → Altinópolis → (108,84 Km) → Américo Brasiliense → (118,14 Km) → Anhembi → (102,26 Km) → Arandu → (34,13 Km) → Águas de Santa Bárbara → (52,54 Km) → **Agudos**

rota**[2]** . demanda**[910]** . trecho**[731,25 Km]**

Agudos → (227,63 Km) → Apiaí → (121,59 Km) → Angatuba → (52,87 Km) → Alambari → (29,53 Km) → Araçoiaba da Serra → (36,67 Km) → Alumínio → (22,57 Km) → Araçariguama → (125,06 Km) → Águas de São Pedro → (115,33 Km) → **Agudos**

rota**[3]** . demanda**[910]** . trecho**[793,64 Km]**

Agudos → (216,35 Km) → Altair → (79,68 Km) → Américo de Campos → (13,41 Km) → Álvares Florence → (102,09 Km) → Aparecida d'Oeste → (71,44 Km) → Andradina → (103,65 Km) → Araçatuba → (50,69 Km) → Alto Alegre → (156,33 Km) → **Agudos**

rota**[4]** . demanda**[910]** . trecho**[835,58 Km]**

Agudos → (172,84 Km) → Americana → (58,26 Km) → Amparo → (28,83 Km) → Águas de Lindóia → (149,82 Km) → Aparecida → (183,13 Km) → Águas da Prata → (29,83 Km) → Aguaí → (71,39 Km) → Analândia → (141,48 Km) → **Agudos**

rota**[5]** . demanda**[910]** . trecho**[670,92 Km]**

Agudos → (86,87 Km) → Álvaro de Carvalho → (94,19 Km) → Adolfo → (156,32 Km) → Adamantina → (45,69 Km) → Alfredo Marcondes → (14,79 Km) → Álvares Machado → (25,73 Km) → Anhumas → (167,78 Km) → Alvinlândia → (79,55 Km) → **Agudos**

Comparando as figuras 4.4 e 4.3 podemos perceber que as rotas geradas são diferentes, e que o resultado obtido pela simulação ilustrada na figura 4.4 é superior em qualidade. Quando aumentamos o valor da constante θ, maior ênfase é dada à distância entre os vértices a serem conectados (Laporte e Semet [14]).

4.1.2 Procedimentos de Melhoria

Os procedimentos de melhoria para o PRV operam em uma rota específica ou em várias rotas ao mesmo tempo. No primeiro caso, qualquer heurística de melhoria para o PCV pode ser aplicada, como por exemplo a heurística λ-Opt de Lin [11] apresentada no capítulo anterior. Já para o segundo caso, trabalhos como os de Dror e Levy [19], Salhi e Rand [20], Waters [21], Fahrion e Wrede [22], fornecem uma descrição de uma série de estratégias que utilizam combinações de métodos com algumas operações elementares de trocas de vértices entre rotas. Tais operações podem ser exemplificadas da seguinte forma:

- troca de vértices: dois grupos de *k* vértices são trocados entre duas rotas; no exemplo a seguir *k*=1 e a troca ocorre com o uso dos vértices *w* e *e*, sendo que as linhas claras correspondem aos arcos substituídos nas operações.

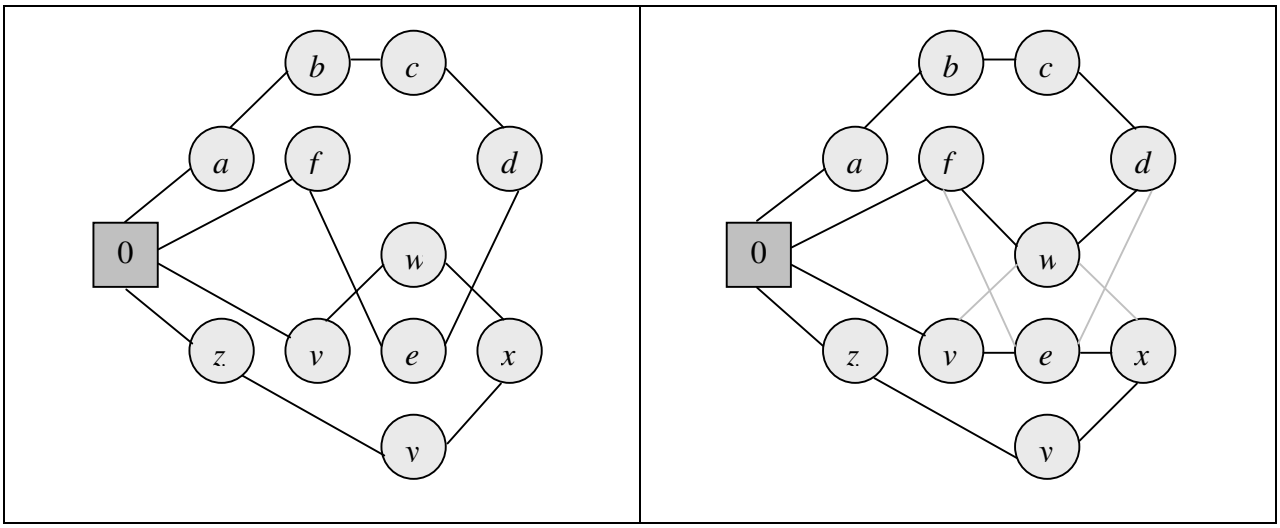


Figura 4.5 – Exemplo de troca de vértices entre rotas

- inserção de vértices: um grupo de *k* vértices é movido de uma para outra rota; no exemplo seguinte *k*=1 e a rota (*a*, *b*, *c*, *d*, *e*, *f*) cedeu, ou doou o vértice *e* para a outra. Novamente, as linhas claras correspondem aos arcos substituídos.

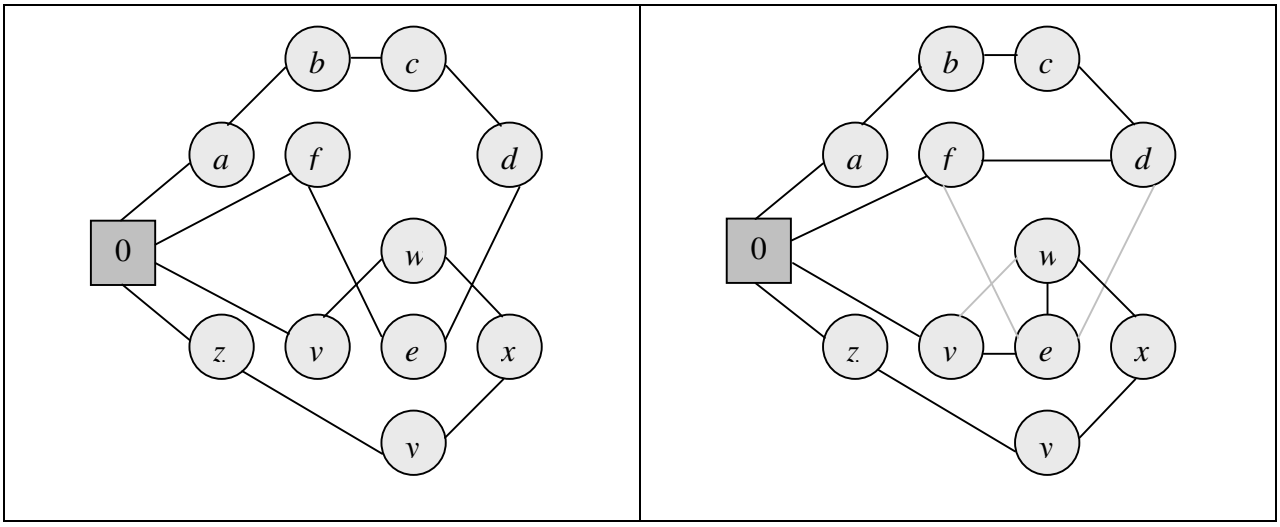


Figura 4.6 – Exemplo de inserção de vértice em rota

Nos trabalhos mencionados, Dror e Levy [19] dedicam parte do artigo para exemplificar os movimentos de trocas de vértices entre rotas e compará-los aos movimentos de trocas de arestas. Salhi e Rand [20] elaboraram uma rotina contendo seis métodos que executam as operações de troca de vértices de maneira diferenciada, utilizando também heurísticas λ-Opt, nas rotas individualmente, para evitar cruzamentos. Waters [21], além das operações ilustradas anteriormente, utilizou trocas de três vértices (um vértice de uma rota e um par de vértices adjacentes de outra), e quatro vértices (dois pares de vértices adjacentes de rotas diferentes). Ele combinou esses procedimentos e ordenou-os, com base em resultados obtidos, da seguinte maneira: 1. 2-Opt; 2. Troca de dois pares adjacentes; 3. Troca de um vértice mais um par de adjacentes; 4. Troca de vértices; 5. Inserção ou doação de vértice. Posteriormente, Fahrion e Wrede [22] apresentaram um procedimento baseado nas estratégias descritas por Waters [21]. Os autores combinaram as idéias básicas dessas estratégias em um procedimento que permite a troca de correntes de vértices de tamanhos variados.

Observando as figuras 4.5 e 4.6, notamos que para realizar a troca de um vértice de uma rota com um de outra rota é necessário remover quatro arestas e incluir quatro novas arestas. Para a operação de doação de vértice é necessário remover três arestas e incluir três novas arestas. De uma forma geral, para efetivar a operação devemos satisfazer a seguinte condição:

Custo das **arestas removidas** – Custo das **arestas incluídas** > 0

Para o programa responsável pelas simulações, implementamos três procedimentos de melhoria. Herdamos a implementação da heurística 2-Opt do simulador do PCV e elaboramos duas pequenas rotinas que combinam as operações de troca e inserção de vértices. Na primeira

rotina, fazemos para todos os vértices (com exceção do depósito) de todas as rotas, todas as possíveis tentativas de troca com k=1 vértice. O algoritmo pode ser descrito da seguinte forma:

Troca de vértices (k=1)

1. Gere uma solução inicial, e guarde em um vetor apenas a identificação de cada rota;
2. Percorra o vetor de rotas da primeira até a penúltima posição;
3. ... Para cada incremento do passo anterior, percorra o vetor de rotas do índice posterior ... ao obtido no passo 2 até a última posição;
4. ..... Caminhe pelos vértices da rota indicada pelo passo 2;
5. ..... Para cada incremento do passo 4, caminhe pelos vértices da rota do passo 3;
6. ..... Verifique a possibilidade de troca de rotas para os vértices indicados nos ..... passos 4 e 5 segundo ilustração anterior; se a troca for vantajosa e não ferir ..... nenhuma outra restrição do problema, efetive a operação.

Para a rotina de inserção de vértices identificamos as rotas como doadoras e receptoras. Cada rota terá o seu momento de doadora e tentará inserir um de seus vértices (com exceção do depósito) em uma das restantes rotas receptoras. O algoritmo pode ser descrito da seguinte forma:

Inserção de vértices (k=1)

1. Gere uma solução inicial, e guarde em um vetor apenas a identificação de cada rota;
2. Percorra o vetor de rotas da primeira até a última posição (rota doadora);
3. ... Para cada incremento do passo anterior, percorra o vetor de rotas da primeira até a ... última posição (rota receptora);
4. ..... Se índices dos contadores dos passos 2 e 3 forem diferentes, execute os passos ..... seguintes;
5. ..... Caminhe pelos vértices da rota doadora;
6. ..... Para cada incremento do passo 5, caminhe pelos vértices da rota receptora;
7. ..... Para cada vértice da rota doadora, procure na rota receptora um par de ..... vértices adjacentes que forneça a melhor posição para inserção do candidato, ... seguindo os critérios ilustrados anteriormente; se nenhuma restrição do ..... problema for violada e se a inserção se mostrar vantajosa, efetive a operação.

No programa, os procedimentos podem ser executados separadamente, mas se os três forem selecionados, a ordem de execução deles é a mesma sugerida pela tela: 1. 2-Opt para todas as rotas individualmente; 2. Troca ou Permuta de vértices; 3. Inserção de vértices. Caso ocorra uma troca ou inserção de vértices, executamos novamente a heurística 2-Opt para as rotas que participaram da operação.

Na figura seguinte, temos uma simulação da execução desses três procedimentos de melhoria, utilizando o cenário dedicado aos exemplos desse capítulo. Como ponto de partida, foi utilizada a solução ilustrada na figura 4.4 (Heurística de Clarke e Wright com $\theta = 1.2$).

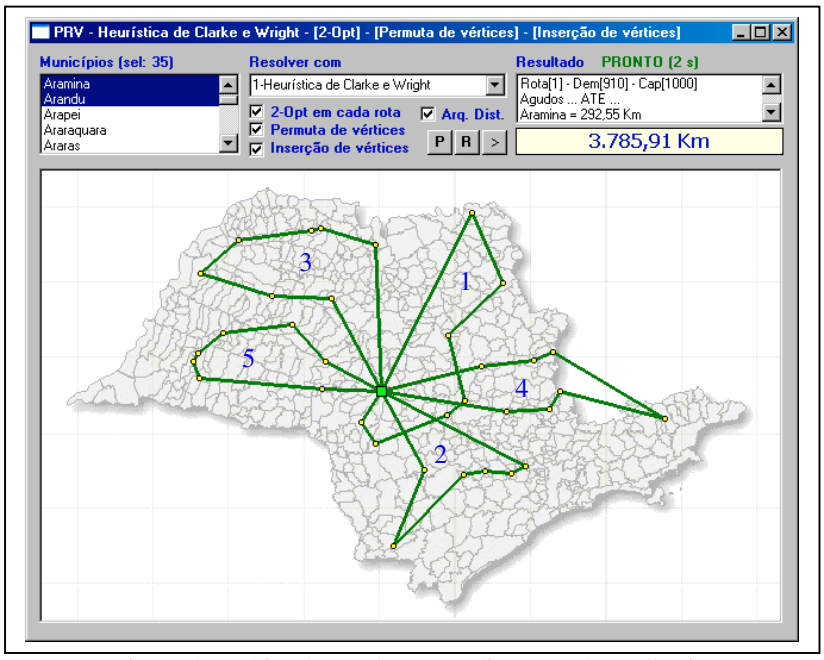


Figura 4.7 – Simulação dos Procedimentos de Melhoria
(35 primeiros municípios de SP, em ordem alfabética, iniciados pela letra A. Depósito=Agudos)

Após os procedimentos de melhoria, as rotas geradas possuem as seguintes configurações:

rota**[1]** . demanda**[910]** . trecho**[835,99 Km]**
Agudos → (292,55 Km) → Aramina → (112,55 Km) → Altinópolis → (108,84 Km) → Américo Brasiliense → (99,64 Km) → Águas de São Pedro → (33,48 Km) → Anhembi → (102,26 Km) → Arandu → (34,13 Km) → Águas de Santa Bárbara → (52,54 Km) → **Agudos**

rota**[2]** . demanda**[780]** . trecho**[705,99 Km]**
Agudos → (127,78 Km) → Angatuba → (121,59 Km) → Apiaí → (143,05 Km) → Alambari → (29,53 Km) → Araçoiaba da Serra → (36,67 Km) → Alumínio → (22,57 Km) → Araçariguama → (224,80 Km) → **Agudos**

rota**[3]** . demanda**[910]** . trecho**[821,18 Km]**
Agudos → (216,35 Km) → Altair → (79,68 Km) → Américo de Campos → (13,41 Km) → Álvares Florence → (102,09 Km) → Aparecida d'Oeste → (71,44 Km) → Andradina → (103,65 Km) → Araçatuba → (81,53 Km) → Adolfo → (153,03 Km) → **Agudos**

rota**[4]** - demanda**[910]** - trecho**[835,58 Km]**
Agudos → (172,84 Km) → Americana → (58,26 Km) → Amparo → (28,83 Km) → Águas de Lindóia → (149,82 Km) → Aparecida → (183,13 Km) → Águas da Prata → (29,83 Km) → Aguaí → (71,39 Km) → Analândia → (141,48 Km) → **Agudos**

rota**[5]** - demanda**[910]** - trecho**[587,17 Km]**
Agudos → (86,87 Km) → Álvaro de Carvalho → (71,90 Km) → Alto Alegre → (94,86 Km) → Adamantina → (45,69 Km) → Alfredo Marcondes → (14,79 Km) → Álvares Machado → (25,73 Km) → Anhumas → (167,78 Km) → Alvinlândia → (79,55 Km) → **Agudos**

Comparando os resultados das figuras 4.7 e 4.4, podemos notar que os procedimentos de melhoria cumpriram suas missões. A heurística 2-Opt alterou um trecho da rota 2 de [Agudos → **(227,63 Km)** → Apiaí → (121,59 Km) → Angatuba → **(52,87 Km)** → Alambari] para [Agudos → **(127,78 Km)** → Angatuba → (121,59 Km) → Apiaí → **(143,05 Km)** → Alambari], onde as arestas em destaque correspondem às utilizadas durante a troca, rendendo uma economia de 9,67 Km.

Em seguida, o procedimento de troca de vértices realizou alterações entre as rotas 3 e 5, e entre as rotas 1 e 2. No primeiro caso, o município de Alto Alegre que pertencia a rota 3, foi trocado com o município de Adolfo que pertencia a rota 5, gerando com isso uma economia de 56,21 Km. No segundo caso, o município de Anhembi que pertencia a rota 1, foi trocado com o município de Águas de São Pedro que pertencia a rota 2, gerando com isso uma economia de 0,58 Km ou 580 metros.

Posteriormente, o procedimento de inserção de vértices realizou uma alteração entre as rotas 1 e 2. Nessa alteração, a rota 2 devolveu o município de Anhembi para a rota 1, inserindo-o entre os municípios de Águas de São Pedro e Arandu, gerando com isso uma nova economia de 0,03 Km ou 30 metros. Somando as economias obtidas pelos procedimentos de melhoria (9,67 + 56,21 + 0,58 + 0,03) temos uma economia total de 66,49 Km, valor esse que também pode ser obtido através da diferença entre os resultados apresentados nas figuras 4.4 e 4.7.

4.1.3 Heurística de Mole e Jameson

Em 1976, Mole e Jameson [23] apresentaram uma heurística construtiva seqüencial, em que o critério utilizado para expandir uma rota pela inclusão de novos vértices conta com duas condições:

- $C1(i, k, j) = Dik + Dkj - \mu Dij$
- $C2(i, k, j) = \lambda D0k - C1(i, k, j)$

A primeira condição é responsável por encontrar a melhor posição para inserção de cada vértice candidato ainda não roteado. A segunda condição é responsável pela seleção do vértice que será incluído pelo movimento de expansão. Nas condições existem dois parâmetros, μ e λ, que podem receber valores diferentes, fazendo com que sejam geradas rotas de diferentes formatos e composições.

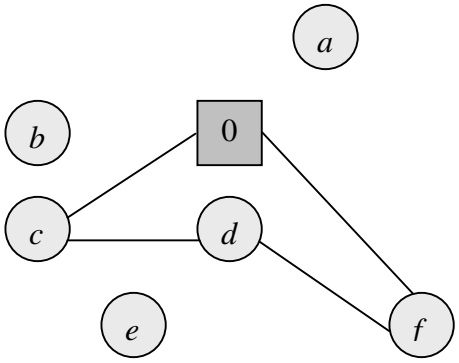
O algoritmo dessa estratégia pode ser descrito através dos seguintes passos:

Heurística de Mole e Jameson

1. Se todos os vértices já estiverem incluídos nas rotas formadas, pare; senão, inicie uma nova rota $(0, k, 0)$ onde k é um vértice que não está contido nas rotas existentes;
2. Para expandir a rota, encontre para cada vértice k não roteado o custo de inserção $C1^*_{(ik,\ k,\ jk)} = \min\{C1_{(r,\ k,\ s)}\}$ para todo par adjacente de vértices r e s da rota em formação, onde ik e jk fornecem o mínimo custo para a primeira condição; se não existirem inserções possíveis retorne ao passo 1;
3. Com os valores mínimos de C1 encontrados para cada vértice no passo anterior, procure o melhor vértice candidato k^* para inserção, satisfazendo a expressão $C2^*_{(ik^*,\ k^*,\ jk^*)} = \max\{C2_{(ik,\ k,\ jk)}\}$; insira o candidato k^* entre os vértices ik^* e jk^*.
4. Aplique uma heurística λ-Opt na rota em formação e retorne ao passo 2.

A figura seguinte mostra um exemplo de um movimento de expansão do algoritmo.

A



B $C1_{(i,\ k,\ j)} = Dik + Dkj - \mu Dij$ $(\mu=1)$
(procurando a melhor posição para inserção)
exemplo: $C1_{(0,\ a,\ c)} = D0a + Dac - \mu D0c$
$C1_{(0,\ a,\ c)} = 18 + 47 - 1*29 = 36$

$C1_{(0,\ a,\ c)} = 36$	$C1_{(c,\ b,\ d)} = 16$
$C1_{(0,\ a,\ f)} = 22$ min(a)	$C1_{(d,\ b,\ f)} = 58$
$C1_{(c,\ a,\ d)} = 50$	$C1_{(0,\ e,\ c)} = 18$
$C1_{(d,\ a,\ f)} = 41$	$C1_{(0,\ e,\ f)} = 31$
$C1_{(0,\ b,\ c)} = 10$ min(b)	**$C1_{(c,\ e,\ d)} = 10$** min(e)
$C1_{(0,\ b,\ f)} = 47$	$C1_{(d,\ e,\ f)} = 28$

C $C2_{(i,\ k,\ j)} = \lambda D0k - C1_{(i,\ k,\ j)}$ $(\lambda=2)$
(selecionando o vértice para inserção)

$C2_{(0,\ a,\ f)} = \lambda D0a - C1_{(0,\ a,\ f)}$
$C2_{(0,\ a,\ f)} = 2*18 - 22 = 14$

$C2_{(0,\ b,\ c)} = \lambda D0b - C1_{(0,\ b,\ c)}$
$C2_{(0,\ b,\ c)} = 2*26 - 10 = 42$

$C2_{(c,\ e,\ d)} = \lambda D0e - C1_{(c,\ e,\ d)}$
$C2_{(c,\ e,\ d)} = 2*29 - 10 = 48$ max(C2)

inserir o vértice ***e*** entre os vértices ***c*** e ***d***

D

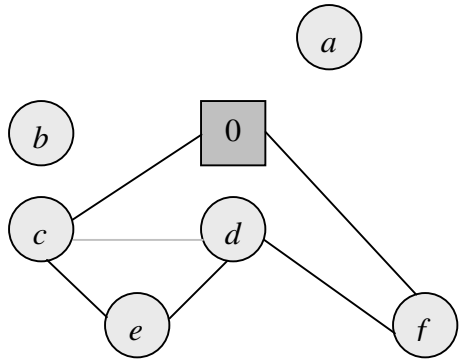


Figura 4.8 – Movimento de Expansão da Heurística de Mole e Jameson

Para os cálculos realizados na figura 4.8, foram utilizados valores que representam as distâncias, em milímetros, entre os centros dos vértices do grafo. São eles: $D0a$=18, $D0b$=26, $D0c$=29, $D0d$=13, $D0e$=29, $D0f$=37, Dab=41, Dac=47, Dad=29, Dae=47, Daf=41, Dbc=13, Dbd=29, Dbe=29, Dbf=58, Dcd=26, Dce=18, Dcf=54, Dde=18, Ddf=29 e Def=39.

Observando a célula B, notamos que o vértice ***a***, por exemplo, pode ser inserido entre os pares adjacentes 0*c*, 0*f*, *cd* e *df*. Entretanto, a melhor posição de inserção é obtida com o par de vértices que minimiza o custo da primeira condição, e para o vértice em questão, a melhor posição é entre os vértices 0 e *f*. Já na célula C, escolhemos qual vértice será inserido na rota, tomando como base o valor máximo obtido através da segunda condição. Um detalhe que merece destaque, é que no exemplo utilizado, o vértice ***e*** foi escolhido para ser inserido entre o vértice ***c*** e o vértice interno ***d***. Esse é um dos pontos positivos dessa heurística em relação a heurística de Clarke e Wright, onde a rota é expandida apenas a partir de um de seus vértices extremos.

Como para as estratégias anteriores, implementamos os passos dessa heurística no programa responsável pelas simulações, e a executamos no mesmo cenário dos exemplos anteriores. Selecionamos também os procedimentos de melhoria. O resultado da simulação está ilustrado na figura a seguir.

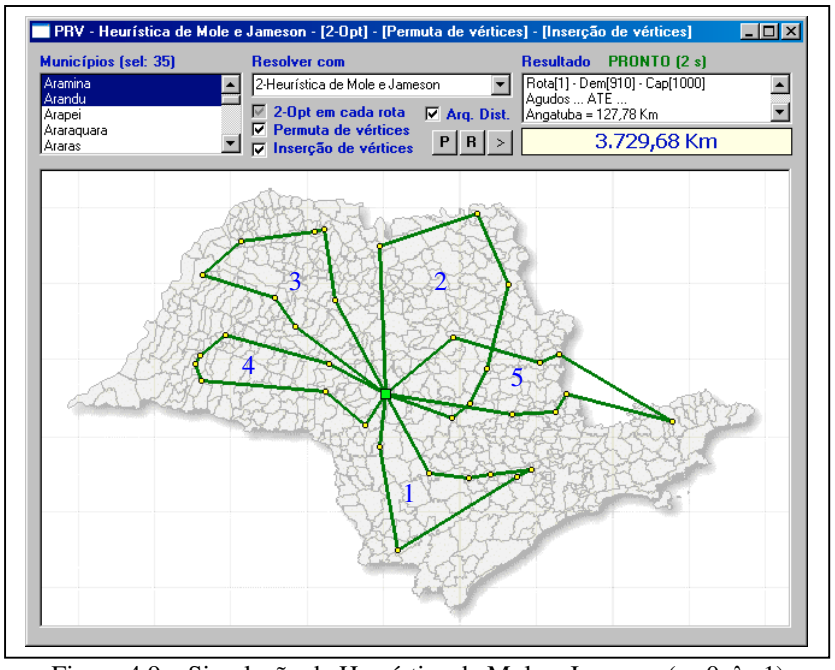


Figura 4.9 – Simulação da Heurística de Mole e Jameson (μ=0, λ=1)
(35 primeiros municípios de SP, em ordem alfabética, iniciados pela letra A. Depósito=Agudos)

Observando a figura 4.9, notamos que o resultado apresentado é melhor que o resultado dos procedimentos anteriores. Maiores detalhes das rotas podem ser observados no Apêndice B.

As constantes μ e λ tornam o método flexível, ou seja, alterando os valores dessas constantes podemos gerar rotas com diferentes formatos. Quando aumentamos o valor de λ, valorizamos a distância entre o depósito e o vértice a ser inserido. Já quando aumentamos o valor de μ, penalizamos as ligações mais longas (Goldbarg e Luna [3]).

4.1.4 Heurística de Gillett e Miller

A Heurística de Gillett e Miller [24] faz parte do grupo das heurísticas de duas fases, que utilizam a estratégia de dividir o problema em partes para resolvê-lo. Na primeira fase dessa heurística, os vértices são organizados em grupos segundo um critério de proximidade. Na segunda fase cada grupo é solucionado de forma independente, onde os grupos correspondem às rotas a serem construídas.

Para agrupar os vértices, é realizada uma varredura circular a partir do depósito central, na qual os vértices vão sendo escolhidos de acordo com o ângulo de sua coordenada polar em relação ao depósito, respeitando as restrições do problema. Em seguida, as rotas são obtidas através da solução do PCV para cada grupo de vértices. Os passos do algoritmo podem ser descritos da seguinte maneira:

Heurística de Gillett e Miller

1. Tome o depósito como origem do sistema de coordenadas polares; para cada vértice calcule suas coordenadas polares em relação ao depósito e guarde-as numa lista;
2. Ordene a lista contendo as coordenadas polares dos vértices em ordem crescente de ângulo;
3. Percorra a lista iniciando pelo topo e siga agrupando os vértices até que alguma restrição, de capacidade por exemplo, seja violada;
4. Armazene o grupo formado em um vetor e inicie um novo grupamento a partir do vértice que violou a restrição; siga com a varredura até o final da lista;
5. Quando a lista for esgotada, obtenha as rotas resolvendo de maneira exata ou aproximada o Problema do Caixeiro Viajante para cada grupo encontrado.

A figura 4.10 ilustra as duas fases dessa estratégia. Na célula A, os grupos de vértices são organizados a partir da varredura circular. Na célula B, as rotas são construídas a partir da solução do PCV para cada um dos grupos escolhidos na primeira fase. Essa estratégia já havia sido utilizada anteriormente por Wren e Holliday [25], mas ela é normalmente atribuída a Gillett e Miller, que a popularizaram.

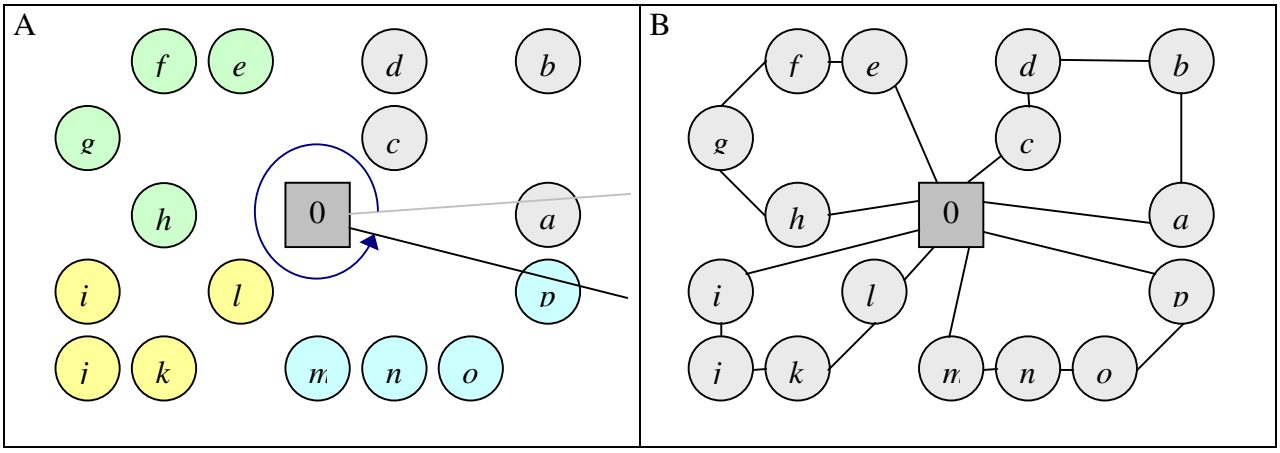


Figura 4.10 – As duas fases da Heurística de Gillett e Miller

A seguir, temos o resultado da simulação realizada com o mesmo cenário utilizado para as simulações anteriores. Na tela de parâmetros, selecionamos para que o algoritmo procurasse pela melhor varredura, conseguindo com isso um resultado melhor que os anteriormente apresentados.

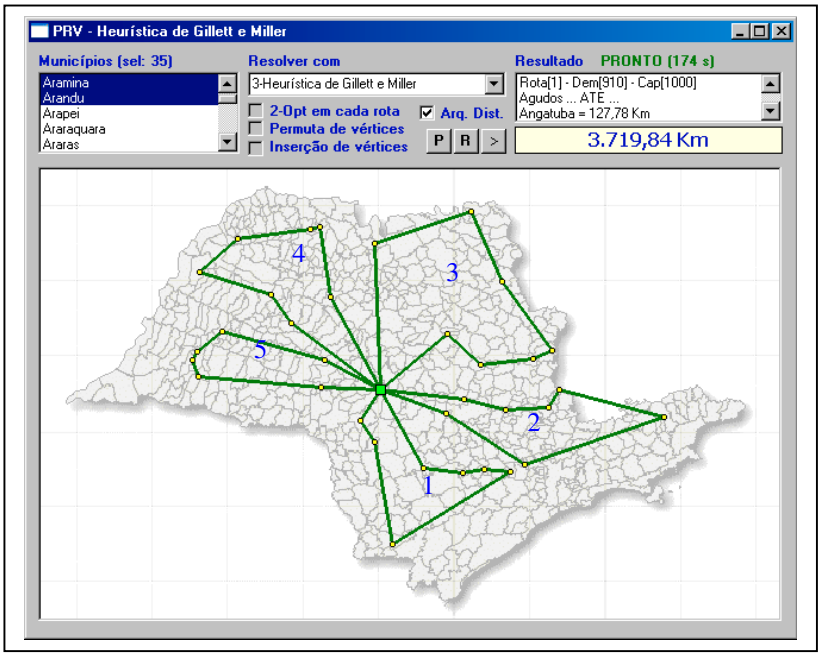


Figura 4.11 – Simulação da Heurística de Gillett e Miller (*Sweep*)
(35 primeiros municípios de SP, em ordem alfabética, iniciados pela letra A. Depósito=Agudos)

Na nossa implementação, resolvemos o PCV para cada grupo de maneira exata. No Apêndice B podemos ver maiores detalhes das rotas da figura 4.11. Quando executamos os passos do algoritmo, obtemos uma solução que pode não ser a melhor dentre as possíveis que o método pode fornecer. Podemos então seguir em busca de uma melhor varredura, deslocando o primeiro vértice para a última posição da lista com as coordenadas polares já ordenadas, e executando o algoritmo novamente. Armazenamos a melhor solução e repetimos o procedimento até que todos os vértices tenham ocupado a posição de início do processo. Podemos ainda repetir todo esse procedimento, só que realizando a varredura no sentido oposto ao utilizado anteriormente. Naturalmente tal esforço consome um tempo maior de processamento.

4.2 Metaheurística Busca Tabu

As metaheurísticas são procedimentos que realizam uma exploração mais aprofundada no espaço das soluções, em busca de melhores resultados. Enquanto as heurísticas clássicas encerram sua busca ao encontrar um ótimo local, as metaheurísticas possuem mecanismos para transpor os ótimos locais e continuar a exploração em busca de melhores soluções.

Gendreau, Laporte e Potvin [26] relatam que nos últimos anos muitas metaheurísticas têm sido propostas para o PRV. Dentre elas, os autores identificam seis tipos principais: *Simulated Annealing* e *Deterministic Annealing* (baseadas na simulação do resfriamento e solidificação da matéria), *Ant Systems* (baseada na simulação do comportamento das formigas em busca de alimento), Busca Tabu, Algoritmos Genéticos, e Redes Neurais. Dentre esses tipos, os autores apontam a Busca Tabu como o método mais eficiente.

Segundo Gendreau, Hertz e Laporte [27], a Busca Tabu foi proposta por Glover em 1977, e dois dos primeiros trabalhos associadas ao PRV são de autoria de Willard em 1989, e de Pureza e França em 1991. Depois disso, uma série de outros autores publicaram os esforços de suas pesquisas, conseguindo resultados expressivos diante de outros métodos para o PRV.

Para este item do capítulo, apresentamos alguns conceitos fundamentais da Busca Tabu. Para isso, estudamos as estratégias apresentadas no trabalho anterior da referida autora, Pureza [28], e implementamos uma versão simplificada de uma delas no aplicativo que realiza as simulações.

Inicialmente, vamos imaginar um cenário onde aplicamos uma heurística rápida qualquer, visando encontrar uma solução de partida para uma instância do PRV, por exemplo. Em seguida,

podemos aplicar alguns procedimentos de melhoria, como apresentado neste capítulo, em busca de um resultado de maior qualidade. Tais procedimentos, conseguem melhorar a solução até encontrar um mínimo local.

Se quisermos continuar com o processo de busca, a primeira coisa a fazer é realizar um movimento de piora, digamos o que causa menor aumento no custo, de maneira que comecemos a escapar desse mínimo local. A partir dessa situação, podemos pesquisar dentre os movimentos disponíveis, um que realiza uma melhora na solução. Existe a possibilidade de que tal movimento faça com que a solução retorne ao mínimo local anteriormente alcançado. Caso isso ocorra, o passo seguinte indicaria novamente o movimento que causa menor aumento no custo, e na seqüência poderíamos novamente retornar ao mínimo local. Essa situação ilustra o fenômeno da ciclagem, capaz de comprometer o processo em busca de melhores soluções. Precisamos, portanto, de um elemento restritivo na busca, capaz de impedir a reversão dos movimentos, fazendo com que o processo siga procurando novas soluções por regiões ainda não pesquisadas.

Poderíamos armazenar todos os movimentos anteriormente efetivados para realizar comparações, mas isso implicaria em maior utilização de memória e esforço de processamento. Além do mais, após sucessivas alterações, a possibilidade de retorno a uma determinada solução diminui. Isso sugere que a restrição feita a certos movimentos pode ser relaxada ao longo do processo. Caso não ocorra mais o risco de ciclagem, tal relaxação é até desejável, pois os movimentos liberados oferecem maior flexibilidade ao método.

Os aspectos anteriormente referidos constituem os dois elementos básicos da Busca Tabu: a restrição da busca, pela classificação dos movimentos tabu (movimentos proibidos), e a liberação dos movimentos através de uma função de curto prazo, que provoca um esquecimento conveniente (Pureza [28]).

Podemos exemplificar a aplicação desses elementos, através da rotina desenvolvida para o programa responsável pelas simulações. Essa rotina, utiliza os movimentos de troca e inserção de vértices ilustrados nas figuras 4.5 e 4.6 desse capítulo. Lembrando que para realizar a troca de um vértice de uma rota com um vértice de outra rota, é necessário remover quatro arestas e incluir quatro outras arestas. Já para o movimento de inserção, ou seja, para uma rota doar um vértice para outra rota, é necessário remover três arestas e incluir outras três.

Nos procedimentos de melhoria, o movimento de troca ou de inserção só era efetivado se a soma dos custos das arestas removidas fosse maior que a soma dos custos das arestas incluídas.

Em outras palavras, o movimento só era efetivado se ele melhorasse a solução. Para a Busca Tabu, quando não é possível melhorar a solução, procuramos um movimento que causa o menor aumento no custo. Dessa forma, podemos escapar dos ótimos locais.

Para implementar os elementos básicos da Busca Tabu na rotina, utilizamos estruturas de dados simples, que armazenam os identificadores das arestas envolvidas nos movimentos de troca e inserção de vértices entre rotas. Como sugerido por Pureza [28], utilizamos duas listas: uma para armazenar os índices das arestas removidas por movimentos anteriores (REM), e outra para armazenar os índices das arestas incluídas por movimentos anteriores (INC). Essas listas são chamadas de **listas tabu**, ou listas de proibições.

O tamanho dessas listas está relacionado com o número de movimentos que desejamos guardar para verificações. Por exemplo, para uma lista de tamanho $T = 5$ movimentos, utilizamos um vetor capaz de armazenar até vinte arestas (5 movimentos * 4 arestas). Lembrando que o movimento de troca simples de vértices entre rotas, trabalha com quatro arestas para cada lista, uma aresta a mais que o movimento de inserção de vértice.

Durante o processo, os índices das arestas envolvidas em cada movimento efetivado são incluídos no início das listas correspondentes, deslocando os demais índices para o final da lista. Para ilustrar essa manobra, juntamente com o exemplo do parágrafo anterior, partimos das listas mostradas na figura 4.12, que já possuem índices de arestas armazenados.

REM

1	2	3	4	11	12	13	14	21	22	23	24	31	32	33	34			

INC

6	7	8	9	16	17	18	19	26	27	28	29	36	37	38	39			

Figura 4.12 – Exemplo das listas tabu

Vamos supor que no movimento de intercâmbio seguinte, as arestas 51, 63, 72, 84 tenham sido removidas da solução, e as arestas 89, 78, 67, 56 tenham sido incluídas, por um movimento de troca de vértices. Após o movimento, as listas tabu ficam preenchidas da seguinte forma:

REM

51	63	72	84	1	2	3	4	11	12	13	14	21	22	23	24	31	32	33	34

INC

89	78	67	56	6	7	8	9	16	17	18	19	26	27	28	29	36	37	38	39

Figura 4.13 – Inclusão de arestas na lista tabu após movimento de troca simples de vértices

Note que os índices das arestas do último movimento foram incluídos no início da lista, deslocando os demais para a direita, no sentido do final da lista. Vamos supor que no movimento seguinte, as arestas 91, 92, 93 tenham sido removidas da solução e que as arestas 99, 98, 97 tenham sido incluídas, por um movimento de inserção de vértice. Após o movimento, as listas de proibições ficam preenchidas da seguinte maneira:

REM

91	92	93	51	63	72	84	1	2	3	4	11	12	13	14	21	22	23	24	31

INC

99	98	97	89	78	67	56	6	7	8	9	16	17	18	19	26	27	28	29	36

Figura 4.14 – Inclusão de arestas na lista tabu após movimento de inserção de vértice

Novamente os índices do movimento mais recente foram incluídos no início da lista, empurrando os demais. Note que após essa manobra, os índices 32, 33, 34, 37, 38 e 39 deixaram de fazer parte das listas tabu. Isso significa que a rotina pode esquecer de verificar essas arestas, ou ainda, que as arestas com esses índices podem novamente ser utilizadas nos movimentos de intercâmbio de vértices entre rotas.

O conteúdo das listas de proibições é verificado antes da efetivação de cada movimento. Em outras palavras, a cada iteração do algoritmo, todos os possíveis movimentos de troca e de inserção de vértices são analisados. Para cada movimento, verificamos se alguma restrição do problema foi violada, e se as arestas que participam do movimento estão contidas nas listas de proibições. Se arestas forem encontradas nas listas, satisfazendo algum critério preestabelecido, então o movimento é classificado como **movimento tabu,** ou movimento proibido. Caso isso ocorra, ele não é efetivado, e o algoritmo passa a analisar outro candidato.

Podemos construir diferentes critérios para classificar um movimento como tabu. Alguns podem ser rígidos demais, fazendo com que a rotina deixe de explorar regiões promissoras do espaço das soluções. Outros podem ser flexíveis demais, fazendo com o que o risco do aparecimento do fenômeno da ciclagem aumente.

Com base no trabalho de Pureza [28], fizemos alguns experimentos e implementamos o seguinte critério: um movimento é classificado como tabu, se todas as arestas removidas por ele estiverem contidas na lista tabu das arestas anteriormente incluídas (INC), ou se todas as arestas incluídas por ele estiverem contidas na lista tabu das arestas anteriormente removidas (REM). Com o auxílio da próxima figura, apresentamos exemplos de movimentos proibidos.

A partir da figura 4.15, vamos supor que o movimento que o algoritmo está analisando esteja propondo a remoção das arestas 6, 7, 8 e a inclusão das arestas 1, 2, 3 na solução.

REM

1	2	3	11	12	13	14	21	22	23	31	32	33	41	42	43	44	51	52	53

INC

6	7	8	16	17	18	19	26	27	28	36	37	38	46	47	48	49	56	57	58

Figura 4.15 – Listas tabu para exemplos de movimentos tabu

O critério implementado procura as arestas de índice 6, 7, 8 que o movimento propõe remover na lista das anteriormente incluídas (INC). Ele também procura as arestas 1, 2, 3 que o movimento pretende incluir na lista de proibições (REM). Como as arestas 6, 7, 8 foram encontradas em (INC) e as arestas 1, 2, 3 foram encontradas em (REM) o movimento é então classificado como proibido, ou movimento tabu. Note que tal movimento corresponde à reversão do movimento anterior.

Vamos supor que um outro movimento, em análise, esteja propondo a remoção das arestas 61, 71, 81, 91 e a inclusão das arestas 21, 31, 41, 51. A rotina também classifica esse movimento como tabu porque todas as arestas, que ele pretende incluir, estão contidas na lista tabu das arestas removidas (REM).

Em outro exemplo, vamos supor que um determinado movimento esteja propondo a remoção das arestas 61, 71, 81, 91 e a inclusão das arestas 41, 42, 43, 99. Esse movimento não será classificado como tabu, porque nenhuma das arestas que ele pretende remover foi encontrada na lista (INC) e pelo menos uma das arestas que ele pretende incluir não foi encontrada na lista (REM). Logo ele pode ser efetivado.

Na nossa implementação, a cada movimento efetivado, acrescentamos um registro em um arquivo, contendo as seguintes informações: iteração ou número do movimento efetivado, tamanho das listas tabu, tipo do movimento (troca ou inserção de vértices), valor atual da solução após o movimento e valor da melhor solução encontrada até o momento. Fazemos isso para facilitar a geração de relatórios e gráficos do comportamento da busca.

Implementamos o atributo *T* (tamanho das listas tabu), como parâmetro de entrada, que pode receber valores entre 3 e 11 (movimentos), armazenando de 12 a 44 arestas para verificação. Quanto menor o tamanho da lista, maior é o risco da ciclagem, entretanto, listas muito restritivas podem influenciar de maneira negativa na busca por soluções de boa qualidade. Pureza [28]

menciona que aparentemente, condições pouco restritivas (risco maior de clicagem) fornecem soluções de melhor qualidade do que condições severas.

Durante o processo, movimentos de intercâmbio de vértices vão sendo realizados até uma condição de parada, que pode ser relacionada com tempo de processamento, ou com número de iterações do algoritmo. Para a nossa implementação, utilizamos o número de tentativas de melhoria da solução, através dos movimentos de intercâmbio de vértices, que desejamos que a rotina execute. Os passos do algoritmo implementado podem ser descritos da seguinte forma:

Metaheurística Busca Tabu

16. Gere uma solução inicial através de uma heurística rápida qualquer.
17. Repita
18. ... Para todas as possíveis trocas, de um vértice, entre duas rotas diferentes, faça:
19. ..... Se com a troca, nenhuma restrição do PRV for violada, faça:
20. ..... Se o movimento de troca não for um movimento tabu, faça:
21. ..... Verifique se é o melhor movimento de troca proposto nessa iteração; caso seja, ... armazene o custo obtido e as arestas que participam do movimento.
22. ... Para todas as possíveis inserções de um vértice de uma rota para outra, faça:
23. ..... Se com a inserção ou doação, nenhuma restrição do PRV for violada, faça:
24. ..... Se o movimento de inserção não for um movimento tabu, faça:
25. ..... Verifique se é o melhor movimento de inserção proposto nessa iteração; caso ..... seja, armazene o custo obtido e as arestas que participam do movimento.
26. ... Dos dois movimentos armazenados pelas duas estratégias, selecione o candidato que ... forneça o maior ganho/lucro ou o menor prejuízo para a solução.
27. ... Efetive o movimento; se esse movimento gerar uma solução melhor, do que a ... melhor solução encontrada até então, armazene os dados da nova solução.
28. ... Atualize as listas tabu com as arestas que participaram do movimento efetivado; ... armazene os dados da iteração no arquivo.
29. Até que número de iterações seja igual ao valor *X*, passado como parâmetro.
30. Aplique a heurística 2-Opt em cada rota.

A seguir, apresentamos o resultado da aplicação dessa rotina no cenário proposto para as estratégias desse capítulo. Para isso, executamos o algoritmo algumas vezes, utilizando diferentes soluções de partida. Curiosamente, a melhor solução foi obtida a partir de uma solução de partida ruim, mostrada na figura 4.16.

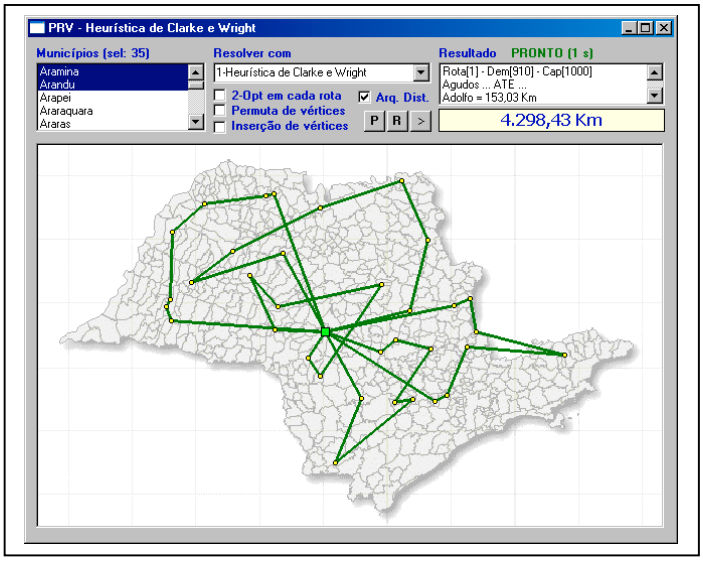


Figura 4.16 – Solução de Partida para Busca Tabu (Clarke e Wright com θ de Gaskell = 0,5)
(35 primeiros municípios de SP, em ordem alfabética, iniciados pela letra A. Depósito=Agudos)

Após 800 tentativas de melhoria, executadas em 368 segundos, chegamos a uma solução melhor do que todas as apresentadas anteriormente neste capítulo (ver figura 4.17).

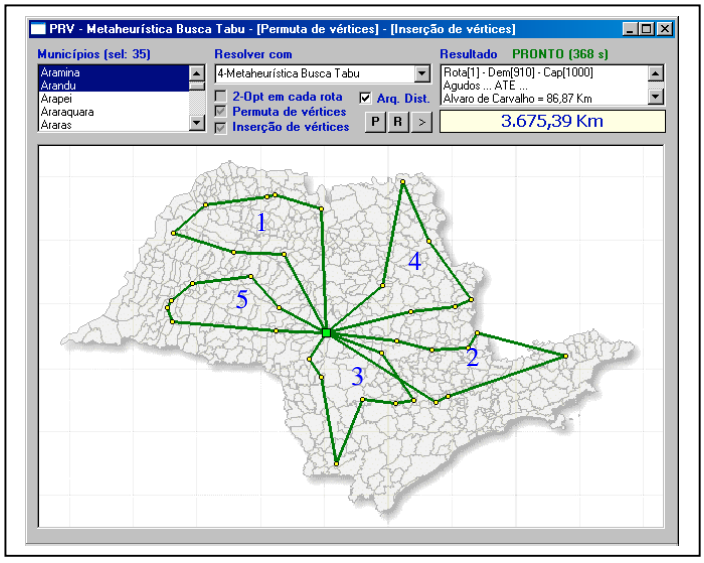


Figura 4.17 – Metaheurística Busca Tabu
(35 primeiros municípios de SP, em ordem alfabética, iniciados pela letra A. Depósito=Agudos)

A composição das rotas pode ser vista no Apêndice B. Na figura seguinte, temos um gráfico que ilustra o perfil da busca. Ele é composto por duas linhas, uma representando o valor da solução encontrada naquele instante, e outra representando o valor da melhor solução encontrada no decorrer do processo.

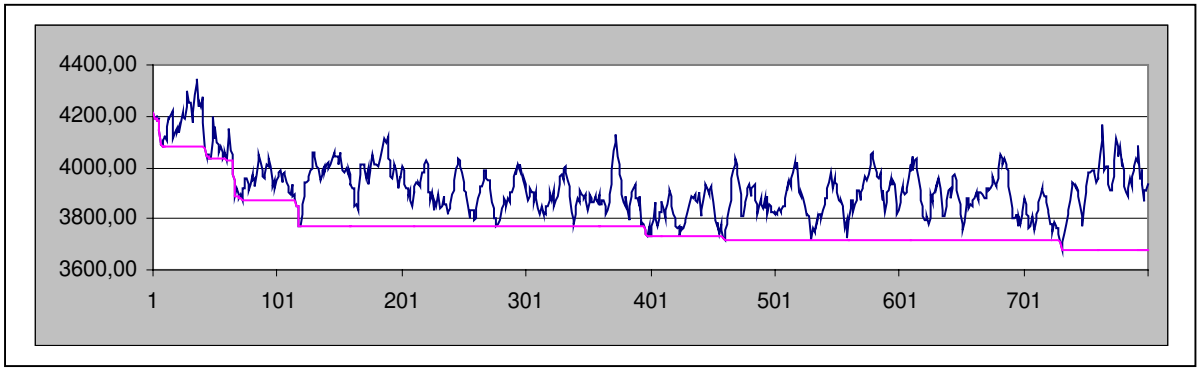


Figura 4.18 – Gráfico Custo x Iteração (Busca Tabu)

Essa solução foi obtida com $T = 7$ movimentos, ou seja, com listas tabu capazes de armazenar 28 arestas cada. A partir do momento que diminuímos o valor de T, aumentamos o risco da ciclagem, que pode ser ilustrada a partir da figura seguinte ($T = 3$ movimentos).

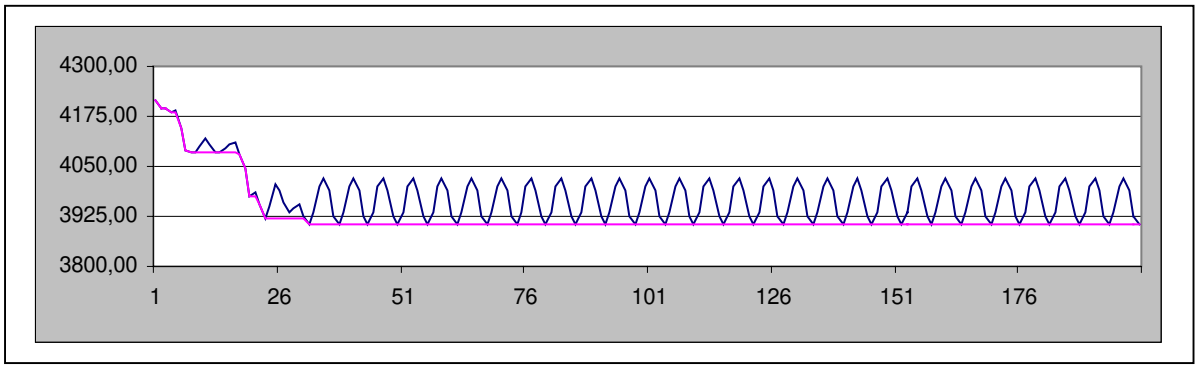


Figura 4.19 – Gráfico Exemplo de Ciclagem

Observe que o processo consegue melhorar a solução até um determinado ponto, onde ocorre a ciclagem. A partir daí, o algoritmo se torna estéril, ou seja, passa a insistir em movimentos repetitivos que não fornecem melhoria para a solução final.

Além dos elementos apresentados, existem outros conceitos relacionados com a Busca Tabu que não foram utilizados na nossa implementação. Pureza [28] dedica um capítulo do seu trabalho para apresentação dos fundamentos dessa estratégia. Dentre os conceitos por ela demonstrados, citamos os seguintes:

- **Critérios de Aspiração** – São elementos relacionados com a qualidade da solução. A partir deles, podemos aceitar uma solução de alta qualidade que utiliza elementos restritivos, como por exemplo, arestas incluídas nas listas tabu. O mais simples dos critérios de aspiração consiste em aceitar todo movimento, proibido ou não, que gere uma solução melhor do que todas as obtidas até aquele momento.
- **Intensificação Regional** – É uma estratégia promovida por funções de memória de médio prazo que comparam e armazenam características comuns de um certo número de boas soluções, coletadas durante um período específico da busca. Tais características são tomadas como um atributo regional de boas soluções. De posse dessas informações, a estratégia procura outras soluções que apresentem tais características, impondo restrições ou penalizações aos movimentos disponíveis.
- **Diversificação Global** – É uma estratégia promovida por funções de memória de longo prazo que estimulam a procura de soluções em regiões ainda não exploradas do espaço de busca. Um exemplo simples apontado pela autora, seria contar o número de vezes que cada aresta aparece nas rotas geradas. Uma penalização proporcional pode ser aplicada a esse número, fazendo com que as arestas que mais participaram dos movimentos de intercâmbio, não sejam tão utilizadas como foram anteriormente. Com isso, novas regiões passam a ser exploradas.

Além de não implementar estes três últimos conceitos, também não nos preocupamos com questões relacionadas ao tempo de processamento, visto que o algoritmo tenta realizar todas as possíveis trocas, mesmo entre vértices distantes, e todas as possíveis doações de vértices, mesmo entre rotas aparentemente distantes. De qualquer forma, mesmo com uma implementação simples da Busca Tabu, conseguimos um resultado melhor que os resultados das outras estratégias.

No Apêndice D, apresentamos testes modificando um pouco o cenário proposto neste capítulo. Alteramos as demandas dos municípios, gerando três novos cenários que foram testados com todas as estratégias apresentadas neste trabalho para o Problema de Roteamento de Veículos.

5 Observando Aspectos Geométricos

Para este trabalho, implementamos um aplicativo que simula a execução de alguns algoritmos e consequentemente auxilia no entendimento dos mesmos. O motivo principal dessa implementação, era o de enriquecer o texto com ilustrações dos resultados das estratégias estudadas, mas com o tempo, passamos a utilizar o aplicativo para observar o comportamento dos algoritmos com um olhar um pouco mais aguçado. Com o módulo desenvolvido para o PCV, selecionamos diversos grupos de municípios e executamos as simulações. É válido mencionar que a interface fornece as soluções em duas dimensões, e que as características observadas são válidas para problemas Euclidianos.

Um dos primeiros aspectos que podemos notar é que, se todos os municípios pertencerem ao casco convexo, o circuito correspondente é ótimo. Em outras palavras, os polígonos convexos são rotas naturalmente ótimas. Bellmore e Nemhauser [8], fazem menção desse fato no terceiro teorema do estudo que publicaram sobre o PCV.

Com essa observação, podemos imaginar uma heurística para o PCV Euclidiano. Inicialmente, com os municípios a serem percorridos, encontramos o casco convexo que abriga todos os demais vértices em seu interior. Em seguida, vamos incluindo os municípios entre cada par adjacente de vértices do circuito, escolhendo o candidato que minimizar a expressão {min_custo = custo das arestas que entram – custo da aresta que sai}. Na seqüência de figuras seguinte podemos visualizar um exemplo dessa estratégia.

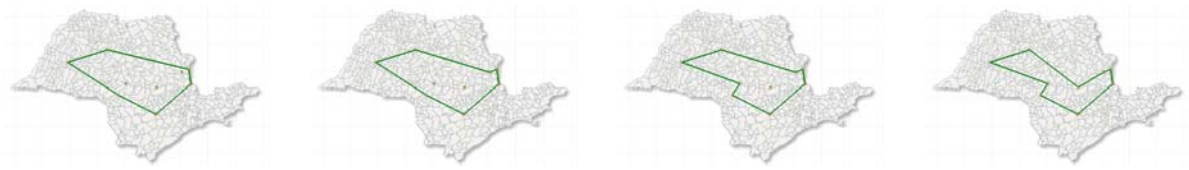


Figura 5.1 – Heurística do Casco Convexo + Inserção mais barata

Na primeira vez que executamos a simulação dessa pequena instância (nove municípios), imaginávamos que a heurística nos devolveria o resultado ótimo, o que não ocorreu.

Partimos então para uma outra tentativa. Novamente, começamos com a construção do casco convexo, só que ao invés de utilizar um critério baseado em custos, procuramos por um vértice *k* ainda não incluído no circuito, e por um par de vértices adjacentes $\{i, j\}$ pertencentes ao circuito, que maximize o ângulo formado entre as arestas $\{i, k\}$ e $\{k, j\}$. Em outras palavras, procuramos inserir um vértice com o qual o ângulo entre as arestas que serão incluídas seja o maior dentre os candidatos. Na seqüência de figuras seguinte podemos visualizar o resultado da simulação para essa estratégia.

Figura 5.2 – Heurística do Casco Convexo + Inserção do maior ângulo

A solução apresentada por essa estratégia é a mesma obtida através do método de planos de corte (solução ótima). Isso ocorreu devido, em parte, à simplicidade do problema. Simulamos para instâncias de vários tamanhos, e notamos que para problemas com número maior de municípios, essas estratégias raramente encontram o valor ótimo. Notamos ainda, que em alguns casos, a que utiliza o critério baseado em custos apresenta valores melhores do que a que utiliza o critério baseado em ângulos. Em outros casos ocorre o inverso.

Na tentativa de aproveitar o melhor das duas heurísticas, combinamos os critérios de escolha do vértice em uma única heurística, cujos passos podem ser descritos da seguinte forma:

Heurística do Casco Convexo Mista

1. Com o conjunto de vértices construa o casco convexo e tome-o como circuito inicial.
2. Varie *varAngAux* de 0º até 180º saltando em intervalos de 3º
3. ... Enquanto existirem vértices ainda não roteados
4. ..... Para cada vértice *k* não incluído no circuito, encontre o par de vértices adjacentes ..... $\{i, j\}$ do circuito que minimize a expressão $C_{ik} + C_{kj} - C_{ij}$; armazene os dados do ..... melhor candidato à inserção com esse critério;
5. ..... Para cada vértice *k* não incluído no circuito, encontre o par de vértices adjacentes ..... $\{i, j\}$ do circuito que maximize o ângulo entre as arestas $_{ik}$ e $_{kj}$; armazene os dados ..... (ângulo e vizinhos) do melhor candidato à inserção com esse critério;

Heurística do Casco Convexo Mista (continuação)

7. ..... Se ângulo encontrado em 5 for maior que *varAngAux* insira o vértice encontrado ..... no passo 5, senão insira o vértice obtido através do passo 4;
8. ... Quando todos os vértices estiverem sido incluídos no circuito, verifique se a solução ... apresentada é melhor que a anteriormente obtida; armazene a melhor solução;
9. Próximo *varAngAux*;
10. Apresente a melhor solução obtida.

Realizamos algumas simulações com as três heurísticas e com o método de planos de corte. De início, esperávamos conseguir com a heurística mista o melhor resultado das heurísticas nela envolvidas, ou seja, ou o resultado obtido utilizando o critério baseado em custos, ou o resultado obtido utilizando o critério baseado em ângulos. Entretanto, em algumas instâncias observamos um valor melhor, do que o melhor valor obtido por uma das heurísticas mencionadas individualmente. Na tabela seguinte podemos observar alguns desses resultados:

Instância	Casco Convexo Custos	Casco Convexo Ângulos	Casco Convexo Mista	Método de Planos De Corte
(SP-18-L) municípios de SP iniciados pela letra L	1587,78 Km	1587,78 Km	1587,78 Km	1587,78 Km
(SP-22-J) municípios de SP iniciados pela letra J	2022,91 Km	2036,08 Km	2022,91 Km	2016,14 Km
(SP-20-A) 20 primeiros mun. de SP iniciados pela letra A	1769,16 Km	1705,82 Km	1705,82 Km	1703,75 Km
(SP-20-N) municípios de SP iniciados pela letra N	2043,91 Km	2182,52 Km	1990,83 Km	1977,59 Km
(SP-45-M) municípios de SP iniciados pela letra M	2547,76 Km	2518,36 Km	2515,68 Km	2502,44 Km

Na primeira linha da tabela, temos um comportamento raro, onde as três heurísticas conseguem obter o valor ótimo. Na segunda e terceira linhas, temos o resultado que esperávamos quando implementamos a heurística mista. Nelas, o valor da heurística mista é igual ao valor do melhor resultado obtido por uma das outras duas heurísticas. Já nas duas últimas linhas, observamos um resultado interessante, onde os critérios combinados geraram um valor melhor do que o melhor valor obtido por um deles.

Intrigados com esses resultados, decidimos verificar com maior intensidade se tal combinação de elementos, (casco convexo, inserção mais barata e inserção a partir do maior

ângulo), já havia sido explorada anteriormente. Depois de algumas pesquisas, encontramos em Lawler *et al*. [29] p. 215, algumas heurísticas para o PCV Euclidiano que utilizam esses mecanismos. Por exemplo, encontramos uma heurística de inserção, atribuída a Stewart em 1977, que utiliza o casco convexo como circuito de partida. A idéia de utilizar ângulos já foi utilizada por Norback e Love em 1977 e 1979. Encontramos também uma heurística que combina esses elementos, denominada CCAO [*Convex hull* (casco convexo), *Cheapest insertion* (inserção mais barata), *Angle selection* (escolha por ângulo) e *Or-opt* (combinação de 2-Opt e 3-Opt)]. Essa heurística fornece bons resultados, por isso achamos interessante descrever os seus passos:

Heurística CCAO

1. Forme um casco convexo com o conjunto de cidades e tome-o como circuito inicial;
2. Para cada cidade k não incluída no circuito, identifique o par adjacente de cidades $_{ik}$ e $_{jk}$ do circuito que minimize a expressão $C_{ikk} + C_{kjk} - C_{ikjk}$
3. Selecione a cidade k^* que maximize o ângulo entre as arestas $\{_{ik}, k\}$ e $\{k, _{jk}\}$, e insira essa cidade entre ik^* e jk^*;
4. Repita os passos 2 e 3 até a obtenção do circuito hamiltoniano;
5. Aplique a heurística Or-Opt (pós otimização).

Embora combinações de critérios utilizando custos e ângulos já tenham sido apresentadas no passado, a realização desses experimentos estimulou a nossa curiosidade quanto ao fato de tais combinações produzirem valores como os relatados nas duas últimas linhas da tabela anterior.

Quando observamos as duas seqüências de figuras, percebemos que para aquela instância, o critério baseado em ângulos se mostrou superior pois a ordem em que os vértices foram sendo inseridos favorecia a inclusão do vértice seguinte em uma vizinhança mais apropriada. Notamos que em algumas instâncias a combinação dos dois critérios favorece ainda mais a organização dos vértices em locais mais apropriados.

Diante desses resultados, decidimos investigar um pouco mais sobre a combinação desses elementos. Resolvemos experimentar combinações de custos e ângulos nos movimentos de intercâmbio de vértices da metaheurística Busca Tabu. Tomamos como ponto de partida a estratégia exposta no capítulo anterior, que foi implementada com base no trabalho de Pureza [28], e elaboramos alguns critérios para realizarmos os ensaios.

5.1 Critérios para Movimentos de Intercâmbio de Vértices

Os critérios para realização dos experimentos deste capítulo são baseados nos movimentos de troca e inserção de vértices ilustrados no capítulo anterior. No capítulo 4, tais movimentos eram executados com a finalidade de diminuir as distâncias a serem percorridas nas rotas. Neste capítulo, variações desses movimentos serão propostas com a finalidade de aumentar os ângulos formados entre os vértices trocados e suas respectivas vizinhanças. Veja os exemplos a seguir:

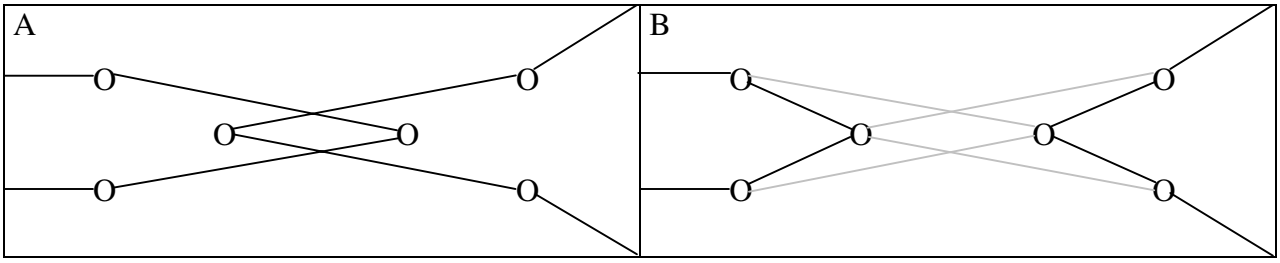


Figura 5.3 – Exemplo de movimento de troca de vértices baseado em distâncias

Na figura 5.3 temos um exemplo de movimento de troca simples de vértices entre duas rotas. As arestas em tons mais claros, mostradas no quadro B, são as que foram removidas pelo movimento de troca. A decisão para efetivação do movimento é tomada em função da distância economizada, lembrando que para a Busca Tabu, quando não encontramos um movimento que melhore a solução, elegemos o que causa menor aumento no custo.

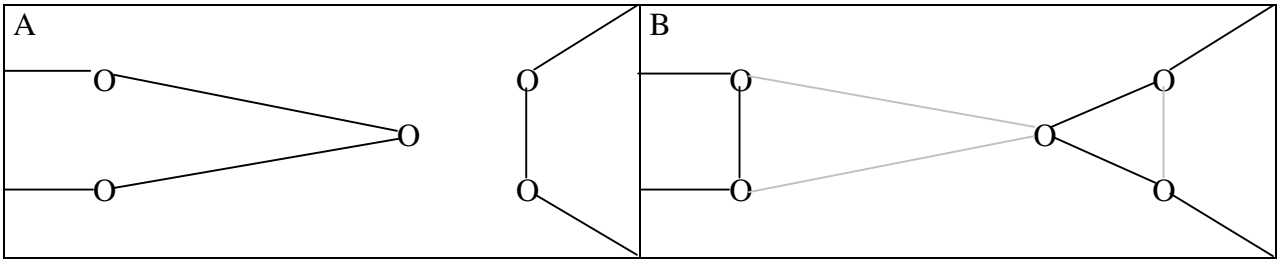


Figura 5.4 – Exemplo de movimento de inserção de vértice baseado em distâncias

Na figura 5.4 temos um exemplo de movimento de inserção de vértice. Note que a rota posicionada à esquerda doa um de seus vértices para a rota posicionada à direita da figura. Novamente, as linhas em tons claros são as que foram removidas durante o movimento. Aqui também, a decisão para efetivação do movimento é tomada em função da economia de distância. No capítulo 4, esses dois tipos de movimentos eram executados para todas as possibilidades de troca e doação de vértices entre diferentes rotas. Durante o processo, o melhor resultado de cada

tipo ia sendo armazenado. Em seguida o movimento efetivado era o que trazia maior lucro ou menor prejuízo para a solução, em termos de distância a percorrer.

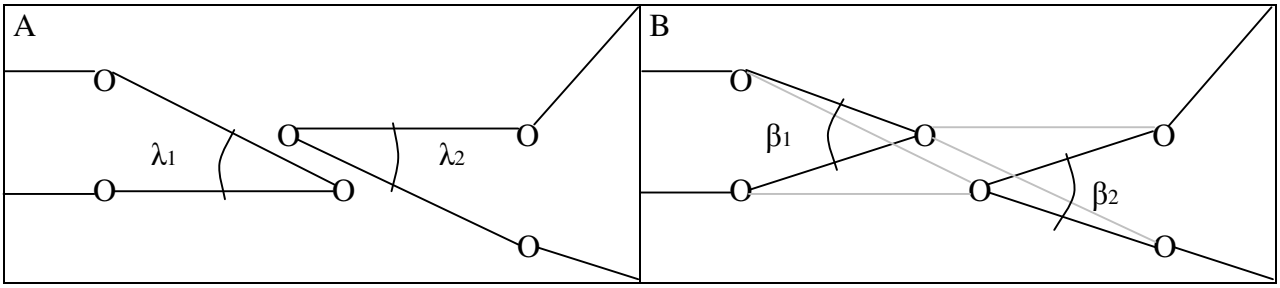


Figura 5.5 – Exemplo de movimento de troca simples de vértices baseado em ângulos

Na figura 5.5 temos um exemplo de movimento de troca simples de vértices entre duas rotas baseado em ângulos. Neste critério procuramos aumentar o ângulo formado pelas arestas que ligam os vértices candidatos à troca com seus respectivos vizinhos. Procuramos por um par de candidatos que maximize a expressão *varAngTroca* = { [($\beta_1 + \beta_2$) / 2] – [($\lambda_1 + \lambda_2$) / 2] }, onde λ_1 e λ_2 são ângulos correspondentes às arestas que foram removidas e β_1 e β_2 são ângulos correspondentes às arestas que foram incluídas. Trabalhamos com a média dos ângulos para poder posteriormente comparar com o resultado do movimento de doação.

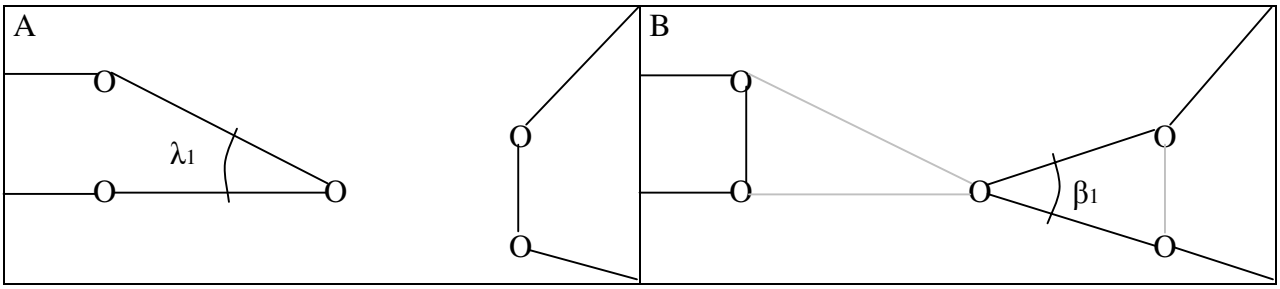


Figura 5.6 – Exemplo de movimento de inserção de vértice baseado em ângulos

Na figura 5.6 temos um exemplo de movimento de doação de vértice de uma rota para a outra. Neste critério procuramos aumentar a diferença entre o ângulo formado pelas arestas que passam a fazer parte da solução e, o ângulo formado pelas arestas que estão sendo removidas. Procuramos por um par de candidatos que maximize a expressão *varAngDoação* = { $\beta_1 - \lambda_1$ }, onde λ_1 é o ângulo formado entre as arestas que estão sendo removidas pelo movimento, e β_1 é o ângulo formado entre as arestas que estão sendo incluídas na solução.

Para o programa que realiza as simulações, implementamos um algoritmo seguindo a mesma idéia da nossa versão simplificada da Busca Tabu apresentada no capítulo 4. Com o apoio das figuras anteriores, podemos descrever os passos desse algoritmo da seguinte forma:

Metaheurística Busca Tabu (Ângulo)

1. Gere uma solução inicial através de uma heurística rápida qualquer.
2. Repita
3. ... Para todas as possíveis trocas de um vértice, utilizando movimentos baseados em ... ângulos, entre duas rotas diferentes faça:
4. ..... Se com a troca, nenhuma restrição do PRV for violada, faça:
5. ..... Se o movimento de troca não for um movimento tabu, faça:
6. ..... Verifique se o valor de *varAngTroca* = { [($\beta_1 + \beta_2$) / 2] – [($\lambda_1 + \lambda_2$) / 2] } é o ..... maior proposto nessa iteração (ver figura 5.5); caso seja, armazene o valor de ..... *varAngTroca* obtido e as arestas que participam do movimento.
7. ... Para todas as possíveis inserções de um vértice de uma rota para outra, utilizando ... movimentos baseados em ângulos, faça:
8. ..... Se com a inserção ou doação, nenhuma restrição do PRV for violada, faça:
9. ..... Se o movimento de inserção não for um movimento tabu, faça:
10. ..... Verifique se o valor de *varAngDoação* = { $\beta_1 – \lambda_1$ } é o maior proposto nessa ..... iteração (ver figura 5.6); caso seja, armazene o de *varAngDoação* obtido e as ..... arestas que participam do movimento.
11. ... Compare os valores *varAngTroca* e *varAngDoação*, selecionando o movimento ... correspondente da variável que possuir o maior valor.
12. ... Efetive o movimento; se esse movimento gerar uma solução melhor, do que a ... melhor solução encontrada até então, armazene os dados da nova solução.
13. ... Atualize as listas tabu com as arestas que participaram do movimento efetivado; ... armazene os dados da iteração no arquivo.
14. Até que número de iterações seja igual ao valor *X*, passado como parâmetro.
15. Aplique a heurística 2-Opt em cada rota.

Para os experimentos, além de observar os resultados obtidos pelo algoritmo da Busca Tabu com movimentos baseados em distâncias (capitulo 4), e os resultados obtidos pelo algoritmo da Busca Tabu com movimentos baseados em ângulos, decidimos realizar combinações desses dois algoritmos. Simplesmente, executamos as duas rotinas de maneira alternada, ou seja, numa determinada iteração efetivamos o movimento de intercâmbio de vértices com o algoritmo baseado em distâncias; em outra iteração, efetivamos o movimento com o algoritmo baseado em ângulos.

Elaboramos, de maneira puramente experimental, três tipos de combinações:

Combinação 1 (C1) – Os algoritmos são alternados com base no tamanho das listas tabu. Por exemplo, para listas de tamanho $T = 5$ movimentos, executamos cinco vezes o algoritmo baseado em distâncias e depois cinco vezes o algoritmo baseado em ângulos. A seguir temos uma tabela, com dados coletados do programa simulador, que ilustra a execução alternada dos algoritmos para listas tabu de tamanho = 3 movimentos.

I	Algoritmo	Tipo de Movimento	Custo Atual	Melhor Solução
13	Baseado em Distâncias	Troca Simples	2733.20	2694.12
14	Baseado em Distâncias	Troca Simples	2750.84	2694.12
15	Baseado em Distâncias	Inserção / Doação	2719.88	2694.12
16	Baseado em Ângulos	Inserção / Doação	2786.13	2694.12
17	Baseado em Ângulos	Inserção / Doação	2647.95	2647.95
18	Baseado em Ângulos	Inserção / Doação	2932.48	2647.95
19	Baseado em Distâncias	Troca Simples	2649.15	2647.95
20	Baseado em Distâncias	Inserção / Doação	2636.40	2636.40
21	Baseado em Distâncias	Inserção / Doação	2610.81	2610.81

Combinação 2 (C2) – Os algoritmos são alternados com base no número de vezes que pioram o custo atual da solução. Enquanto um dos algoritmo conseguir melhorar o valor obtido pelo movimento anterior, deixamos a rotina utilizá-lo. Em outras palavras, enquanto um algoritmo está sendo produtivo (melhorando o custo atual da solução), deixamos ele continuar trabalhando; a partir do momento que ele piora o custo atual, alternamos para o outro algoritmo.

Para não deixar a combinação muito rígida, implementamos o critério dos perdões. Na primeira vez o algoritmo *A* piora o custo atual, alternamos para o algoritmo *B* (*varPerdões* = 0). O algoritmo *B* poderá piorar uma vez o custo atual (*varPerdões* = 1), ou seja, sua piora será perdoada; quando ela piorar pela segunda vez passamos a utilizar novamente o algoritmo *A*. Agora o algoritmo *A* terá chance a dois perdões (*varPerdões* = 2), ou seja, poderá piorar o custo atual da solução por dois movimentos; quando ele piorar pela terceira vez, alternamos para o

algoritmo *B* e inicializamos novamente o número de perdões (*varPerdões* = 0). A seguir temos uma tabela, com dados coletados do programa simulador, que exemplifica a execução alternada dos algoritmos desta combinação.

I	varPerdões	Algoritmo	Tipo de Movimento	Custo Atual	Melhor Solução
01	0	Baseado em Distâncias	Troca Simples	2759.11	2759.11
02		Baseado em Distâncias	Troca Simples	2744.40	2744.40
03		Baseado em Distâncias	Inserção / Doação	2738.79	2738.79
04		Baseado em Distâncias	Inserção / Doação	2694.12	2694.12
05		Baseado em Distâncias	Troca Simples	Piorou 2713.95	2694.12
06	1	Baseado em Ângulos	Inserção / Doação	Piorou 3031.39	2694.12
07		Baseado em Ângulos	Inserção / Doação	2963.76	2694.12
08		Baseado em Ângulos	Inserção / Doação	2813.36	2694.12
09		Baseado em Ângulos	Inserção / Doação	2719.62	2694.12
10		Baseado em Ângulos	Troca Simples	2663.53	2663.53
11		Baseado em Ângulos	Troca Simples	2643.23	2643.23
12		Baseado em Ângulos	Inserção / Doação	Piorou 2648.66	2643.23
13	2	Baseado em Distâncias	Inserção / Doação	2646.44	2643.23
14		Baseado em Distâncias	Troca Simples	2598.11	2598.11
15		Baseado em Distâncias	Inserção / Doação	2572.52	2572.52
16		Baseado em Distâncias	Troca Simples	Piorou 2592.82	2572.52
17		Baseado em Distâncias	Troca Simples	Piorou 2607.53	2572.52
18		Baseado em Distâncias	Troca Simples	Piorou 2667.91	2572.52
19	0	Baseado em Ângulos	Inserção / Doação	2633.90	2572.52
20		Baseado em Ângulos	Inserção / Doação	2582.73	2572.52
21		Baseado em Ângulos	Inserção / Doação	2559.23	2559.23
22		Baseado em Ângulos	Inserção / Doação	Piorou 2846.90	2559.23
23	1	Baseado em Distâncias	Inserção / Doação	2559.01	2559.01
24		Baseado em Distâncias	Troca Simples	Piorou 2627.26	2559.01
25		Baseado em Distâncias	Troca Simples	Piorou 2656.09	2559.01
26	2	Baseado em Ângulos	Inserção / Doação	Piorou 2664.19	2559.01

Observando a tabela, notamos que a combinação inicia com movimentos de intercâmbio baseados em distâncias e com *varPerdões* = 0. Na quinta tentativa de melhoria da solução, foi efetivado um movimento que piorou o custo atual de 2694.12 para 2713.95; como essa piora não pode ser perdoada (*varPerdões* = 0), alternamos para o algoritmo com movimentos baseados em ângulos. Este, por sua vez, inicia seu trabalho na sexta iteração, piorando o custo atual de 2713.95 para 3031.39; como *varPerdões* = 1, essa primeira piora é perdoada, e a rotina segue melhorando o custo atual até a tentativa 12, quando o custo atual é piorado de 2643.23 para 2648.44. Essa segunda piora não é perdoada, e alternamos para o algoritmo baseado em distâncias que poderá agora piorar a solução por duas vezes sem que seja penalizado (*varPerdões* = 2). Este, por sua vez, promove melhoras do custo atual até a iteração 16, a partir da qual realiza dois movimentos de piora, perdoados, e um terceiro movimento que faz com que o procedimento volte a trabalhar com o algoritmo baseado em ângulos. E assim seguimos combinando os algoritmos.

Combinação 3 (C3) – É uma simples combinação, das duas combinações anteriores. Dividimos o número total de tentativas (iterações) em seis partes, e alternamos a execução das combinações 1 e 2 em cada uma dessas partes. A tabela seguinte, nos mostra um exemplo dessa prática, para 1500 tentativas de melhoria da solução.

1 a 250	251 a 500	501 a 750	751 a 1000	1001 a 1250	1251 a 1500
Combinação 1	Combinação 2	Combinação 1	Combinação 2	Combinação 1	Combinação 2

A primeira linha corresponde à variação das tentativas de melhoria da solução, ou iterações da rotina principal que utiliza os algoritmos que realizam movimentos de intercâmbio de vértices baseados em distâncias e ângulos.

5.2 Estratégia para Testes

Dividimos a etapa de testes em duas fases distintas. A primeira tem como finalidade observar algumas características demostradas pelos critérios e combinações apresentados no item anterior. Na segunda fase, selecionamos as duas abordagens que apresentaram maior eficiência na fase anterior, para verificar seus resultados diante de outras estratégias.

Nessas duas fases, os critérios estudados foram testados com sete dos quatorze problemas elaborados por Christofides, Mingozzi e Toth (1979). Os arquivos, com os dados de cada instância, foram copiados do endereço (http://neo.lcc.uma.es/radi-aeb/WebVRP/index.html). Foram escolhidos os problemas (1, 2, 3, 4, 5, 11, 12), pois estes possuem apenas restrições de capacidade. Nos problemas de 1 a 5, os vértices estão distribuídos de maneira aleatória no plano, enquanto que, nos problemas 11 e 12, os vértices estão distribuídos em grupos.

Ainda com relação aos arquivos, um fato curioso é que os problemas (4, 5, 11) possuem pares de vértices com as mesmas coordenadas. O arquivo do problema 11 por exemplo, possui dois vértices com as coordenadas (46;89). Já o arquivo do problema 4, possui dois vértices com as coordenadas (20;26) e dois vértices com as coordenadas (56;37). O arquivo do problema 5 possui um par de vértices para cada um dos seguintes pares de coordenadas: (20;26), (22;22) (30;60), (36;26), (45;35), (55;20), (55;45) e (56;37). Em outras palavras, é como se existisse uma cidade sobre a outra, ou em outra possível interpretação, é como se existissem dois pedidos de entrega para aquele mesmo local. Achamos interessante relatar isso, pois tomamos um susto ao perceber o fato; naquele momento pensamos que tínhamos deixado escapar algum defeito, pois a

interface pode, por exemplo, mostrar um vértice fazendo parte de duas rotas; só que na verdade existem dois vértices ali, um para cada rota.

Nas duas fases executamos os testes com cinco tamanhos de lista tabu diferentes (3, 5, 7, 9 e 11), onde os números entre parênteses representam o número de movimentos de intercâmbio de vértices. Em outras palavras, para o primeiro caso são armazenadas na lista tabu as arestas envolvidas nos três últimos movimentos de intercâmbio (troca ou doação) de vértices entre rotas. Lembrando que são utilizadas duas listas, uma para armazenar as arestas que foram removidas e outra para armazenar as arestas incluídas em movimentos anteriores.

Na primeira fase partimos de uma mesma solução de início para todos os critérios. Utilizamos a solução de partida adotada na heurística de Clarke e Wright, apresentada no capítulo anterior, onde cada vértice é atendido individualmente por uma rota. Procedemos assim pois algumas soluções de partida podem favorecer a um ou outro critério. Da forma que executamos, a solução de partida é ruim para todos os critérios, que devem realizar o agrupamento dos vértices desde o início com seus próprios movimentos de troca e doação. Para essa fase, fixamos o número de 1500 movimentos de intercâmbio de vértices, e cada problema foi executado 25 vezes.

Na segunda fase utilizamos vinte diferentes soluções de partida. Essas soluções foram obtidas com o uso da heurística de Clarke e Wright, iniciando a constante de Gaskell com 0.1, e incrementando esse valor em 0.1 até conseguirmos vinte soluções compostas por diferentes rotas. Nessa fase, fixamos o número de 1000 movimentos de intercâmbio de vértices, e cada problema foi executado 200 vezes.

Os testes foram executados numa estação com processador de 1.7GHz e 512 MB de RAM, no período de 23 de fevereiro à 18 de abril de 2004. Para isso, elaboramos uma rotina que capturava os parâmetros que seriam utilizados de um arquivo e disparava a execução do processo para cada grupo de parâmetros. Durante esse período, os resultados foram armazenados em arquivos no formato texto, para posterior análise e montagem das tabelas apresentadas a seguir.

5.3 Resultados Computacionais

Os resultados dos experimentos estão apresentados em tabelas. Inicialmente, apresentamos os resultados da primeira fase, seguido de alguns comentários. Na seqüência, apresentamos os resultados da segunda fase, seguido de comentários e de uma tabela comparativa, que contém valores obtidos por outras estratégias.

PROBLEMA 1 – 51 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

T	Busca Tabu Custo	Busca Tabu Angulo	Busca Tabu Combinação 1 – C1	Busca Tabu Combinação 2 – C2	Busca Tabu Combinação 3 – C3
3	592,85 A=9,52 B=72 (640,33) C=889 [354(1)] D=539 [539(180)] E=1428 [893(181)] F=0 [0(0)] G=91 (632,15) H=SIM (98)	551,50 A=9,27 B=61 (769,05) C=334 [184(3)] D=1105 [802(66)] E=0 [0(0)] F=1439 [986(69)] G=319 (552,37) H=SIM (344)	542,73 A=8,25 B=57 (683,86) C=987 [491(12)] D=456 [290(26)] E=720 [621(25)] F=723 [160(13)] G=613 (550,51) H=SIM (746)	563,74 A=8,33 B=72 (640,33) C=1215 [590(6)] D=213 [169(25)] E=986 [650(22)] F=442 [109(9)] G=507 (564,61) H=SIM (680)	549,01 A=8,37 B=72 (640,33) C=988 [502(13)] D=440 [328(27)] E=797 [552(26)] F=631 [278(14)] G=653 (550,72) H=SIM (1325)
5	**537,32** A=8,28 B=73 (647,86) C=1083 [459(6)] D=344 [344(12)] E=1427 [803(18)] F=0 [0(0)] G=572 (557,55) H=NÃO	**533,65** A=8,32 B=69 (679,00) C=792 [413(3)] D=639 [495(24)] E=0 [0(0)] F=1431 [908(27)] G=174 (553,69) H=NÃO	544,79 A=8,25 B=55 (688,93) C=915 [475(5)] D=530 [424(21)] E=720 [529(15)] F=725 [370(11)] G=114 (556,77) H=NÃO	558,33 A=8,30 B=73 (647,86) C=924 [438(15)] D=503 [418(22)] E=858 [572(24)] F=569 [284(13)] G=364 (562,27) H=NÃO	554,35 A=8,32 B=73 (647,86) C=949 [457(11)] D=478 [381(22)] E=790 [526(24)] F=637 [312(9)] G=1199 (576,99) H=NÃO
7	548,91 A=8,23 B=78 (643,99) C=1165 [514(13)] D=257 [255(12)] E=1422 [769(25)] F=0 [0(0)] G=595 (552,07) H=NÃO	541,77 A=8,30 B=59 (773,94) C=812 [425(2)] D=629 [497(17)] E=0 [0(0)] F=1441 [922(19)] G=152 (596,82) H=NÃO	558,87 A=8,40 B=64 (684,50) C=830 [434(6)] D=606 [466(25)] E=716 [529(18)] F=720 [371(13)] G=537 (561,96) H=NÃO	**535,19** A=8,32 B=78 (643,99) C=944 [479(8)] D=478 [383(21)] E=910 [629(22)] F=512 [233(7)] G=586 (551,81) H=NÃO	543,92 A=8,33 B=78 (643,99) C=890 [465(6)] D=532 [431(19)] E=802 [582(20)] F=620 [314(5)] G=289 (552,48) H=NÃO
9	551,06 A=8,35 B=70 (671,44) C=1049 [410(9)] D=381 [377(21)] E=1430 [787(30)] F=0 [0(0)] G=541 (563,77) H=NÃO	555,27 A=8,32 B=63 (745,90) C=761 [397(6)] D=676 [523(17)] E=0 [0(0)] F=1437 [920(23)] G=628 (589,61) H=NÃO	**531,83** A=8,32 B=63 (676,61) C=905 [487(5)] D=532 [412(21)] E=717 [536(17)] F=720 [363(9)] G=205 (571,20) H=NÃO	563,50 A=8,35 B=70 (671,44) C=932 [466(9)] D=498 [403(18)] E=905 [623(21)] F=525 [246(6)] G=253 (569,79) H=NÃO	**543,50** A=8,33 B=70 (671,44) C=850 [409(13)] D=580 [466(22)] E=810 [561(29)] F=620 [314(6)] G=325 (549,89) H=NÃO
11	573,42 A=8,42 B=82 (621,56) C=1043 [412(5)] D=375 [375(4)] E=1418 [787(9)] F=0 [0(0)] G=305 (585,57) H=NÃO	583,25 A=8,30 B=63 (737,28) C=766 [400(1)] D=671 [513(17)] E=0 [0(0)] F=1437 [913(18)] G=100 (593,90) H=NÃO	567,98 A=8,35 B=55 (761,09) C=840 [449(4)] D=605 [485(27)] E=719 [532(18)] F=726 [402(13)] G=209 (580,38) H=NÃO	557,20 A=8,33 B=82 (621,56) C=879 [423(9)] D=539 [436(11)] E=883 [603(19)] F=535 [256(1)] G=904 (573,07) H=NÃO	567,91 A=8,33 B=82 (621,56) C=917 [469(8)] D=501 [411(11)] E=805 [572(18)] F=613 [308(1)] G=182 (574,11) H=NÃO

Legenda

A = Tempo de processamento em minutos, em máquina com processador de 1.7GHz

B = Movimento de ocorrência do primeiro mínimo local (custo da solução no primeiro mínimo local)

C = Número de movimentos de troca de vértices entre rotas [C1(C2)]

C1 = Número de movimentos C que reduziram o valor do movimento anterior do processo

C2 = Número de movimentos C1 que encontraram o melhor valor do processo até aquele instante

D = Número de movimentos de inserção, ou doação de vértice de uma rota para outra [D1(D2)]

D1 = Número de movimentos D que reduziram o valor do movimento anterior do processo

D2 = Número de movimentos D1 que encontraram o melhor valor do processo até aquele instante

E = Número de movimentos que utilizaram o critério de menor custo para o intercâmbio de vértices [E1(E2)]

E1 = Número de movimentos E que reduziram o valor do movimento anterior do processo

E2 = Número de movimentos E1 que encontraram o melhor valor do processo até aquele instante

F = Número de movimentos que utilizaram o critério de maior ângulo para o intercâmbio de vértices [F1(F2)]

F1 = Número de movimentos F que reduziram o valor do movimento anterior do processo

F2 = Número de movimentos F1 que encontraram o melhor valor do processo até aquele instante

G = Movimento de ocorrência da melhor solução (custo da melhor solução – antes da aplicação da 2-Opt)

H = Ocorrência de ciclagem (a partir de que movimento a ciclagem foi percebida)

PROBLEMA 2 – 76 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

T	Busca Tabu Custo	Busca Tabu Angulo	Busca Tabu Combinação 1 – C1	Busca Tabu Combinação 2 – C2	Busca Tabu Combinação 3 – C3
3	909,35 A=21,70 B=109 (962,86) C=1386 [689(175)] D=5 [5(3)] E=1391 [694(178)] F=0 [0(0)] G=125 (942,63) H=SIM (138)	904,78 A=21,53 B=99 (1027,55) C=1283 [792(3)] D=118 [92(14)] E=0 [0(0)] F=1401 [884(17)] G=132 (918,44) H=SIM (309)	866,78 A=21,33 B=93 (1014,40) C=1195 [672(7)] D=212 [129(27)] E=702 [483(20)] F=705 [318(14)] G=211 (870,37) H=SIM (276)	**853,78** A=21,17 B=109 (962,86) C=1304 [650(19)] D=87 [73(15)] E=914 [579(28)] F=477 [144(6)] G=379 (867,13) H=SIM (862)	**854,23** A=21,15 B=109 (962,86) C=1208 [644(26)] D=183 [135(13)] E=793 [559(31)] F=598 [220(8)] G=872 (863,01) H=NÃO
5	863,73 A=21,10 B=109 (962,86) C=1365 [673(102)] D=26 [26(5)] E=1391 [699(107)] F=0 [0(0)] G=366 (912,70) H=SIM (398)	901,04 A=21,50 B=107 (964,89) C=1024 [556(4)] D=369 [298(11)] E=0 [0(0)] F=1393 [854(15)] G=738 (912,02) H=NÃO	875,32 A=21,28 B=96 (998,02) C=1211 [686(8)] D=193 [147(18)] E=700 [508(18)] F=704 [325(8)] G=613 (895,39) H=NÃO	856,81 A=21,17 B=109 (962,86) C=1096 [552(16)] D=295 [212(9)] E=923 [610(21)] F=468 [154(4)] G=370 (863,70) H=NÃO	870,11 A=21,18 B=109 (962,86) C=1082 [571(16)] D=309 [223(12)] E=826 [582(20)] F=565 [212(8)] G=636 (873,53) H=NÃO
7	**860,93** A=21,22 B=94 (1005,65) C=1264 [565(15)] D=142 [142(23)] E=1406 [707(38)] F=0 [0(0)] G=1265 (886,28) H=NÃO	**882,10** A=21,65 B=106 (971,93) C=980 [527(11)] D=414 [337(16)] E=0 [0(0)] F=1394 [864(27)] G=366 (883,11) H=NÃO	861,10 A=21,42 B=83 (1076,89) C=1109 [577(23)] D=308 [252(33)] E=709 [501(39)] F=708 [328(17)] G=962 (862,84) H=NÃO	875,61 A=21,17 B=94 (1005,65) C=1234 [703(19)] D=172 [136(18)] E=952 [669(34)] F=454 [170(3)] G=623 (882,49) H=NÃO	886,46 A=21,18 B=94 (1005,65) C=1149 [616(20)] D=257 [209(17)] E=827 [583(34)] F=579 [242(3)] G=1176 (891,66) H=NÃO
9	873,38 A=21,27 B=109 (983,51) C=1161 [505(24)] D=230 [230(30)] E=1391 [735(54)] F=0 [0(0)] G=1347 (888,04) H=NÃO	897,05 A=21,57 B=106 (971,93) C=937 [480(12)] D=457 [362(9)] E=0 [0(0)] F=1394 [842(21)] G=547 (902,35) H=NÃO	872,81 A=21,32 B=86 (1051,96) C=1098 [640(22)] D=316 [248(14)] E=708 [548(26)] F=706 [340(10)] G=657 (891,46) H=NÃO	880,05 A=21,27 B=109 (983,51) C=1103 [623(19)] D=288 [225(19)] E=943 [672(27)] F=448 [176(11)] G=273 (889,29) H=NÃO	867,42 A=21,28 B=109 (983,51) C=1130 [627(20)] D=261 [206(20)] E=782 [557(28)] F=609 [276(12)] G=332 (874,02) H=NÃO
11	866,29 A=21,33 B=110 (955,72) C=1258 [562(23)] D=132 [132(15)] E=1390 [694(38)] F=0 [0(0)] G=371 (902,73) H=NÃO	911,26 A=21,60 B=105 (959,63) C=914 [476(5)] D=481 [367(5)] E=0 [0(0)] F=1395 [843(10)] G=136 (920,23) H=NÃO	**853,84** A=21,40 B=82 (1107,34) C=1077 [578(22)] D=341 [283(41)] E=708 [526(45)] F=710 [335(18)] G=342 (863,39) H=NÃO	891,05 A=21,32 B=110 (955,72) C=1072 [589(11)] D=318 [246(16)] E=922 [645(26)] F=468 [190(1)] G=340 (899,37) H=NÃO	881,48 A=21,55 B=110 (955,72) C=956 [507(12)] D=434 [345(16)] E=799 [574(25)] F=591 [278(3)] G=525 (888,93) H=NÃO

Legenda

A = Tempo de processamento em minutos, em máquina com processador de 1.7GHz

B = Movimento de ocorrência do primeiro mínimo local (custo da solução no primeiro mínimo local)

C = Número de movimentos de troca de vértices entre rotas [C1(C2)]

C1 = Número de movimentos C que reduziram o valor do movimento anterior do processo

C2 = Número de movimentos C1 que encontraram o melhor valor do processo até aquele instante

D = Número de movimentos de inserção, ou doação de vértice de uma rota para outra [D1(D2)]

D1 = Número de movimentos D que reduziram o valor do movimento anterior do processo

D2 = Número de movimentos D1 que encontraram o melhor valor do processo até aquele instante

E = Número de movimentos que utilizaram o critério de menor custo para o intercâmbio de vértices [E1(E2)]

E1 = Número de movimentos E que reduziram o valor do movimento anterior do processo

E2 = Número de movimentos E1 que encontraram o melhor valor do processo até aquele instante

F = Número de movimentos que utilizaram o critério de maior ângulo para o intercâmbio de vértices [F1(F2)]

F1 = Número de movimentos F que reduziram o valor do movimento anterior do processo

F2 = Número de movimentos F1 que encontraram o melhor valor do processo até aquele instante

G = Movimento de ocorrência da melhor solução (custo da melhor solução – antes da aplicação da 2-Opt)

H = Ocorrência de ciclagem (a partir de que movimento a ciclagem foi percebida)

PROBLEMA 3 – 101 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

T	Busca Tabu Custo	Busca Tabu Angulo	Busca Tabu Combinação 1 – C1	Busca Tabu Combinação 2 – C2	Busca Tabu Combinação 3 – C3
3	992,36 A=42,32 B=177 (1038,79) C=452 [114(2)] D=871 [871(10)] E=1323 [985(12)] F=0 [0(0)] G=194 (1031,99) H=SIM (238)	897,28 A=39,35 B=124 (1312,57) C=635 [209(8)] D=741 [584(68)] E=0 [0(0)] F=1376 [793(76)] G=412 (916,48) H=SIM (583)	844,40 A=39,07 B=141 (1030,97) C=615 [256(11)] D=744 [557(51)] E=678 [504(39)] F=681 [309(23)] G=1113 (851,47) H=SIM (1150)	844,19 A=39,67 B=177 (1038,79) C=687 [302(16)] D=636 [444(58)] E=907 [618(67)] F=416 [128(7)] G=605 (847,05) H=SIM (910)	853,26 A=38,77 B=177 (1038,79) C=1216 [489(12)] D=107 [86(44)] E=680 [308(42)] F=643 [267(14)] G=327 (870,78) H=SIM (376)
5	992,36 A=42,28 B=176 (1038,79) C=587 [90(2)] D=737 [737(6)] E=1324 [827(8)] F=0 [0(0)] G=189 (1031,99) H=SIM (281)	851,21 A=40,23 B=124 (1317,04) C=556 [239(29)] D=820 [656(78)] E=0 [0(0)] F=1376 [895(107)] G=842 (860,32) H=NÃO	848,00 A=38,70 B=135 (1080,71) C=576 [243(23)] D=789 [606(50)] E=680 [477(42)] F=685 [372(31)] G=1218 (854,00) H=NÃO	853,68 A=39,42 B=176 (1038,79) C=706 [305(14)] D=618 [482(51)] E=796 [526(57)] F=528 [261(8)] G=481 (858,62) H=NÃO	862,23 A=39,08 B=176 (1038,79) C=698 [322(21)] D=626 [498(55)] E=717 [500(61)] F=607 [320(15)] G=1200 (862,23) H=NÃO
7	954,31 A=41,63 B=178 (1037,19) C=613 [176(5)] D=709 [709(32)] E=1322 [885(37)] F=0 [0(0)] G=1428 (980,63) H=NÃO	**850,06** A=40,25 B=124 (1317,53) C=541 [254(19)] D=835 [666(80)] E=0 [0(0)] F=1376 [920(99)] G=975 (855,97) H=NÃO	848,96 A=38,78 B=134 (1086,74) C=617 [289(14)] D=749 [567(59)] E=681 [510(52)] F=685 [346(21)] G=302 (856,16) H=NÃO	847,69 A=39,12 B=178 (1037,19) C=605 [250(14)] D=717 [561(44)] E=831 [575(43)] F=491 [236(15)] G=1423 (847,69) H=NÃO	**844,76** A=38,90 B=178 (1037,19) C=619 [276(13)] D=703 [558(39)] E=688 [477(33)] F=634 [357(19)] G=1484 (846,49) H=NÃO
9	927,58 A=40,50 B=180 (1032,19) C=768 [241(8)] D=552 [552(25)] E=1320 [793(33)] F=0 [0(0)] G=579 (945,12) H=NÃO	872,57 A=39,98 B=121 (1370,69) C=510 [245(18)] D=869 [690(75)] E=0 [0(0)] F=1379 [935(93)] G=961 (891,4) H=NÃO	861,69 A=39,33 B=122 (1244,57) C=564 [254(24)] D=814 [630(91)] E=690 [494(67)] F=688 [390(48)] G=1433 (870,47) H=NÃO	**838,82** A=39,63 B=180 (1032,19) C=549 [226(5)] D=771 [600(48)] E=783 [536(34)] F=537 [290(19)] G=1320 (851,05) H=NÃO	845,65 A=39,12 B=180 (1032,19) C=495 [218(7)] D=825 [637(39)] E=720 [526(32)] F=600 [329(14)] G=1414 (846,78) H=NÃO
11	**880,50** A=40,12 B=195 (963,90) C=774 [247(11)] D=531 [531(41)] E=1305 [778(52)] F=0 [0(0)] G=932 (890,74) H=NÃO	876,54 A=39,95 B=121 (1370,69) C=508 [240(16)] D=871 [692(67)] E=0 [0(0)] F=1379 [932(83)] G=920 (886,48) H=NÃO	**838,16** A=38,80 B=130 (1190,91) C=605 [286(24)] D=765 [571(55)] E=686 [485(60)] F=684 [372(19)] G=889 (857,03) H=NÃO	856,99 A=39,87 B=195 (963,90) C=580 [256(20)] D=725 [552(38)] E=866 [617(50)] F=439 [191(8)] G=1356 (863,94) H=NÃO	868,33 A=40,33 B=195 (963,90) C=549 [260(15)] D=756 [579(34)] E=732 [518(43)] F=573 [321(6)] G=1337 (882,67) H=NÃO

Legenda

A = Tempo de processamento em minutos, em máquina com processador de 1.7GHz

B = Movimento de ocorrência do primeiro mínimo local (custo da solução no primeiro mínimo local)

C = Número de movimentos de troca de vértices entre rotas [C1(C2)]

C1 = Número de movimentos C que reduziram o valor do movimento anterior do processo

C2 = Número de movimentos C1 que encontraram o melhor valor do processo até aquele instante

D = Número de movimentos de inserção, ou doação de vértice de uma rota para outra [D1(D2)]

D1 = Número de movimentos D que reduziram o valor do movimento anterior do processo

D2 = Número de movimentos D1 que encontraram o melhor valor do processo até aquele instante

E = Número de movimentos que utilizaram o critério de menor custo para o intercâmbio de vértices [E1(E2)]

E1 = Número de movimentos E que reduziram o valor do movimento anterior do processo

E2 = Número de movimentos E1 que encontraram o melhor valor do processo até aquele instante

F = Número de movimentos que utilizaram o critério de maior ângulo para o intercâmbio de vértices [F1(F2)]

F1 = Número de movimentos F que reduziram o valor do movimento anterior do processo

F2 = Número de movimentos F1 que encontraram o melhor valor do processo até aquele instante

G = Movimento de ocorrência da melhor solução (custo da melhor solução – antes da aplicação da 2-Opt)

H = Ocorrência de ciclagem (a partir de que movimento a ciclagem foi percebida)

PROBLEMA 12 – 101 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

T	Busca Tabu Custo	Busca Tabu Angulo	Busca Tabu Combinação 1 – C1	Busca Tabu Combinação 2 – C2	Busca Tabu Combinação 3 – C3
3	882,72 A=40,92 B=181 (1007,17) C=742 [168(5)] D=577 [577(88)] E=1319 [745(93)] F=0 [0(0)] G=203 (943,31) H=SIM (221)	929,02 A=39,83 B=103 (1793,11) C=829 [416(19)] D=568 [387(66)] E=0 [0(0)] F=1397 [803(85)] G=453 (934,61) H=SIM (623)	959,54 A=41,60 B=94 (1778,51) C=20 [15(7)] D=1386 [734(70)] E=702 [701(49)] F=704 [48(28)] G=239 (977,89) H=SIM (299)	**858,62** A=40,72 B=181 (1007,17) C=622 [300(10)] D=697 [416(18)] E=975 [674(27)] F=344 [42(1)] G=249 (910,74) H=SIM (601)	**857,97** A=40,03 B=181 (1007,17) C=539 [298(10)] D=780 [482(24)] E=776 [612(33)] F=543 [168(1)] G=282 (891,41) H=NÃO
5	**831,55** A=40,70 B=183 (1006,58) C=641 [177(6)] D=676 [676(17)] E=1317 [853(23)] F=0 [0(0)] G=419 (902,97) H=NÃO	915,32 A=40,28 B=103 (1793,43) C=687 [336(5)] D=710 [558(77)] E=0 [0(0)] F=1397 [894(82)] G=805 (921,05) H=SIM (971)	929,22 A=39,43 B=125 (1155,54) C=612 [352(10)] D=763 [555(30)] E=685 [564(29)] F=690 [343(11)] G=1495 (948,57) H=NÃO	917,37 A=39,52 B=183 (1006,58) C=736 [345(6)] D=581 [416(14)] E=889 [611(20)] F=428 [150(0)] G=318 (933,71) H=NÃO	876,37 A=39,83 B=183 (1006,58) C=494 [263(12)] D=823 [529(22)] E=785 [619(31)] F=532 [173(3)] G=1050 (906,63) H=NÃO
7	835,47 A=41,12 B=186 (999,83) C=687 [221(7)] D=627 [627(26)] E=1314 [848(33)] F=0 [0(0)] G=502 (897,37) H=NÃO	**898,56** A=40,60 B=102 (1825,93) C=428 [237(11)] D=970 [719(82)] E=0 [0(0)] F=1398 [956(93)] G=927 (902,87) H=NÃO	904,47 A=39,73 B=93 (1641,28) C=508 [296(19)] D=899 [633(91)] E=702 [591(83)] F=705 [338(27)] G=1001 (923,41) H=NÃO	882,61 A=40,58 B=186 (999,83) C=499 [224(19)] D=815 [580(43)] E=928 [672(55)] F=386 [132(7)] G=900 (906,10) H=NÃO	878,23 A=40,07 B=186 (999,83) C=476 [240(13)] D=838 [566(42)] E=792 [617(46)] F=522 [189(9)] G=1071 (915,94) H=NÃO
9	874,15 A=40,72 B=187 (1021,95) C=961 [365(10)] D=352 [352(21)] E=1313 [717(31)] F=0 [0(0)] G=787 (935,07) H=NÃO	918,25 A=40,57 B=102 (1839,54) C=377 [226(16)] D=1021 [770(89)] E=0 [0(0)] F=1398 [996(105)] G=1318 (923,57) H=NÃO	893,02 A=40,12 B=102 (1651,54) C=486 [274(22)] D=912 [657(101)] E=699 [577(91)] F=699 [354(32)] G=981 (893,50) H=NÃO	872,43 A=40,20 B=187 (1021,95) C=610 [304(28)] D=703 [515(50)] E=942 [695(73)] F=371 [124(5)] G=947 (894,36) H=NÃO	889,94 A=40,25 B=187 (1021,95) C=519 [269(24)] D=794 [565(81)] E=794 [610(93)] F=519 [224(12)] G=1452 (902,57) H=NÃO
11	854,26 A=40,85 B=181 (1020,23) C=788 [275(16)] D=531 [531(31)] E=1319 [806(47)] F=0 [0(0)] G=786 (916,03) H=NÃO	924,53 A=40,65 B=102 (1841,02) C=418 [250(18)] D=980 [726(89)] E=0 [0(0)] F=1398 [976(107)] G=690 (939,10) H=NÃO	**857,41** A=39,83 B=105 (1480,42) C=493 [333(20)] D=902 [627(66)] E=697 [620(63)] F=698 [340(23)] G=1463 (880,08) H=NÃO	870,84 A=40,00 B=181 (1020,23) C=555 [259(18)] D=764 [522(48)] E=946 [676(63)] F=373 [105(3)] G=885 (932,52) H=NÃO	907,32 A=40,22 B=181 (1020,23) C=497 [261(12)] D=822 [584(29)] E=785 [598(33)] F=534 [247(8)] G=1260 (923,13) H=NÃO

Legenda

A = Tempo de processamento em minutos, em máquina com processador de 1.7GHz

B = Movimento de ocorrência do primeiro mínimo local (custo da solução no primeiro mínimo local)

C = Número de movimentos de troca de vértices entre rotas [C1(C2)]

C1 = Número de movimentos C que reduziram o valor do movimento anterior do processo

C2 = Número de movimentos C1 que encontraram o melhor valor do processo até aquele instante

D = Número de movimentos de inserção, ou doação de vértice de uma rota para outra [D1(D2)]

D1 = Número de movimentos D que reduziram o valor do movimento anterior do processo

D2 = Número de movimentos D1 que encontraram o melhor valor do processo até aquele instante

E = Número de movimentos que utilizaram o critério de menor custo para o intercâmbio de vértices [E1(E2)]

E1 = Número de movimentos E que reduziram o valor do movimento anterior do processo

E2 = Número de movimentos E1 que encontraram o melhor valor do processo até aquele instante

F = Número de movimentos que utilizaram o critério de maior ângulo para o intercâmbio de vértices [F1(F2)]

F1 = Número de movimentos F que reduziram o valor do movimento anterior do processo

F2 = Número de movimentos F1 que encontraram o melhor valor do processo até aquele instante

G = Movimento de ocorrência da melhor solução (custo da melhor solução – antes da aplicação da 2-Opt)

H = Ocorrência de ciclagem (a partir de que movimento a ciclagem foi percebida)

PROBLEMA 11 – 121 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

T	Busca Tabu Custo	Busca Tabu Angulo	Busca Tabu Combinação 1 – C1	Busca Tabu Combinação 2 – C2	Busca Tabu Combinação 3 – C3
3	1539,03 A=61,37 B=258 (1698,44) C=738 [251(10)] D=504 [504(140)] E=1242 [755(150)] F=0 [0(0)] G=299 (1561,29) H=SIM (302)	1348,60 A=56,42 B=131 (3575,78) C=1160 [593(165)] D=209 [179(131)] E=0 [0(0)] F=1369 [772(296)] G=378 (1370,81) H=SIM (384)	1292,56 A=55,50 B=154 (3059,96) C=694 [465(29)] D=652 [466(104)] E=672 [611(79)] F=674 [320(54)] G=1411 (1300,36) H=NÃO	1237,55 A=55,22 B=258 (1698,44) C=879 [486(45)] D=363 [277(64)] E=844 [604(102)] F=398 [159(7)] G=1077 (1265,86) H=SIM (1212)	1274,76 A=55,38 B=253 (1698,52) C=695 [364(30)] D=552 [392(40)] E=722 [539(55)] F=525 [217(15)] G=653 (1280,74) H=SIM (1363)
5	**1410,57** A=60,53 B=268 (1592,35) C=608 [221(18)] D=624 [622(43)] E=1232 [843(61)] F=0 [0(0)] G=805 (1441,56) H=SIM (499)	**1343,80** A=56,95 B=131 (3576,73) C=610 [347(9)] D=759 [616(149)] E=0 [0(0)] F=1369 [963(158)] G=425 (1345,55) H=NÃO	**1239,07** A=55,45 B=136 (3676,80) C=776 [507(75)] D=588 [466(149)] E=680 [596(148)] F=684 [377(76)] G=1397 (1245,79) H=NÃO	**1219,40** A=55,27 B=267 (1597,03) C=703 [417(75)] D=530 [400(82)] E=890 [682(146)] F=343 [135(11)] G=1244 (1231,14) H=NÃO	1281,48 A=54,95 B=255 (1700,07) C=706 [455(13)] D=539 [379(31)] E=729 [606(32)] F=516 [228(12)] G=784 (1305,69) H=NÃO
7	1539,53 A=61,35 B=276 (1591,14) C=629 [251(12)] D=595 [544(33)] E=1224 [795(45)] F=0 [0(0)] G=812 (1566,76) H=SIM (861)	1477,60 A=56,37 B=131 (3576,73) C=733 [436(6)] D=636 [522(116)] E=0 [0(0)] F=1369 [958(122)] G=300 (1487,69) H=NÃO	1281,03 A=55,45 B=167 (2771,57) C=756 [528(36)] D=577 [446(102)] E=667 [618(91)] F=666 [356(47)] G=372 (1290,43) H=NÃO	1317,22 A=55,35 B=270 (1597,17) C=715 [410(41)] D=515 [388(65)] E=870 [654(103)] F=360 [144(3)] G=777 (1325,80) H=NÃO	1267,76 A=55,62 B=257 (1621,67) C=705 [457(37)] D=538 [415(51)] E=723 [603(69)] F=520 [269(19)] G=860 (1283,98) H=NÃO
9	1531,79 A=60,82 B=269 (1597,70) C=782 [284(19)] D=449 [426(43)] E=1231 [710(62)] F=0 [0(0)] G=415 (1563,97) H=NÃO	1416,21 A=57,00 B=131 (3575,16) C=708 [418(9)] D=661 [536(127)] E=0 [0(0)] F=1369 [954(136)] G=357 (1433,21) H=NÃO	1353,64 A=55,75 B=160 (2971,11) C=702 [509(37)] D=638 [490(105)] E=672 [632(92)] F=668 [367(50)] G=568 (1360,00) H=NÃO	1276,29 A=55,30 B=266 (1602,38) C=759 [430(32)] D=475 [380(57)] E=834 [622(84)] F=400 [188(5)] G=436 (1288,76) H=NÃO	**1259,56** A=55,32 B=259 (1604,27) C=704 [454(34)] D=537 [400(65)] E=747 [616(76)] F=494 [238(23)] G=1500 (1293,62) H=NÃO
11	1504,44 A=61,73 B=300 (1541,17) C=764 [297(4)] D=436 [423(13)] E=1200 [720(17)] F=0 [0(0)] G=1451 (1540,35) H=NÃO	1424,37 A=57,13 B=132 (3575,44) C=598 [391(26)] D=770 [626(122)] E=0 [0(0)] F=1368[1017(148)] G=945 (1440,31) H=NÃO	1293,15 A=56,12 B=131 (3767,80) C=720 [511(93)] D=649 [507(144)] E=686 [626(170)] F=683 [392(67)] G=1439 (1318,20) H=NÃO	1269,17 A=56,50 B=300 (1541,17) C=632 [410(44)] D=568 [462(68)] E=955 [791(105)] F=245 [81(7)] G=1467 (1296,72) H=NÃO	1293,05 A=55,45 B=261 (1702,26) C=631 [424(51)] D=608 [472(64)] E=737 [630(90)] F=502 [266(25)] G=1105 (1305,72) H=NÃO

Legenda

A = Tempo de processamento em minutos, em máquina com processador de 1.7GHz

B = Movimento de ocorrência do primeiro mínimo local (custo da solução no primeiro mínimo local)

C = Número de movimentos de troca de vértices entre rotas [C1(C2)]

C1 = Número de movimentos C que reduziram o valor do movimento anterior do processo

C2 = Número de movimentos C1 que encontraram o melhor valor do processo até aquele instante

D = Número de movimentos de inserção, ou doação de vértice de uma rota para outra [D1(D2)]

D1 = Número de movimentos D que reduziram o valor do movimento anterior do processo

D2 = Número de movimentos D1 que encontraram o melhor valor do processo até aquele instante

E = Número de movimentos que utilizaram o critério de menor custo para o intercâmbio de vértices [E1(E2)]

E1 = Número de movimentos E que reduziram o valor do movimento anterior do processo

E2 = Número de movimentos E1 que encontraram o melhor valor do processo até aquele instante

F = Número de movimentos que utilizaram o critério de maior ângulo para o intercâmbio de vértices [F1(F2)]

F1 = Número de movimentos F que reduziram o valor do movimento anterior do processo

F2 = Número de movimentos F1 que encontraram o melhor valor do processo até aquele instante

G = Movimento de ocorrência da melhor solução (custo da melhor solução – antes da aplicação da 2-Opt)

H = Ocorrência de ciclagem (a partir de que movimento a ciclagem foi percebida)

PROBLEMA 4 – 151 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

T	Busca Tabu Custo	Busca Tabu Angulo	Busca Tabu Combinação 1 – C1	Busca Tabu Combinação 2 – C2	Busca Tabu Combinação 3 – C3
3	1303,78 A=104,97 B=311 (1378,71) C=592 [293(149)] D=597 [597(12)] E=1189 [890(161)] F=0 [0(0)] G=341 (1341,53) H=SIM (351)	1143,01 A=99,57 B=177 (2154,77) C=825 [369(129)] D=498 [456(134)] E=0 [0(0)] F=1323 [825(263)] G=556 (1155,64) H=SIM (569)	**1068,10** A=99,40 B=171 (1820,94) C=615 [316(51)] D=714 [525(132)] E=663 [513(104)] F=666 [328(79)] G=1186 (1104,42) H=SIM (1181)	1110,77 A=100,47 B=311 (1378,71) C=578 [312(27)] D=611 [483(62)] E=827 [629(78)] F=362 [166(11)] G=647 (1128,02) H=SIM (849)	1072,02 A=99,58 B=255 (1556,32) C=615 [285(41)] D=630 [479(107)] E=694 [537(102)] F=551 [227(46)] G=1212 (1075,04) H=SIM (1375)
5	1297,61 A=106,68 B=297 (1435,12) C=564 [146(20)] D=639 [639(57)] E=1203 [785(77)] F=0 [0(0)] G=916 (1325,06) H=SIM (978)	1117,22 A=96,83 B=178 (2154,68) C=906 [438(20)] D=416 [375(138)] E=0 [0(0)] F=1322 [813(158)] G=493 (1137,22) H=SIM (690)	1108,37 A=97,77 B=215 (1525,04) C=591 [348(16)] D=694 [479(78)] E=640 [547(63)] F=645 [280(31)] G=694 (1113,85) H=SIM (889)	1107,78 A=99,90 B=297 (1435,12) C=694 [351(35)] D=509 [365(105)] E=853 [609(122)] F=350 [107(18)] G=1497 (1121,53) H=NÃO	**1068,14** A=98,30 B=265 (1505,36) C=643 [323(26)] D=592 [484(108)] E=652 [491(94)] F=583 [316(40)] G=1404 (1081,00) H=NÃO
7	1285,32 A=104,58 B=310 (1414,37) C=570 [133(11)] D=620 [620(34)] E=1190 [753(45)] F=0 [0(0)] G=728 (1320,94) H=NÃO	**1070,34** A=102,60 B=178 (2154,68) C=558 [269(36)] D=764 [633(146)] E=0 [0(0)] F=1322 [902(182)] G=1217 (1086,80) H=NÃO	1081,94 A=97,90 B=190 (1667,69) C=598 [326(34)] D=712 [565(104)] E=653 [514(85)] F=657 [377(53)] G=1452 (1092,97) H=NÃO	1087,24 A=99,72 B=310 (1414,37) C=576 [263(32)] D=614 [492(100)] E=734 [517(101)] F=456 [238(31)] G=1055 (1096,75) H=NÃO	1080,68 A=98,37 B=263 (1516,59) C=529 [258(37)] D=708 [573(115)] E=688 [532(108)] F=549 [299(44)] G=1476 (1087,54) H=NÃO
9	**1135,04** A=102,30 B=304 (1421,92) C=712 [271(35)] D=484 [483(85)] E=1196 [754(120)] F=0 [0(0)] G=811 (1177,57) H=NÃO	1096,54 A=100,45 B=178 (2154,68) C=527 [276(34)] D=795 [645(142)] E=0 [0(0)] F=1322 [921(176)] G=1470 (1099,89) H=NÃO	1091,28 A=97,40 B=207 (1559,05) C=523 [277(18)] D=770 [591(117)] E=645 [518(93)] F=648 [350(42)] G=1450 (1092,52) H=NÃO	1109,51 A=99,55 B=304 (1421,92) C=530 [241(29)] D=666 [559(96)] E=774 [576(105)] F=422 [224(20)] G=1388 (1124,35) H=NÃO	1097,12 A=97,98 B=260 (1502,65) C=551 [290(36)] D=689 [572(95)] E=658 [503(88)] F=582 [359(43)] G=1497 (1097,57) H=NÃO
11	1157,94 A=104,75 B=282 (1489,31) C=629 [161(34)] D=589 [589(91)] E=1218 [750(125)] F=0 [0(0)] G=1483 (1204,62) H=NÃO	1121,90 A=101,30 B=178 (2154,68) C=542 [279(27)] D=780 [641(140)] E=0 [0(0)] F=1322 [920(167)] G=1351 (1128,90) H=NÃO	1107,83 A=97,58 B=215 (1531,72) C=543 [292(35)] D=742 [590(111)] E=642 [499(105)] F=643 [383(41)] G=1420 (1116,99) H=NÃO	**1073,31** A=98,43 B=282 (1489,31) C=478 [221(36)] D=740 [576(106)] E=825 [614(127)] F=393 [183(15)] G=1044 (1076,22) H=NÃO	1095,10 A=98,58 B=263 (1499,79) C=506 [265(39)] D=731 [567(113)] E=685 [527(108)] F=552 [305(44)] G=1388 (1102,48) H=NÃO

Legenda

A = Tempo de processamento em minutos, em máquina com processador de 1.7GHz

B = Movimento de ocorrência do primeiro mínimo local (custo da solução no primeiro mínimo local)

C = Número de movimentos de troca de vértices entre rotas [C1(C2)]

C1 = Número de movimentos C que reduziram o valor do movimento anterior do processo

C2 = Número de movimentos C1 que encontraram o melhor valor do processo até aquele instante

D = Número de movimentos de inserção, ou doação de vértice de uma rota para outra [D1(D2)]

D1 = Número de movimentos D que reduziram o valor do movimento anterior do processo

D2 = Número de movimentos D1 que encontraram o melhor valor do processo até aquele instante

E = Número de movimentos que utilizaram o critério de menor custo para o intercâmbio de vértices [E1(E2)]

E1 = Número de movimentos E que reduziram o valor do movimento anterior do processo

E2 = Número de movimentos E1 que encontraram o melhor valor do processo até aquele instante

F = Número de movimentos que utilizaram o critério de maior ângulo para o intercâmbio de vértices [F1(F2)]

F1 = Número de movimentos F que reduziram o valor do movimento anterior do processo

F2 = Número de movimentos F1 que encontraram o melhor valor do processo até aquele instante

G = Movimento de ocorrência da melhor solução (custo da melhor solução – antes da aplicação da 2-Opt)

H = Ocorrência de ciclagem (a partir de que movimento a ciclagem foi percebida)

PROBLEMA 5 – 200 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

T	Busca Tabu Custo	Busca Tabu Angulo	Busca Tabu Combinação 1 – C1	Busca Tabu Combinação 2 – C2	Busca Tabu Combinação 3 – C3
3	1624,05 A=201,15 B=383 (1732,28) C=710 [116(65)] D=407 [407(60)] E=1117 [523(125)] F=0 [0(0)] G=473 (1641,89) H=SIM (493)	1597,78 A=208,75 B=246 (2579,45) C=606 [269(15)] D=648 [514(113)] E=0 [0(0)] F=1254 [783(128)] G=597 (1630,04) H=SIM (636)	1546,24 A=200,68 B=261 (2181,04) C=370 [209(33)] D=869 [580(116)] E=618 [541(89)] F=621 [248(60)] G=538 (1550,21) H=SIM (609)	1557,68 A=200,42 B=383 (1732,28) C=498 [230(18)] D=619 [356(37)] E=829 [564(43)] F=288 [22(12)] G=565 (1569,34) H=SIM (680)	1461,72 A=192,50 B=295 (2007,20) C=553 [235(50)] D=652 [403(115)] E=713 [527(124)] F=492 [111(41)] G=777 (1465,00) H=SIM (1023)
5	1481,87 A=199,03 B=379 (1700,52) C=754 [297(34)] D=367 [367(64)] E=1121 [664(98)] F=0 [0(0)] G=1191 (1498,17) H=NÃO	1459,42 A=202,10 B=246 (2579,45) C=461 [245(44)] D=793 [642(163)] E=0 [0(0)] F=1254 [887(207)] G=1188 (1464,78) H=NÃO	1421,22 A=190,17 B=265 (2021,45) C=566 [331(41)] D=669 [504(129)] E=615 [525(122)] F=620 [310(48)] G=673 (1425,81) H=SIM (859)	1420,35 A=190,22 B=379 (1700,52) C=671 [331(80)] D=450 [344(66)] E=740 [516(116)] F=381 [159(30)] G=1428 (1425,00) H=NÃO	1411,35 A=192,42 B=306 (1877,35) C=572 [277(66)] D=622 [468(153)] E=675 [496(149)] F=519 [249(70)] G=1354 (1413,23) H=NÃO
7	1503,59 A=199,02 B=374 (1742,27) C=726 [233(32)] D=400 [400(64)] E=1126 [633(96)] F=0 [0(0)] G=811 (1523,20) H=SIM (966)	**1434,06** A=200,28 B=247 (2569,50) C=490 [265(32)] D=763 [634(175)] E=0 [0(0)] F=1253 [899(207)] G=1283 (1442,05) H=NÃO	**1404,15** A=195,58 B=260 (2212,20) C=655 [356(71)] D=585 [474(153)] E=618 [477(151)] F=622 [353(73)] G=1228 (1414,67) H=NÃO	1428,09 A=195,90 B=374 (1742,27) C=471 [220(41)] D=655 [518(105)] E=820 [627(141)] F=306 [111(5)] G=1389 (1436,11) H=NÃO	**1410,38** A=192,33 B=286 (2036,51) C=495 [283(78)] D=719 [569(152)] E=685 [562(174)] F=529 [290(56)] G=1197 (1418,98) H=NÃO
9	1438,51 A=195,13 B=382 (1743,91) C=714 [286(68)] D=404 [404(91)] E=1118 [690(159)] F=0 [0(0)] G=1500 (1457,81) H=NÃO	1449,66 A=201,28 B=247 (2569,50) C=385 [226(34)] D=868 [724(170)] E=0 [0(0)] F=1253 [950(204)] G=1347 (1455,54) H=NÃO	1410,73 A=192,08 B=281 (2061,96) C=486 [286(58)] D=733 [570(133)] E=609 [510(140)] F=610 [346(51)] G=1290 (1423,91) H=NÃO	1402,65 A=196,42 B=382 (1743,91) C=473 [248(79)] D=645 [498(127)] E=778 [592(184)] F=340 [154(22)] G=1470 (1409,80) H=NÃO	1413,33 A=192,52 B=299 (1877,95) C=394 [213(57)] D=807 [596(131)] E=706 [570(134)] F=495 [239(54)] G=1321 (1419,02) H=NÃO
11	**1434,55** A=197,23 B=391 (1689,39) C=703 [261(46)] D=406 [406(103)] E=1109 [667(149)] F=0 [0(0)] G=1486 (1458,65) H=NÃO	1449,73 A=202,43 B=247 (2569,50) C=426 [238(22)] D=827 [699(169)] E=0 [0(0)] F=1253 [937(191)] G=1486 (1465,44) H=NÃO	1425,08 A=189,33 B=260 (2179,46) C=586 [352(45)] D=654 [510(118)] E=620 [515(119)] F=620 [347(44)] G=1155 (1432,99) H=NÃO	**1373,51** A=193,95 B=391 (1689,39) C=495 [241(43)] D=614 [469(128)] E=762 [563(145)] F=347 [147(26)] G=1477 (1378,55) H=NÃO	1418,22 A=192,47 B=287 (2030,74) C=577 [329(55)] D=636 [520(137)] E=740 [608(155)] F=473 [241(37)] G=960 (1424,28) H=NÃO

Legenda

A = Tempo de processamento em minutos, em máquina com processador de 1.7GHz

B = Movimento de ocorrência do primeiro mínimo local (custo da solução no primeiro mínimo local)

C = Número de movimentos de troca de vértices entre rotas [C1(C2)]

C1 = Número de movimentos C que reduziram o valor do movimento anterior do processo

C2 = Número de movimentos C1 que encontraram o melhor valor do processo até aquele instante

D = Número de movimentos de inserção, ou doação de vértice de uma rota para outra [D1(D2)]

D1 = Número de movimentos D que reduziram o valor do movimento anterior do processo

D2 = Número de movimentos D1 que encontraram o melhor valor do processo até aquele instante

E = Número de movimentos que utilizaram o critério de menor custo para o intercâmbio de vértices [E1(E2)]

E1 = Número de movimentos E que reduziram o valor do movimento anterior do processo

E2 = Número de movimentos E1 que encontraram o melhor valor do processo até aquele instante

F = Número de movimentos que utilizaram o critério de maior ângulo para o intercâmbio de vértices [F1(F2)]

F1 = Número de movimentos F que reduziram o valor do movimento anterior do processo

F2 = Número de movimentos F1 que encontraram o melhor valor do processo até aquele instante

G = Movimento de ocorrência da melhor solução (custo da melhor solução – antes da aplicação da 2-Opt)

H = Ocorrência de ciclagem (a partir de que movimento a ciclagem foi percebida)

As tabelas apresentadas até este ponto, correspondem aos dados coletados da primeira fase de testes. Lembrando que nessa fase, os experimentos foram conduzidos com a finalidade de observar algumas características dos critérios e combinações apresentados anteriormente. Nessas tabelas, o número localizado no topo de cada célula corresponde ao valor final da solução, obtido após aplicação de uma heurística de melhoria (2-Opt), em cada rota. Os valores em negrito correspondem ao melhor valor encontrado por cada critério ou combinação.

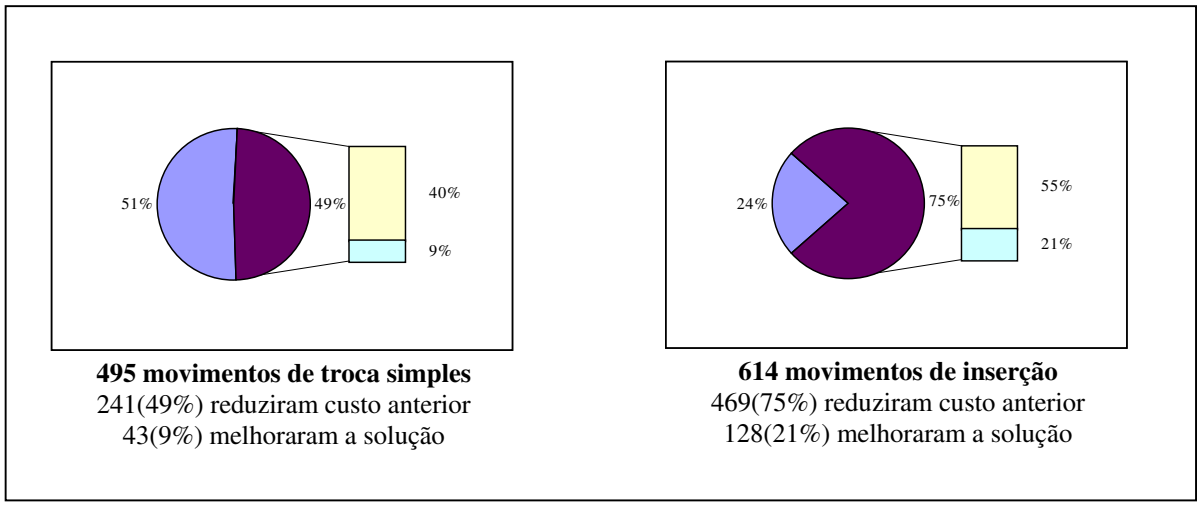
Como exemplo, vamos ler os dados do melhor resultado obtido pela **Combinação 2 (C2)** para o problema 5 (página anterior). Lembrando que para essa fase a solução de partida para cada execução é a mesma adotada na heurística de Clarke e Wright [10], apresentada no capítulo anterior, onde cada vértice é atendido individualmente por uma rota. A idéia é deixar que os critérios e combinações construam as rotas, com seus movimentos de intercâmbio de vértices, desde o início. Então, no início temos muitos movimentos de melhora até chegarmos ao primeiro mínimo local, onde começamos a nossa leitura (fase tabu).

O melhor resultado obtido pela Combinação 2 (C2) para o problema mencionado, ocorreu com o tamanho das listas tabu $T = 11$ movimentos. Nessa célula, podemos observar que o primeiro mínimo local foi encontrado na tentativa 391, que apresentou custo igual a 1689,39. Daí em diante, a Busca Tabu foi melhorando o custo da solução até a tentativa de número 1477, quando apresentou custo igual a 1378,55. Da iteração 1478 à 1500 o processo não obteve soluções melhores. Após a condição de parada (1500 tentativas de melhoria), aplicamos a heurística 2-Opt em cada rota, que melhorou o custo da solução de 1378,55 para **1373,51**.

Após o primeiro mínimo local, o procedimento efetivou 1109 movimentos, sendo 495 movimentos de troca simples de vértices e 614 movimentos de inserção de vértice. Dos 495 movimentos de troca, 241 reduziram o valor obtido pelo movimento anterior, sendo que desses, 43 promoveram melhora no custo da solução. Dos 614 movimentos de inserção, 469 reduziram o valor obtido pelo movimento anterior, sendo que desses, 128 melhoraram o custo da solução.

Podemos também verificar a variação da utilização dos algoritmos na combinação. Dos 1109 movimentos, 762 foram obtidos em função de distâncias, e 347 em função de ângulos. Dos 762 baseados em distâncias, 563 diminuíram o valor obtido pelo movimento anterior, sendo que desses, 145 melhoraram o custo da solução. Dos 347 movimentos baseados em ângulos, 147 reduziram o valor obtido pelo movimento anterior, sendo que desses, 26 melhoraram o custo da solução.

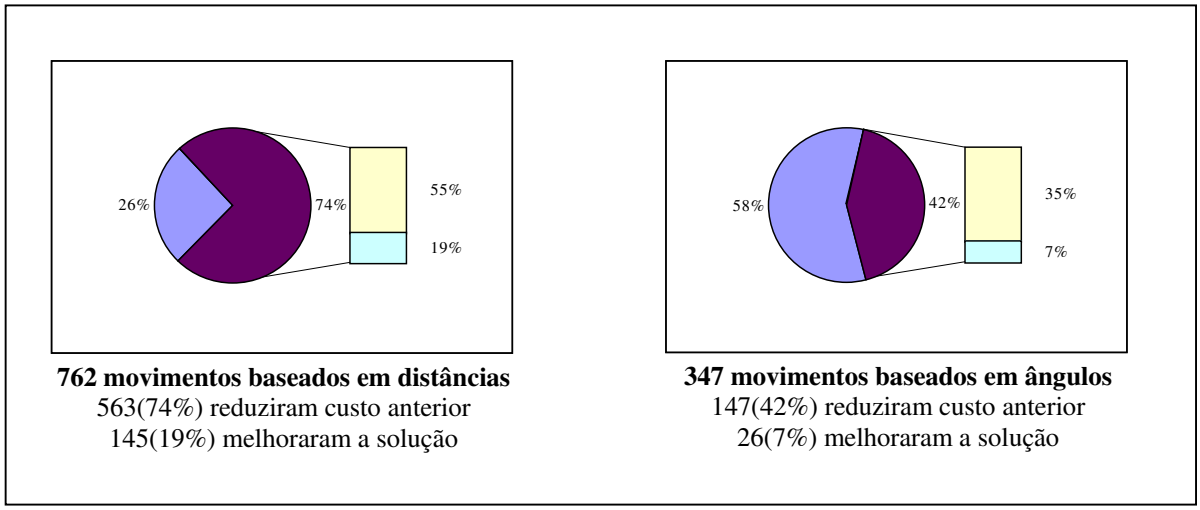
Nas figuras 5.7 e 5.8 podemos observar a representação gráfica desses valores.



495 movimentos de troca simples
241(49%) reduziram custo anterior
43(9%) melhoraram a solução

614 movimentos de inserção
469(75%) reduziram custo anterior
128(21%) melhoraram a solução

Figura 5.7 – Exemplo de gráfico de análise dos movimentos efetivados na Busca Tabu



762 movimentos baseados em distâncias
563(74%) reduziram custo anterior
145(19%) melhoraram a solução

347 movimentos baseados em ângulos
147(42%) reduziram custo anterior
26(7%) melhoraram a solução

Figura 5.8 – Exemplo de gráfico de análise dos critérios utilizados na Busca Tabu

Podemos ainda, encontrar a contribuição da Busca Tabu para a qualidade da solução final, seguindo o exemplo encontrado em Pureza [28]. Para isso, resolvemos a expressão:

$$C_{BT} = ((Z_m - Z_t) * 100) / Z_m$$

onde: Z_m = Custo do primeiro mínimo local

Z_t = Custo da melhor solução obtida na fase tabu

Resolvendo,

$$C_{BT} = ((Z_m - Z_t) * 100) / Z_m = ((1689,39 - 1378,55) * 100) / 1689,39$$

$$C_{BT} = 18,4\ \%$$

Uma outra informação que pode ser obtida da célula em questão, é o tempo total de processamento. Para efetivar os 1500 movimentos de intercâmbio de vértices, a rotina testa todas as possibilidades de troca e inserção de vértices para cada tentativa. Esse procedimento consome muito tempo (193,95 minutos para esse caso), entretanto, nos nossos experimentos estamos mais interessados nas combinações entre os movimentos que utilizam o critério de seleção de vértices baseado em distância, com os movimentos que utilizam o critério de seleção de vértices baseado em ângulos. Por isso, não nos preocupamos com estratégias para diminuição do tempo de execução.

Um fato que observamos, durante a execução desses e de outros testes, é que com a combinação dos critérios, obtivemos um retardo no aparecimento da ciclagem. Em outras palavras, aparentemente a combinação dos critérios deixa o processo mais resistente ao aparecimento do fenômeno da ciclagem.

Após observação dos resultados da primeira fase, escolhemos as combinações **(C1)** e **(C2)** para a realização dos testes da segunda fase. Para a segunda fase, diminuímos o número de tentativas de melhoria da solução de 1500 para 1000, pois nela vamos utilizar soluções de partida de melhor qualidade, obtidas através da heurística de Clarke e Wright [10]. Para cada problema, geramos vinte diferentes soluções de partida, alterando o valor da constante θ de Gaskell [15].

Para cada solução de partida, executamos as rotinas das duas combinações com os mesmos tamanhos de listas utilizados na primeira fase. Apresentamos os resultados em tabelas, onde os valores em negrito correspondem à melhor solução encontrada por cada combinação. Além dos resultados, incluímos também as configurações das rotas, tanto da solução de partida como da solução final, encontrada pela combinação que obteve o melhor valor para o problema. Para a melhor solução encontrada, apresentamos ainda o gráfico do perfil da Busca Tabu, que ilustra o comportamento do processo, e a figura correspondente à composição das rotas.

O objetivo dessa fase é obter valores para compará-los com os de outras estratégias, e verificar se a utilização das combinações dos critérios apresentados pode ser vista como algo interessante ou não. A seguir, apresentamos os dados coletados.

PROBLEMA 1 – 51 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

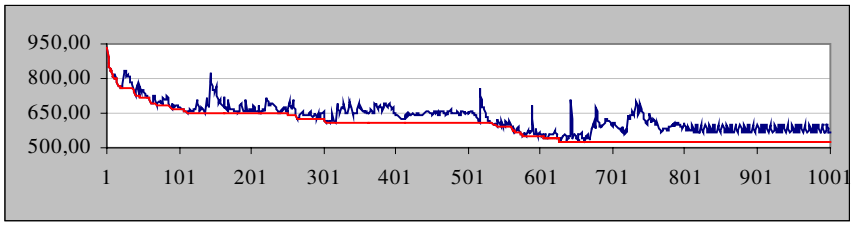
CONST θ	*T* LISTA TABU 3		*T* LISTA TABU 5		*T* LISTA TABU 7		*T* LISTA TABU 9		*T* LISTA TABU 11	
GASKELL	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
0,1	**524,59**	531,27	545,51	551,78	533,72	575,00	586,10	560,81	574,64	577,62
0,2	564,30	535,92	533,64	537,86	562,89	539,36	571,83	561,01	567,40	568,38
0,3	563,73	569,24	555,08	551,15	570,76	544,66	560,92	542,33	599,18	569,60
0,4	553,37	570,00	547,81	553,08	579,53	565,27	537,67	584,43	581,40	575,36
0,5	563,43	543,91	540,60	547,81	556,82	547,61	578,75	555,24	583,83	557,96
0,6	577,29	566,30	531,90	541,41	556,90	553,33	575,55	570,67	586,30	576,65
0,7	569,79	569,89	567,22	569,86	557,90	554,50	582,74	545,52	586,62	569,55
0,8	578,77	556,06	564,55	549,51	543,10	555,63	561,46	561,86	584,85	540,19
1,0	555,66	569,78	564,39	565,87	559,74	560,29	567,10	558,22	586,07	586,07
1,2	541,87	559,95	558,73	546,23	546,36	582,39	551,86	546,51	575,77	561,47
1,3	583,59	553,05	556,87	532,05	550,08	529,27	537,81	583,78	582,10	567,38
1,5	546,30	539,58	554,24	565,10	584,62	561,09	555,23	560,94	581,94	560,29
1,6	533,64	574,58	553,92	541,82	581,17	550,88	562,48	548,21	560,45	561,63
1,7	536,95	539,85	540,24	538,76	549,53	560,09	567,43	553,84	552,41	567,43
1,9	531,42	540,05	556,06	538,76	558,07	537,38	567,43	553,84	562,97	552,04
2,3	549,16	542,42	554,85	535,52	558,07	551,82	567,43	567,43	567,43	559,47
2,7	560,01	541,45	559,83	561,49	564,71	555,16	548,84	547,09	563,87	564,71
2,8	536,36	555,87	550,82	559,78	548,02	560,84	535,02	541,34	551,58	561,04
3,6	546,54	542,84	537,96	539,85	559,94	551,01	544,44	559,18	560,17	561,86
3,7	552,35	538,73	538,62	**527,67**	552,23	564,84	551,10	554,12	561,58	543,00

DADOS DA MELHOR SOLUÇÃO ENCONTRADA

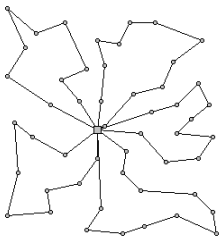
Situação de partida (custo = **936,41**)

Rota													Demanda	Custo
0	5	38	48	14	4	37	49	2	1	32	0		159	164,97
0	28	31	43	3	20	35	36	40	39	33	45	17	0	160	332,89
0	50	29	21	34	30	10	42	19	41	13	24	0		156	176,70
0	22	8	26	7	23	25	44	15	9	16	46	0		158	170,68
0	27	6	18	11	47	12	0							144	91,17

Gráfico custo x movimentos de intercâmbio de vértices



Solução após busca tabu + 2-Opt



Rotas da solução (custo = **524,59**)

Rota												Demanda	Custo
0	8	26	31	28	3	36	35	20	22	1	32	0	149	118,50
0	38	9	30	34	50	16	21	29	2	11	0		159	99,34
0	46	5	49	10	39	33	45	15	44	37	12	0	160	99,26
0	27	48	23	7	43	24	25	14	6	0			152	98,45
0	18	13	41	40	19	42	17	4	47	0			157	109,04

Observação: O melhor valor encontrado para esse problema é 524,61. Gendreau, Hertz e Laporte [27] mencionam que Hadjiconstantinou e Christofides provaram que este resultado é ótimo. Nos nossos testes, trabalhamos com duas casas decimais, por isso obtivemos um resultado com um pequeno desvio. Quando observamos as rotas, percebemos que trata-se da mesma solução reportada pelos autores, que já foi alcançada por várias estratégias.

PROBLEMA 2 – 76 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

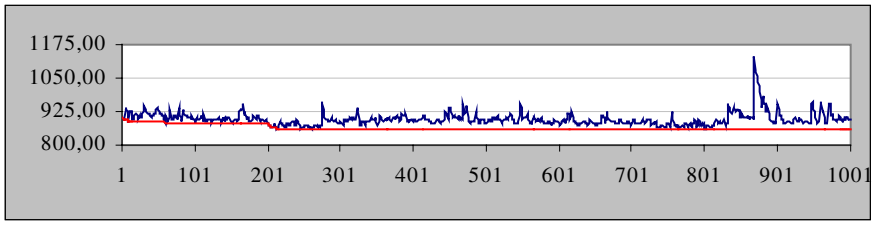
CONST θ GASKELL	*T* LISTA TABU 3		*T* LISTA TABU 5		*T* LISTA TABU 7		*T* LISTA TABU 9		*T* LISTA TABU 11	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
0,1	888,63	911,88	885,93	870,48	901,86	870,87	857,50	878,63	918,06	879,44
0,2	910,33	909,16	876,90	877,39	888,06	882,31	887,83	880,84	897,62	895,13
0,3	905,29	895,81	872,92	866,45	892,97	855,99	884,32	864,17	879,15	891,15
0,4	902,69	894,04	873,32	847,60	875,47	875,35	876,25	883,36	893,75	886,56
0,5	905,45	893,12	873,76	879,77	868,59	869,94	891,81	868,71	891,81	880,64
0,6	860,45	873,92	875,23	873,34	872,70	865,48	899,07	889,89	874,55	882,94
0,7	882,41	879,30	863,45	874,42	863,62	882,82	865,79	864,38	882,82	888,93
0,8	884,76	884,76	863,09	884,76	875,98	861,92	850,18	862,65	884,76	884,76
0,9	868,06	879,78	864,10	863,23	868,35	879,78	879,78	879,78	879,78	879,78
1,0	877,24	869,73	866,51	873,88	873,88	856,94	864,66	862,98	873,88	873,88
1,1	846,65	**846,65**	850,75	**846,65**	848,29	860,97	848,29	860,97	848,29	860,97
1,2	864,33	854,18	873,41	869,77	872,56	872,56	871,17	872,56	866,55	872,56
1,3	854,59	859,10	872,13	852,17	871,82	861,65	871,82	864,64	871,82	871,82
1,4	850,00	853,33	849,34	854,83	856,14	854,89	861,71	861,71	861,71	861,71
1,5	**844,68**	854,04	854,46	866,51	862,00	859,57	860,42	861,50	860,42	857,00
1,6	857,14	881,98	853,85	865,03	867,45	873,82	856,27	875,42	856,27	876,42
1,7	875,80	858,76	867,71	853,75	875,93	851,40	868,31	874,09	864,07	878,00
2,0	865,44	872,65	871,02	859,49	875,22	894,33	875,59	869,83	880,75	875,67
2,2	873,71	867,72	857,17	867,25	871,77	868,38	878,22	871,77	891,17	873,08
2,3	874,24	863,03	872,43	860,47	881,78	882,70	865,20	872,83	865,31	879,25

DADOS DA MELHOR SOLUÇÃO ENCONTRADA

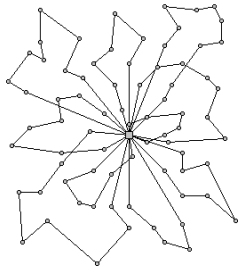
Situação de partida (custo = **900,26**)

Rota												Demanda	Custo
0	47	36	69	71	60	70	20	37	5	29	0	136	103,47
0	11	66	65	38	0							107	75,64
0	59	14	53	7	67	0						122	79,47
0	8	35	19	54	13	57	15	48	30	0		133	92,99
0	10	31	25	55	18	50	32	0				135	116,97
0	34	46	52	27	45	4	0					133	41,90
0	17	51	68	75	0							62	32,75
0	44	3	24	49	16	63	33	6	0			136	84,03
0	23	56	43	41	42	64	22	61	0			131	126,62
0	58	72	39	9	40	12	26	0				134	62,96
0	2	1	73	62	28	21	74	0				135	83,46

Gráfico custo x movimentos de intercâmbio de vértices



Solução após busca tabu + 2-Opt



Rotas da Solução (custo = **844,68**)

Rota												Demanda	Custo
0	47	36	69	71	60	70	20	37	29	45	0	136	103,46
0	58	38	65	66	11	0						128	77,93
0	67	34	52	27	4	75	0					135	37,81
0	46	54	13	57	15	5	48	30	0			140	84,32
0	72	10	31	25	55	18	50	44	51	0		137	122,67
0	8	35	19	59	14	53	7	0				133	86,35
0	17	3	24	49	16	63	33	6	0			139	80,51
0	23	56	41	43	42	64	22	1	73	0		140	115,24
0	26	12	39	9	32	40	0					140	58,20
0	68	2	62	28	61	21	74	0				136	78,19

PROBLEMA 3 – 101 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

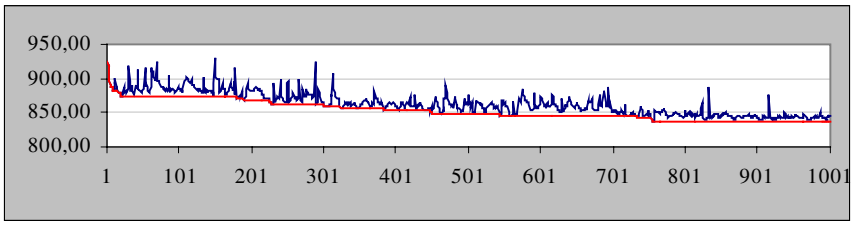
CONST θ GASKELL	*T* LISTA TABU 3		*T* LISTA TABU 5		*T* LISTA TABU 7		*T* LISTA TABU 9		*T* LISTA TABU 11	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
0,1	896,98	872,28	844,41	862,47	889,87	872,33	868,25	869,71	850,60	848,93
0,2	911,18	872,95	923,56	858,35	875,99	844,40	849,93	877,75	866,25	864,98
0,3	912,18	870,03	906,67	887,58	854,83	856,72	877,15	859,27	888,26	859,87
0,4	888,62	851,75	887,31	865,78	879,02	870,16	889,75	852,52	847,88	873,75
0,5	859,46	848,51	840,01	856,99	848,23	853,54	851,87	861,66	864,63	855,56
0,6	873,07	851,01	851,26	854,52	851,20	856,20	853,39	845,24	850,76	853,93
0,8	846,99	881,39	843,94	852,71	855,51	841,19	860,60	846,34	852,98	845,96
0,9	848,79	881,39	870,88	846,27	849,18	850,75	859,25	854,88	861,90	851,65
1,0	855,52	856,41	843,95	854,90	859,14	850,49	867,53	844,52	851,81	863,93
1,1	860,65	852,55	861,71	867,97	850,52	845,81	853,85	853,73	858,23	852,22
1,2	853,21	847,04	853,32	850,07	853,64	849,65	844,02	858,68	847,53	858,68
1,3	854,34	863,22	852,53	845,73	850,81	863,22	845,45	842,72	854,95	855,17
1,4	852,25	851,80	848,56	855,13	849,55	848,84	850,07	850,28	850,01	849,08
1,5	848,89	846,19	847,73	846,29	856,35	855,07	850,65	849,76	858,12	851,79
1,8	864,65	865,93	855,23	**833,16**	862,83	864,79	851,83	844,08	854,02	854,23
1,9	849,92	845,98	866,53	849,21	855,60	843,86	863,43	851,23	859,05	856,51
2,0	869,37	857,78	860,18	843,90	846,54	858,54	850,93	857,64	854,49	854,90
2,1	854,35	842,08	**838,76**	838,68	849,32	842,02	855,91	848,41	849,13	861,08
2,2	859,92	856,92	860,94	849,14	852,26	852,52	851,16	849,12	850,74	865,18
2,4	853,23	862,12	855,70	842,79	850,44	847,25	852,57	840,91	853,81	860,11

DADOS DA MELHOR SOLUÇÃO ENCONTRADA

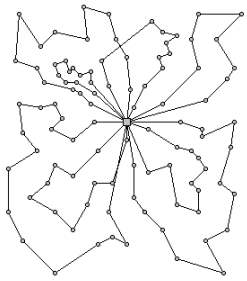
Situação de partida (custo = **923,87**)

Rota																		Demanda	Custo
0	95	97	92	98	37	100	91	85	93	59	99	96	6	89	0			199	63,40
0	94	13	58	40	21	73	53	0										111	57,06
0	76	50	33	79	3	77	80	68	29	24	54	12	26	0				195	102,74
0	51	9	81	78	34	35	65	71	66	20	30	70	0					172	134,65
0	32	90	63	11	64	49	36	47	19	48	82	7	0					193	141,62
0	5	84	17	45	46	8	83	60	18	52	88	62	10	31	27	28	0	199	140,30
0	72	74	41	22	75	56	23	67	39	25	55	4	0					192	121,08
0	61	16	86	44	38	14	43	15	57	42	87	2	1	69	0			197	163,02

Gráfico custo x movimentos de intercâmbio de vértices



Solução após busca tabu + 2-Opt



Rotas da Solução (custo = **833,16**)

Rota																		Demanda	Custo
0	95	97	92	37	98	100	91	85	93	59	94	0						188	58,45
0	13	87	41	22	75	74	72	73	21	40	53	0						166	75,96
0	1	50	51	9	81	33	79	3	77	76	28	0						165	79,85
0	12	80	68	24	29	78	34	35	71	65	66	20	30	70	69	0		199	138,67
0	27	31	88	62	11	19	47	48	82	7	52	0						193	90,99
0	89	18	83	60	5	84	17	45	8	46	36	49	64	63	90	32	10	0	198	159,15
0	54	55	25	39	67	23	56	4	26	0									153	107,08
0	6	96	99	61	16	86	38	44	14	42	43	15	57	2	58	0			196	123,01

PROBLEMA 12 – 101 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

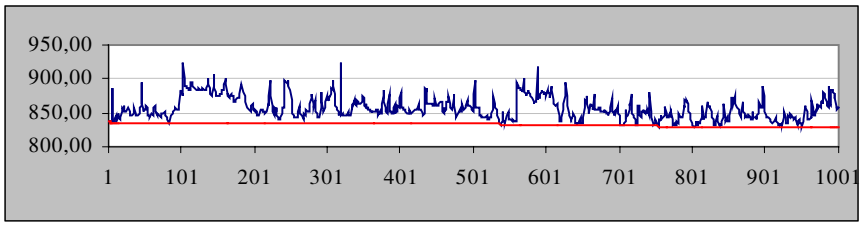
CONST θ GASKELL	*T* LISTA TABU 3		*T* LISTA TABU 5		*T* LISTA TABU 7		*T* LISTA TABU 9		*T* LISTA TABU 11	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
0,1	968,61	944,78	939,74	915,70	912,52	915,19	893,19	914,98	884,76	891,45
0,2	844,12	831,86	903,34	853,13	899,13	844,11	885,39	842,01	861,20	854,65
0,3	842,80	841,80	863,04	841,48	872,76	858,19	835,08	844,62	867,62	848,19
0,4	834,21	835,59	834,38	831,84	833,60	835,90	837,15	825,24	834,15	835,87
0,5	829,01	829,75	832,81	831,13	829,76	823,15	838,07	830,21	829,85	825,27
0,6	827,12	827,12	827,98	826,71	826,80	826,99	826,80	826,71	826,80	826,71
0,7	828,24	828,15	826,71	820,70	826,71	826,71	826,71	826,71	826,71	826,71
0,8	828,24	828,15	828,46	828,46	828,46	828,46	828,46	828,46	828,46	828,46
0,9	828,65	828,56	828,87	827,12	828,87	828,65	828,87	828,65	828,87	828,65
1,0	**820,29**	820,70	820,70	820,70	820,70	820,70	820,70	820,70	820,70	820,70
1,1	820,70	820,02	820,70	820,70	820,70	820,70	820,70	820,70	820,70	820,70
1,2	826,25	826,25	826,25	822,45	826,25	823,52	826,25	823,52	826,25	823,52
1,3	827,29	820,70	826,25	820,70	826,25	820,70	826,25	820,70	820,70	820,70
1,4	827,82	822,27	827,82	822,27	827,82	830,76	827,82	823,34	827,82	827,82
1,5	827,82	822,27	827,93	823,20	827,82	830,76	827,82	**819,61**	827,82	827,82
2,0	827,82	820,70	827,93	822,27	827,82	830,06	827,82	823,34	827,82	827,82
2,2	829,15	822,15	829,38	823,72	829,27	831,67	829,27	824,79	829,27	822,15
2,3	834,33	831,58	829,27	825,13	829,27	826,95	829,27	829,38	829,27	832,62
2,4	827,58	830,14	827,14	825,13	828,88	820,29	827,14	826,20	827,14	827,41
2,8	834,00	829,49	843,65	824,17	836,56	826,39	826,95	826,95	826,95	826,95

DADOS DA MELHOR SOLUÇÃO ENCONTRADA

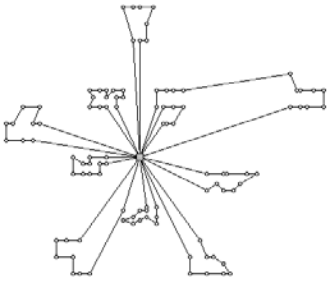
Situação de partida (custo = **838,64**)

Rota															Demanda	Custo
0	99	100	97	93	92	94	95	96	98	0					190	95,96
0	91	89	88	85	84	83	82	86	87	90	0				170	76,86
0	80	79	77	73	70	71	76	78	81	0					150	127,30
0	63	65	67	69	66	62	74	72	61	64	68	0			200	64,15
0	60	58	56	53	54	55	57	59	0						200	103,73
0	41	40	42	46	44	45	48	51	50	52	49	47	43	0	160	66,29
0	34	36	39	38	37	35	31	33	32	0					200	97,23
0	23	26	28	30	29	27	25	24	22	21	20	0			170	51,22
0	12	14	16	15	19	18	17	13	0						190	95,89
0	7	3	5	75	1	2	4	6	9	8	11	10	0		180	60,01

Gráfico custo x movimentos de intercâmbio de vértices



Solução após busca tabu + 2-Opt



Rotas da Solução (custo = **819,61**)

Rota															Demanda	Custo
0	99	100	97	93	92	94	95	96	98	0					190	95,96
0	75	1	2	4	6	9	11	8	7	3	5	0			170	56,18
0	69	68	64	61	72	80	79	77	73	70	71	76	78	81	0	200	137,02
0	66	62	74	63	65	67	0									150	43,59
0	59	60	58	56	53	54	55	57	0							200	101,89
0	43	42	41	40	44	46	45	48	51	50	52	49	47	0		160	64,81
0	34	36	39	38	37	35	31	33	32	0						200	97,23
0	20	24	25	27	29	30	28	26	23	22	21	0				170	50,81
0	10	12	14	16	15	19	18	17	13	0						200	96,04
0	90	87	86	83	82	84	85	88	89	91	0					170	76,08

Observação: O melhor valor encontrado para esse problema é 819,56. Entretanto, quando comparamos as rotas acima com as reportadas por Gendreau, Hertz e Laporte [27], notamos que trata-se da mesma solução. Isso ocorreu porque estamos trabalhando com duas casas decimais, enquanto que, os autores trabalharam com quatro.

PROBLEMA 11 – 121 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

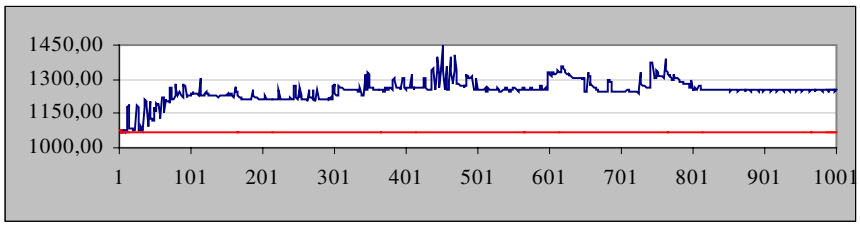
CONST θ GASKELL	*T* LISTA TABU 3		*T* LISTA TABU 5		*T* LISTA TABU 7		*T* LISTA TABU 9		*T* LISTA TABU 11	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
0,1	1101,08	1065,78	1148,47	1070,22	1087,55	1069,52	1124,32	1069,06	1138,93	1130,51
0,2	1063,64	1060,92	1074,42	1068,53	1129,01	1070,73	1129,12	1072,24	1097,57	1070,21
0,3	1114,52	1114,52	1088,48	1111,85	1101,13	1114,52	1114,52	1114,52	1114,52	1104,57
0,4	1068,59	1097,96	1085,94	1100,03	1099,26	1093,81	1096,96	1098,46	1096,96	1095,14
0,5	1110,12	1100,24	1102,49	1077,40	1100,00	1098,17	1100,00	1088,29	1105,47	1091,43
0,6	1087,47	1086,19	1087,47	1067,72	1075,65	1067,91	1075,65	1067,91	1071,66	1070,93
0,7	1078,03	1062,32	1078,03	1078,03	1078,03	1078,03	1062,59	1078,03	1062,59	1078,03
0,8	1070,53	1068,61	1066,08	1072,16	1069,82	1071,14	1069,82	1059,96	1069,82	1068,96
0,9	**1048,12**	**1048,12**	**1048,12**	**1048,12**	**1048,12**	**1048,12**	**1048,12**	**1048,12**	**1048,12**	**1048,12**
1,0	1049,33	1049,33	1049,33	1049,33	1049,33	1049,33	1049,33	1049,33	1049,33	1049,33
1,1	1061,96	1061,96	1061,96	1061,96	1061,96	1061,96	1061,96	1061,96	1061,96	1061,96
1,2	1061,83	1062,89	1063,56	1062,89	1063,56	1062,89	1062,89	1062,89	1061,07	1062,89
1,3	1058,64	1054,27	1054,41	1056,15	1054,24	1056,15	1056,15	1056,15	1055,92	1056,15
1,4	1063,24	1054,57	1054,84	1057,25	1061,45	1055,04	1061,19	1055,04	1059,85	1055,71
1,5	1061,61	1062,47	1060,59	1062,47	1061,45	1064,93	1065,06	1064,93	1064,93	1064,93
1,6	1067,01	1052,17	1050,41	1052,17	1056,37	1056,43	1056,37	1056,43	1056,43	1056,43
1,9	1067,51	1056,56	1062,28	1056,56	1061,67	1056,56	1056,56	1056,56	1056,56	1056,56
2,0	1067,51	1056,56	1062,28	1056,56	1061,67	1056,56	1056,56	1056,56	1056,56	1056,56
2,3	1055,11	1055,16	1055,96	1062,28	1057,70	1062,28	1054,63	1062,28	1063,61	1062,28
2,4	1058,66	1064,42	1057,03	1064,98	1060,40	1064,98	1057,33	1064,98	1066,31	1064,98

DADOS DA MELHOR SOLUÇÃO ENCONTRADA (com C1 e *T* LISTA TABU = 3)

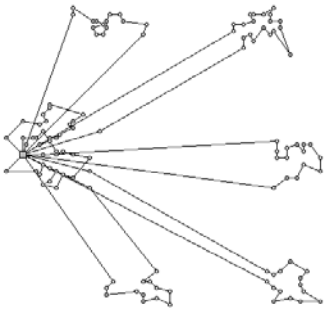
Situação de partida (custo = **1075,48**)

Rota																			Demanda	Custo
0	87	92	89	90	114	108	18	113	83	117	84	112	85	81	119	0			187	67,61
0	82	111	86	95	96	93	94	97	115	116	100	103	104	107	99	101	102	106		
	105	120	0																194	80,94
0	67	69	70	71	74	72	75	78	80	79	77	76	73	68	98	0			200	145,67
0	53	55	56	58	60	63	66	64	62	61	65	59	57	54	52	110	0		199	214,48
0	118	37	38	39	42	41	44	47	46	49	50	51	48	45	43	40	91	0	199	205,06
0	17	16	19	25	28	32	35	36	34	33	30	31	27	24	22	23	26	29		
	20	21	109	0															197	220,44
0	2	1	7	6	5	3	4	9	10	11	15	14	13	12	8	88	0		199	141,28

Gráfico custo x movimentos de intercâmbio de vértices



Solução após busca tabu + 2-Opt



Rotas da Solução (custo = **1048,12**)

Rota																			Demanda	Custo
0	87	92	89	91	90	114	18	118	108	83	113	117	84	112	81	119	0		187	62,29
0	82	111	86	85	93	96	94	97	115	116	100	103	107	104	99	101	102	106		
	105	120	0																193	81,07
0	67	69	70	71	74	72	75	78	80	79	77	76	73	68	98	0			200	145,67
0	53	55	58	56	60	63	66	64	62	61	65	59	57	54	52	110	0		199	214,21
0	40	43	45	48	51	50	49	46	47	44	41	42	39	38	37	95	0		200	199,62
0	17	16	19	25	22	24	27	33	30	31	34	36	29	35	32	28	26	23		
	20	21	109	0															197	207,95
0	8	12	13	14	15	11	9	7	6	5	10	4	3	1	2	88	0		199	137,31

Observação: Neste problema, podemos notar que a solução encontrada foi obtida no começo do processo, nas tentativas iniciais. Gendreau, Hertz e Laporte [27], relatam um comportamento semelhante obtido em seus testes.

PROBLEMA 4 – 151 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

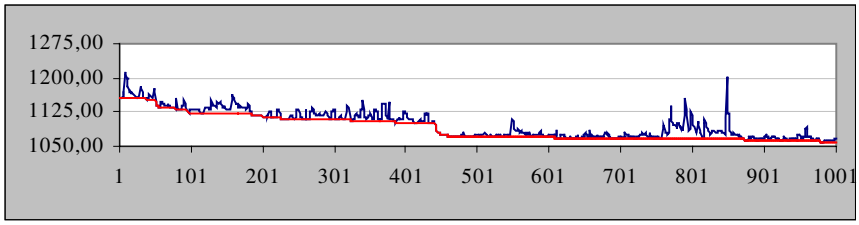
CONST θ GASKELL	*T* LISTA TABU 3		*T* LISTA TABU 5		*T* LISTA TABU 7		*T* LISTA TABU 9		*T* LISTA TABU 11	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
0,1	1155,85	1152,20	1182,23	1199,46	1162,42	1168,20	1135,49	1129,77	1115,24	1151,55
0,2	1144,41	1097,79	1120,88	1156,01	1084,69	1131,78	1084,25	1105,99	1108,69	1093,82
0,3	1156,84	1134,76	1090,95	1077,26	1081,36	1102,13	1077,61	1084,01	1080,61	1064,29
0,4	1094,82	1093,82	1090,20	1093,04	1086,00	1095,42	1095,41	1092,03	1081,81	1087,45
0,5	1090,65	1090,88	1090,73	1095,36	1093,28	1092,16	1084,32	1079,50	1094,76	1087,97
0,6	1115,88	1081,18	1083,15	1070,21	1086,95	1097,65	1097,10	1087,31	1095,19	1086,87
0,7	1086,33	1114,61	1078,99	1081,66	1090,35	1092,89	1081,55	1095,16	1082,33	1091,12
0,8	1122,90	1114,83	1092,58	1080,26	1120,84	1075,32	1089,39	1099,70	1102,19	1080,66
0,9	1143,53	1111,34	1090,78	1115,81	1096,04	1082,37	1092,67	1077,63	1106,23	1077,86
1,0	1091,00	1104,72	**1058,67**	1112,58	1096,16	1088,76	1078,81	1087,04	1084,66	1068,35
1,1	1104,69	1083,11	1088,33	1075,55	1082,64	1078,23	1078,01	1095,20	1087,35	1076,50
1,2	1085,00	1133,08	1067,89	1070,67	1072,59	1075,63	1076,13	1072,36	1082,99	1081,23
1,3	1078,98	1088,46	1082,68	1067,78	1060,83	1069,84	1073,40	1064,83	1085,29	1085,50
1,4	1081,13	1091,87	1084,34	1072,89	1072,59	1083,36	1075,92	1079,29	1081,96	1073,36
1,5	1099,09	1105,24	1083,64	1095,63	1077,93	1079,68	1070,45	1074,15	1092,90	1079,29
1,6	1076,45	1075,50	1082,78	1082,27	1078,21	1067,60	1074,11	1077,38	1081,64	**1059,88**
1,7	1085,63	1106,50	1082,55	1081,22	1073,55	1099,34	1081,79	1075,70	1076,40	1083,61
1,8	1098,11	1073,97	1079,55	1079,14	1074,24	1082,42	1077,64	1086,18	1089,32	1088,60
1,9	1093,06	1080,70	1087,49	1085,94	1083,22	1082,06	1062,04	1067,68	1084,84	1075,90
2,0	1100,72	1093,14	1079,37	1070,81	1067,43	1071,57	1092,53	1089,84	1084,21	1074,12

DADOS DA MELHOR SOLUÇÃO ENCONTRADA

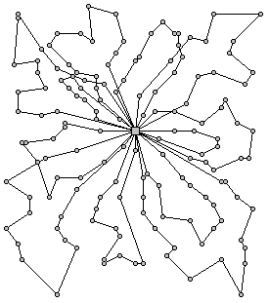
Situação de partida (custo = **1157,59**)

Rota	Demanda	Custo
0 114 8 46 124 47 36 143 49 64 11 107 19 123 7 146 0	197	128,70
0 15 43 14 119 44 38 140 86 141 16 113 17 84 0	198	129,59
0 120 9 103 20 128 66 71 65 136 35 135 34 78 129 0	197	137,04
0 127 88 148 62 10 108 126 63 90 32 131 30 122 70 101 69 0	200	92,99
0 5 118 60 83 125 45 48 82 106 0	192	83,02
0 51 81 33 79 121 29 24 134 130 54 4 110 149 0	195	108,65
0 105 26 28 111 27 0	89	36,32
0 13 94 95 59 104 99 96 6 147 112 0	184	44,53
0 117 97 92 37 98 93 61 85 91 100 142 42 144 87 137 0	198	86,07
0 50 102 76 116 77 3 68 150 80 109 12 138 0	195	66,60
0 89 18 52 31 132 1 2 115 73 74 72 21 40 58 53 0	196	122,63
0 55 25 139 39 67 23 56 75 133 22 41 145 57 0	194	121,45

Gráfico custo x movimentos de intercâmbio de vértices



Solução após busca tabu + 2-Opt



Rotas da Solução (custo = **1058,67**)

Rota	Demanda	Custo
0 18 114 46 124 47 36 143 49 64 11 107 19 123 7 146 0	200	128,71
0 117 92 91 44 38 140 86 141 16 61 113 17 84 5 0	199	113,84
0 69 101 70 30 20 128 66 71 65 136 35 135 34 78 129 79 0	199	123,68
0 127 88 148 62 10 108 126 63 90 32 131 31 27 0	196	89,30
0 89 118 60 83 125 45 8 48 82 106 52 0	199	77,33
0 76 77 3 121 29 24 134 130 54 109 26 105 0	172	78,80
0 132 1 122 51 103 9 120 81 33 102 50 111 0	199	77,47
0 53 40 21 72 74 133 22 41 145 115 2 137 13 0	172	67,10
0 94 95 97 37 100 119 14 142 42 43 15 57 144 87 0	195	95,64
0 28 116 68 150 80 12 138 0	117	45,81
0 147 6 96 104 99 93 85 98 59 112 0	187	49,16
0 149 110 4 55 25 139 39 67 23 56 75 73 58 0	200	111,83

PROBLEMA 5 – 200 VÉRTICES – (Christofides-Mingozzi-Toth-1979)

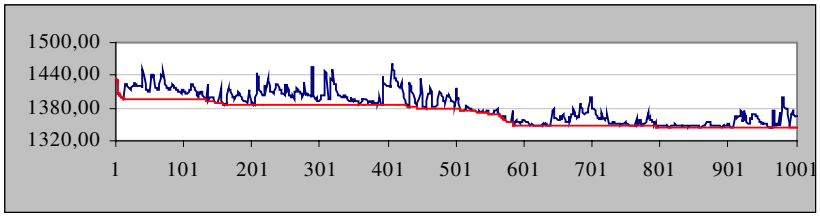
CONST θ GASKELL	*T* LISTA TABU 3		*T* LISTA TABU 5		*T* LISTA TABU 7		*T* LISTA TABU 9		*T* LISTA TABU 11	
	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
0,1	1445,40	1475,75	1502,08	1446,19	1458,58	1431,49	1465,91	1417,30	1508,72	1483,79
0,2	1409,13	1453,93	1435,16	1397,96	1409,22	1460,38	1468,04	1440,43	1420,61	1416,96
0,3	1561,41	1552,63	1548,84	1521,01	1557,34	1520,26	1548,28	1515,07	1562,37	1536,78
0,4	1436,04	1446,73	1437,47	1412,07	1412,94	1421,50	1423,66	1413,25	1421,88	1428,62
0,5	1431,44	1455,29	1384,52	1398,64	1397,46	1406,90	1399,26	1374,73	1396,27	1402,45
0,6	1401,17	1419,78	1397,17	1408,06	1381,99	1386,70	1390,11	1362,26	1373,66	1394,16
0,7	1404,16	1402,00	1379,80	1390,41	1372,40	1367,64	1378,94	1392,92	1389,06	1399,18
0,8	1390,42	1375,91	1386,19	1380,64	1374,11	1389,01	1385,71	1363,85	1371,07	1353,88
0,9	1386,95	1420,99	1385,29	1381,37	1370,97	1377,72	1391,98	1385,24	1401,63	1390,09
1,0	1402,06	1386,00	1382,76	1380,53	1372,46	1402,07	1380,67	1395,20	1384,72	1376,85
1,1	1364,58	1376,03	1361,81	1356,70	1362,50	1376,22	1350,91	1356,83	1366,62	1361,79
1,2	1357,15	1362,27	1353,43	1353,57	1356,95	1353,97	1362,35	1361,56	1357,55	1366,29
1,3	1365,10	1354,34	1354,60	1361,40	1364,89	1353,29	1360,46	1355,25	1357,48	1353,12
1,4	1355,46	1349,27	1350,04	1345,96	1352,36	1350,19	1349,75	1346,59	1360,26	1350,53
1,5	1352,30	1355,53	1363,47	1349,69	1352,90	1349,20	1348,47	1355,50	1360,81	1351,60
1,6	1389,51	1376,84	1374,24	1369,87	1395,89	1353,34	1394,54	1359,35	1371,69	1363,36
1,7	1393,48	1349,02	1370,43	1354,25	1352,80	1367,27	1371,39	1365,16	1348,70	1346,82
1,8	1359,06	1368,95	1357,19	1355,20	**1343,68**	**1344,39**	1359,89	1353,39	1364,68	1370,85
1,9	1360,26	1369,64	1370,73	1352,74	1386,12	1350,61	1364,34	1351,53	1370,67	1373,74
2,0	1384,11	1367,26	1381,18	1353,92	1359,58	1354,71	1346,89	1356,09	1370,31	1363,81

DADOS DA MELHOR SOLUÇÃO ENCONTRADA

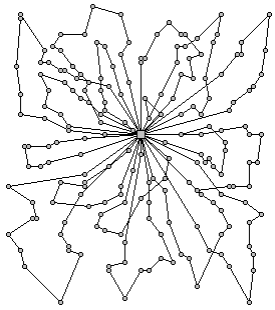
Situação de partida (custo = **1435,21**)

Rota	Demanda	Custo
0 53 105 180 40 152 58 112 156 0	158	32,79
0 46 124 168 47 36 143 49 64 11 175 107 19 123 88 0	193	131,21
0 171 74 197 72 73 21 198 110 179 195 149 26 0	194	73,07
0 9 120 81 33 185 79 129 121 29 24 163 134 12 0	197	96,51
0 155 4 139 187 39 67 170 25 55 165 130 54 0	199	95,50
0 138 154 28 111 50 102 157 0	138	51,85
0 109 177 150 80 68 158 3 77 116 196 76 184 0	197	58,05
0 169 78 34 164 135 35 136 65 71 161 103 51 0	192	113,71
0 27 167 132 69 101 162 31 190 127 52 146 0	166	49,10
0 66 188 20 128 160 131 32 181 63 126 90 108 0	200	115,78
0 122 30 70 189 10 159 62 148 182 7 194 106 0	196	78,18
0 16 191 141 44 119 14 38 140 86 113 17 173 0	197	111,38
0 117 95 97 92 151 59 93 104 99 96 94 183 6 0	191	51,72
0 192 91 193 100 37 98 85 61 84 5 118 60 0	196	82,94
0 153 48 82 114 8 174 45 125 199 83 18 0	192	80,15
0 172 42 142 43 15 57 144 87 137 13 147 166 89 0	185	96,31
0 23 186 56 75 133 22 41 145 115 178 2 1 176 0	195	116,96

Gráfico custo x movimentos de intercâmbio de vértices



Solução após busca tabu + 2-Opt



Rotas da Solução (custo = **1343,68**)

Rota	Demanda	Custo
0 13 87 172 42 97 95 94 183 112 156 0	200	54,21
0 7 123 19 107 175 11 64 49 143 36 47 168 124 46 153 0	200	131,03
0 40 73 171 74 75 23 186 56 110 179 195 149 26 0	193	81,97
0 102 79 129 169 121 29 24 163 134 177 12 154 0	198	79,35
0 155 4 139 187 39 67 170 25 55 165 130 54 0	199	95,50
0 50 157 33 81 120 9 66 188 20 122 1 132 0	197	91,03
0 138 109 150 80 68 158 3 77 116 196 184 28 0	198	55,55
0 76 185 78 34 164 135 35 136 65 71 161 103 51 0	200	111,59
0 127 190 88 182 48 82 194 106 52 146 0	183	64,66
0 27 108 90 126 63 181 32 131 160 128 30 70 101 69 0	197	96,79
0 111 176 162 189 10 159 62 148 31 167 0	197	66,13
0 117 98 193 91 191 141 44 38 140 86 113 17 84 60 89 0	200	102,83
0 53 58 152 105 0	79	22,30
0 6 96 104 99 93 85 16 61 173 5 118 0	200	64,42
0 18 114 8 174 45 125 199 83 166 0	148	63,75
0 147 59 92 151 37 100 192 119 14 142 43 15 57 144 137 0	198	90,20
0 180 21 198 72 197 133 22 41 145 115 178 2 0	199	72,37

A seguir, apresentamos uma tabela contendo o resultado de algumas estratégias, juntamente com os melhores resultados obtidos nos experimentos.

#	AUTORES (ANO) ESTRATÉGIA	P1-51	P2-76	P3-101	P12-101	P11-121	P4-151	P5-200
01	Clarke e Wright (1964) Heurística das Economias - *Savings*	585	900	886	831	1079	1204	1540
02	Gillett e Miller (1974) Heurística da Varredura - *Sweep*	532	874	851	937	1266	1079	1389
03	Mole e Jameson (1976) *Generalized Savings Algorithm*	575	910	882	879	1100	1259	1545
04	Pureza e França (1991) Busca Tabu	536	842	851	826	1049	1081	
05	Osman (1993) *Simulated Annealing*	528	838,62	829,18	826	1176	1058	1378
06	Osman (1993) Busca Tabu	**524,61**	844	835	819,59	**1042,11**	1044,35	1334,55
07	Taillard (1993) Busca Tabu	**524,61**	**835,26**	**826,14**	**819,56**	**1042,11**	**1028,42**	1298,79
08	Gendreau, Hertz, e Laporte (1994) Busca Tabu – Taburoute *standard*	**524,61**	835,77	829,45	**819,56**	1073,47	1036,16	1322,65
09	Gendreau, Hertz, e Laporte (1994) Busca Tabu – Taburoute *best*	**524,61**	835,32	**826,14**	**819,56**	**1042,11**	1031,07	1311,35
10	Rochat e Taillard (1995) Busca Tabu							**1291,45**
11	Xu e Kelly (1996) Busca Tabu	**524,61**	**835,26**	**826,14**	**819,56**	**1042,11**	1029,56	1298,58
12	Rego e Roucairol (1996) Busca Tabu	**524,61**	835,32	827,53	**819,56**	**1042,11**	1044,35	1334,55
13	Toth e Vigo (1998) Busca Tabu	**524,61**	838,60	828,56	**819,56**	1042,87	1033,21	1318,25
C1	Busca Tabu Combinação 1	**524,59***	844,68	838,76	820,29	1048,12	1058,67	1343,68
C2	Busca Tabu Combinação 2	527,67	846,65	833,16	**819,61***	1048,12	1059,88	1344,39

Os resultados em negrito correspondem aos melhores valores reportados até o momento. Os resultados marcados com (*), apesar de diferenças nas casas decimais, correspondem aos

melhores valores reportados, pois a composição das rotas encontradas é a mesma relatada por Gendreau, Hertz e Laporte [27].

Observando esses resultados, podemos verificar que foram encontrados valores intermediários, apesar da simplicidade da estratégia empregada. Isso nos faz crer que o uso de critérios de escolha de vértices baseados em distâncias, combinado com critérios baseados em ângulos (na Busca Tabu), pode ser uma estratégia interessante para o PRV.

Entretanto, após todos os testes, ficamos intrigados com algumas questões: Será que as combinações adotadas (C1 e C2) são igualmente interessantes? Será que a maneira como elas foram idealizadas exerce papel decisivo na qualidade das soluções obtidas? Na tentativa de responder a essas questões, realizamos novos testes, apresentados na próxima seção.

5.4 Verificando as Combinações de Critérios

Para verificar a saúde das combinações propostas no item 5.1, realizamos novos testes com o cenário da primeira fase. Só que agora, combinamos os critérios de seleção de vértices (baseado em distâncias e baseado em ângulos), de maneira aleatória, ou seja, o processo alterna entre um e outro critério de maneira aleatória (*). Na tabela seguinte, podemos comparar os resultados desses testes com os melhores resultados obtidos para cada problema por uma das combinações propostas.

T	P1-51	P2-76	P3-101	P12-101	P11-121	P4-151	P5-200
3	C1 542,73 * 550,47	C2 853,78 * 858,45	C2 844,19 * 873,03	C3 857,97 * 876,89	C2 1237,55 * 1270,53	C1 1068,10 * 1129,40	C3 1461,72 * 1492,23
5	C1 544,79 * 539,65	C2 856,81 * 861,82	C1 848,00 * 883,58	C3 876,37 * 853,99	C2 1219,40 * 1299,30	C3 1068,14 * 1076,24	C3 1411,35 * 1462,80
7	C2 535,19 * 552,96	C1 861,10 * 858,82	C3 844,76 * 841,11	C3 878,23 * 884,87	C3 1267,76 * 1193,28	C3 1080,68 * 1108,65	C1 1404,15 * 1405,36
9	C1 531,83 * 562,27	C3 867,42 * 879,52	C2 838,82 * 876,44	C2 872,43 * 884,79	C3 1259,56 * 1220,82	C1 1091,28 * 1093,68	C2 1402,65 * 1386,76
11	C2 557,20 * 544,75	C1 853,84 * 865,70	C1 838,16 * 847,85	C1 857,41 * 883,58	C2 1269,17 * 1308,61	C2 1073,31 * 1081,76	C2 1373,51 * 1366,68

Observando os números, notamos que em 9 das 35 comparações (26%), os resultados obtidos pela estratégia aleatória é superior ao resultados das combinações propostas, e em algumas outras comparações, os valores são muito próximos. Isso nos leva a pensar que podemos melhorar as combinações propostas nesse trabalho, ou ainda, que o simples fato de combinar os elementos “distâncias e ângulos” é mais interessante do que as combinações elaboradas para os nossos experimentos.

6 Conclusão

Neste trabalho, estudamos algumas abordagens para problemas de roteamento. Apresentamos estratégias clássicas propostas para o Problema do Caixeiro Viajante e para o Problema de Roteamento de Veículos, e vimos que algumas das idéias utilizadas para um desses problemas podem ser aproveitadas ou adaptadas para o outro problema. Para auxiliar no entendimento das estratégias, desenvolvemos um programa, com o qual realizamos várias simulações que foram aproveitadas para ilustrar o trabalho.

Durante o estudo, algumas observações foram sendo feitas, e a partir delas, foram realizados alguns experimentos que combinam dois diferentes critérios de seleção de vértices, para realização de movimentos de intercâmbio entre rotas. Um dos critérios utiliza a distância entre vértices e vizinhanças para eleger os candidatos ao movimento. O outro critério utiliza o ângulo formado pelas arestas que ligam cada candidato aos seus vizinhos, para eleição dos vértices a serem utilizados no movimento. Os resultados desses experimentos, foram comparados com os de outras estratégias, sendo que em dois dos problemas testados, os valores obtidos correspondem ao melhor valor já reportado para os problemas.

Diante dos resultados, concluímos que combinar elementos como distâncias e ângulos na elaboração de critérios de seleção de vértices, para movimentos de intercâmbio entre rotas a serem utilizados pela Metaheurística Busca Tabu, é uma alternativa interessante para o Problema de Roteamento de Veículos. Concluímos também que podemos melhorar as combinações de critérios apresentadas neste trabalho, visto que obtivemos valores próximos simplesmente combinando tais critérios de maneira aleatória.

Para próximos trabalhos, podemos incluir os mecanismos da Busca Tabu (Critérios de Abstração, Intensificação Regional e Diversificação Global) não explorados na nossa implementação. Podemos também armazenar os vértices nas listas tabu, ao invés das arestas, facilitando com isso o emprego de heurísticas (2-Opt e/ou 3-Opt) em cada rota e a qualquer momento, pois não temos que nos preocupar com as arestas incluídas nas listas tabu. Estamos pensando também numa alternativa, em que obtemos a solução exata para cada rota, para verificar posteriormente a atuação dos critérios apresentados diante desse cenário.

Apêndice A

Composição dos circuitos hamiltonianos correspondentes às simulações do capítulo 3.

Heurística de Bellmore e Nemhauser [2.394,52 Km]

São Paulo → (27,77 Km) → Jandira → (38,61 Km) → Jundiaí → (18,99 Km) → Jarinu → (50,30 Km) → Joanópolis → (51,38 Km) → Jacareí → (28,91 Km) → Jambeiro → (146,38 Km) → Jaguariúna → (91,80 Km) → Jumirim → (118,78 Km) → Jaú → (118,21 Km) → Jaboticabal → (63,45 Km) → Jardinópolis → (76,78 Km) → Jaborandi → (95,61 Km) → Jeriquara → (216,01 Km) → Jaci → (22,45 Km) → José Bonifácio → (107,18 Km) → Júlio Mesquita → (103,86 Km) → João Ramalho → (106,86 Km) → Junqueirópolis → (165,98 Km) → Jales → (556,57 Km) → Jacupiranga → (55,88 Km) → Juquiá → (71,19 Km) → Juquitiba → (61,55 Km) → **São Paulo**

Heurística de inserção [2.332,59 Km]

São Paulo → (73,47 Km) → Jacareí → (28,91 Km) → Jambeiro → (69,76 Km) → Joanópolis → (77,52 Km) → Jaguariúna → (203,70 Km) → Jardinópolis → (80,84 Km) → Jeriquara → (95,61 Km) → Jaborandi → (122,41 Km) → Jaci → (122,30 Km) → Jales → (124,62 Km) → José Bonifácio → (187,81 Km) → Junqueirópolis → (106,86 Km) → João Ramalho → (103,86 Km) → Júlio Mesquita → (173,77 Km) → Jaboticabal → (118,21 Km) → Jaú → (118,78 Km) → Jumirim → (108,04 Km) → Jarinu → (18,99 Km) → Jundiaí → (38,61 Km) → Jandira → (114,51 Km) → Juquiá → (55,88 Km) → Jacupiranga → (126,57 Km) → Juquitiba → (61,55 Km) → **São Paulo**

Heurística das economias [2.093,68 Km]

São Paulo → (27,77 Km) → Jandira → (47,86 Km) → Juquitiba → (71,19 Km) → Juquiá → (55,88 Km) → Jacupiranga → (179,61 Km) → Jumirim → (118,78 Km) → Jaú → (130,67 Km) → Júlio Mesquita → (103,86 Km) → João Ramalho → (106,86 Km) → Junqueirópolis → (165,98 Km) → Jales → (122,30 Km) → Jaci → (22,45 Km) → José Bonifácio → (138,64 Km) → Jaborandi → (63,58 Km) → Jaboticabal → (129,66 Km) → Jeriquara → (80,84 Km) → Jardinópolis → (203,70 Km) → Jaguariúna → (77,52 Km) → Joanópolis → (69,76 Km) → Jambeiro → (28,91 Km) → Jacareí → (80,53 Km) → Jarinu → (18,99 Km) → Jundiaí → (48,33 Km) → **São Paulo**

Heurística 2-Opt [2.062,37 Km]

São Paulo → (73,47 Km) → Jacareí → (28,91 Km) → Jambeiro → (69,76 Km) → Joanópolis → (50,30 Km) → Jarinu → (18,99 Km) → Jundiaí → (54,01 Km) → Jaguariúna → (91,80 Km) → Jumirim → (118,78 Km) → Jaú → (118,21 Km) → Jaboticabal → (63,45 Km) → Jardinópolis → (80,84 Km) → Jeriquara → (95,61 Km) → Jaborandi → (122,41 Km) → Jaci → (22,45 Km) → José Bonifácio → (124,62 Km) → Jales → (165,98 Km) → Junqueirópolis → (106,86 Km) → João Ramalho → (103,86 Km) → Júlio Mesquita → (349,34 Km) → Jacupiranga → (55,88 Km) → Juquiá → (71,19 Km) → Juquitiba → (47,86 Km) → Jandira → (27,77 Km) → **São Paulo**

Método de Planos de Corte [2.025,03 Km]

São Paulo → (73,47 Km) → Jacareí → (28,91 Km) → Jambeiro → (69,76 Km) → Joanópolis → (50,30 Km) → Jarinu → (18,99 Km) → Jundiaí → (54,01 Km) → Jaguariúna → (211,74 Km) → Jaboticabal → (63,45 Km) → Jardinópolis → (80,84 Km) → Jeriquara → (95,61 Km) → Jaborandi → (122,41 Km) → Jaci → (22,45 Km) → José Bonifácio → (124,62 Km) → Jales → (165,98 Km) → Junqueirópolis → (106,86 Km) → João Ramalho → (103,86 Km) → Júlio Mesquita → (130,67 Km) → Jaú → (118,78 Km) → Jumirim → (179,61 Km) → Jacupiranga → (55,88 Km) → Juquiá → (71,19 Km) → Juquitiba → (47,86 Km) → Jandira → (27,77 Km) → **São Paulo**

Apêndice B

Composições das rotas correspondentes às simulações do capítulo 4.

Heurística de Clarke e Wright [4.016,27 Km]

rota**[1]** . demanda**[910]** . trecho**[1.097,67 Km]**
Agudos → (227,63 Km) → Apiaí → (143,05 Km) → Alambari → (29,53 Km) → Araçoiaba da Serra → (89,91 Km) → Americana → (76,13 Km) → Analândia → (126,32 Km) → Altinópolis → (112,55 Km) → Aramina → (292,55 Km) → **Agudos**

rota**[2]** . demanda**[780]** . trecho**[501,72 Km]**
Agudos → (122,76 Km) → Américo Brasiliense → (99,64 Km) → Águas de São Pedro → (33,48 Km) → Anhembi → (82,97 Km) → Angatuba → (76,20 Km) → Arandu → (34,13 Km) → Águas de Santa Bárbara → (52,54 Km) → **Agudos**

rota**[3]** . demanda**[910]** . trecho**[793,64 Km]**
Agudos → (216,35 Km) → Altair → (79,68 Km) → Américo de Campos → (13,41 Km) → Álvares Florence → (102,09 Km) → Aparecida d'Oeste → (71,44 Km) → Andradina → (103,65 Km) → Araçatuba → (50,69 Km) → Alto Alegre → (156,33 Km) → **Agudos**

rota**[4]** . demanda**[910]** . trecho**[934,04 Km]**
Agudos → (212,18 Km) → Aguaí → (29,83 Km) → Águas da Prata → (60,28 Km) → Águas de Lindóia → (149,82 Km) → Aparecida → (158,39 Km) → Amparo → (87,55 Km) → Araçariguama → (22,57 Km) → Alumínio → (213,42 Km) → **Agudos**

rota**[5]** . demanda**[910]** . trecho**[689,20 Km]**
Agudos → (153,03 Km) → Adolfo → (156,32 Km) → Adamantina → (45,69 Km) → Alfredo Marcondes → (14,79 Km) → Álvares Machado → (25,73 Km) → Anhumas → (173,18 Km) → Álvaro de Carvalho → (40,91 Km) → Alvinlândia → (79,55 Km) → **Agudos**

Heurística de Clarke e Wright com constante θ de Gaskell igual a 1.2 [3.852,40 Km]

rota**[1] .** demanda**[780] .** trecho**[821,01 Km]**
Agudos → (292,55 Km) → Aramina → (112,55 Km) → Altinópolis → (108,84 Km) → Américo Brasiliense → (118,14 Km) → Anhembi → (102,26 Km) → Arandu → (34,13 Km) → Águas de Santa Bárbara → (52,54 Km) → **Agudos**

rota**[2] .** demanda**[910] .** trecho**[731,25 Km]**
Agudos → (227,63 Km) → Apiaí → (121,59 Km) → Angatuba → (52,87 Km) → Alambari → (29,53 Km) → Araçoiaba da Serra → (36,67 Km) → Alumínio → (22,57 Km) → Araçariguama → (125,06 Km) → Águas de São Pedro → (115,33 Km) → **Agudos**

rota**[3] .** demanda**[910] .** trecho**[793,64 Km]**
Agudos → (216,35 Km) → Altair → (79,68 Km) → Américo de Campos → (13,41 Km) → Álvares Florence → (102,09 Km) → Aparecida d'Oeste → (71,44 Km) → Andradina → (103,65 Km) → Araçatuba → (50,69 Km) → Alto Alegre → (156,33 Km) → **Agudos**

rota**[4] .** demanda**[910] .** trecho**[835,58 Km]**
Agudos → (172,84 Km) → Americana → (58,26 Km) → Amparo → (28,83 Km) → Águas de Lindóia → (149,82 Km) → Aparecida → (183,13 Km) → Águas da Prata → (29,83 Km) → Aguaí → (71,39 Km) → Analândia → (141,48 Km) → **Agudos**

rota**[5] .** demanda**[910] .** trecho**[670,92 Km]**
Agudos → (86,87 Km) → Álvaro de Carvalho → (94,19 Km) → Adolfo → (156,32 Km) → Adamantina → (45,69 Km) → Alfredo Marcondes → (14,79 Km) → Álvares Machado → (25,73 Km) → Anhumas → (167,78 Km) → Alvinlândia → (79,55 Km) → **Agudos**

Procedimentos de Melhoria [3.785,91 Km]

rota**[1] .** demanda**[910] .** trecho**[835,99 Km]**
Agudos → (292,55 Km) → Aramina → (112,55 Km) → Altinópolis → (108,84 Km) → Américo Brasiliense → (99,64 Km) → Águas de São Pedro → (33,48 Km) → Anhembi → (102,26 Km) → Arandu → (34,13 Km) → Águas de Santa Bárbara → (52,54 Km) → **Agudos**

rota**[2] .** demanda**[780] .** trecho**[705,99 Km]**
Agudos → (127,78 Km) → Angatuba → (121,59 Km) → Apiaí → (143,05 Km) → Alambari → (29,53 Km) → Araçoiaba da Serra → (36,67 Km) → Alumínio → (22,57 Km) → Araçariguama → (224,80 Km) → **Agudos**

rota**[3] .** demanda**[910] .** trecho**[821,18 Km]**
Agudos → (216,35 Km) → Altair → (79,68 Km) → Américo de Campos → (13,41 Km) → Álvares Florence → (102,09 Km) → Aparecida d'Oeste → (71,44 Km) → Andradina → (103,65 Km) → Araçatuba → (81,53 Km) → Adolfo → (153,03 Km) → **Agudos**

rota**[4] -** demanda**[910] -** trecho**[835,58 Km]**
Agudos → (172,84 Km) → Americana → (58,26 Km) → Amparo → (28,83 Km) → Águas de Lindóia → (149,82 Km) → Aparecida → (183,13 Km) → Águas da Prata → (29,83 Km) → Aguaí → (71,39 Km) → Analândia → (141,48 Km) → **Agudos**

rota**[5] -** demanda**[910] -** trecho**[587,17 Km]**
Agudos → (86,87 Km) → Álvaro de Carvalho → (71,90 Km) → Alto Alegre → (94,86 Km) → Adamantina → (45,69 Km) → Alfredo Marcondes → (14,79 Km) → Álvares Machado → (25,73 Km) → Anhumas → (167,78 Km) → Alvinlândia → (79,55 Km) → **Agudos**

Heurística de Mole e Jameson [3.729,68 Km]

rota**[1] -** demanda**[910] -** trecho**[712,05 Km]**
Agudos → (127,78 Km) → Angatuba → (52,87 Km) → Alambari → (29,53 Km) → Araçoiaba da Serra → (56,72 Km) → Araçariguama → (22,57 Km) → Alumínio → (193,63 Km) → Apiaí → (154,62 Km) → Arandu → (74,33 Km) → **Agudos**

rota**[2] -** demanda**[780] -** trecho**[782,35 Km]**
Agudos → (216,35 Km) → Altair → (141,41 Km) → Aramina → (112,55 Km) → Altinópolis → (126,32 Km) → Analândia → (56,74 Km) → Águas de São Pedro → (33,48 Km) → Anhembi → (95,50 Km) → **Agudos**

rota**[3] -** demanda**[910] -** trecho**[755,68 Km]**
Agudos → (153,03 Km) → Adolfo → (105,04 Km) → Américo de Campos → (13,41 Km) → Álvares Florence → (102,09 Km) → Aparecida d'Oeste → (71,44 Km) → Andradina → (103,65 Km) → Araçatuba → (50,69 Km) → Alto Alegre → (156,33 Km) → **Agudos**

rota**[4] -** demanda**[910] -** trecho**[611,97 Km]**
Agudos → (52,54 Km) → Águas de Santa Bárbara → (72,23 Km) → Alvinlândia → (167,78 Km) → Anhumas → (25,73 Km) → Álvares Machado → (14,79 Km) → Alfredo Marcondes → (45,69 Km) → Adamantina → (146,34 Km) → Álvaro de Carvalho → (86,87 Km) → **Agudos**

rota**[5] -** demanda**[910] -** trecho**[867,63 Km]**
Agudos → (172,84 Km) → Americana → (58,26 Km) → Amparo → (28,83 Km) → Águas de Lindóia → (149,82 Km) → Aparecida → (183,13 Km) → Águas da Prata → (29,83 Km) → Aguaí → (122,16 Km) → Américo Brasiliense → (122,76 Km) → **Agudos**

Heurística de Gillet e Miller [3.719,84 Km]

rota**[1]** . demanda**[910]** . trecho**[681,77 Km]**
Agudos → (127,78 Km) → Angatuba → (52,87 Km) → Alambari → (29,53 Km) → Araçoiaba da Serra → (36,67 Km) → Alumínio → (193,63 Km) → Apiaí → (154,62 Km) → Arandu → (34,13 Km) → Águas de Santa Bárbara → (52,54 Km) → **Agudos**

rota**[2]** . demanda**[910]** . trecho**[835,53 Km]**
Agudos → (115,33 Km) → Águas de São Pedro → (57,95 Km) → Americana → (58,26 Km) → Amparo → (28,83 Km) → Águas de Lindóia → (149,82 Km) → Aparecida → (199,35 Km) → Araçariguama → (130,49 Km) → Anhembi → (95,50 Km) → **Agudos**

rota**[3]** . demanda**[910]** . trecho**[880,11 Km]**
Agudos → (122,76 Km) → Américo Brasiliense → (63,51 Km) → Analândia → (71,39 Km) → Aguaí → (29,83 Km) → Águas da Prata → (122,31 Km) → Altinópolis → (112,55 Km) → Aramina → (141,41 Km) → Altair → (216,35 Km) → **Agudos**

rota**[4]** . demanda**[910]** . trecho**[755,68 Km]**
Agudos → (153,03 Km) → Adolfo → (105,04 Km) → Américo de Campos → (13,41 Km) → Álvares Florence → (102,09 Km) → Aparecida d'Oeste → (71,44 Km) → Andradina → (103,65 Km) → Araçatuba → (50,69 Km) → Alto Alegre → (156,33 Km) → **Agudos**

rota**[5]** . demanda**[780]** . trecho**[566,75 Km]**
Agudos → (79,55 Km) → Alvinlândia → (167,78 Km) → Anhumas → (25,73 Km) → Álvares Machado → (14,79 Km) → Alfredo Marcondes → (45,69 Km) → Adamantina → (146,34 Km) → Álvaro de Carvalho → (86,87 Km) → **Agudos**

Metaheurística Busca Tabu [3.675,39 Km]

rota**[1]** - demanda**[910]** - trecho**[821,18 Km]**
Agudos → (153,03 Km) → Adolfo → (81,53 Km) → Araçatuba → (103,65 Km) → Andradina → (71,44 Km) → Aparecida d'Oeste → (102,09 Km) → Álvares Florence → (13,41 Km) → Américo de Campos → (79,68 Km) → Altair → (216,35 Km) → **Agudos**

rota**[2]** - demanda**[910]** - trecho**[845,53 Km]**
Agudos → (213,42 Km) → Alumínio → (22,57 Km) → Araçariguama → (199,35 Km) → Aparecida → (149,82 Km) → Águas de Lindóia → (28,83 Km) → Amparo → (58,26 Km) → Americana → (57,95 Km) → Águas de Sao Pedro → (115,33 Km) → **Agudos**

rota**[3]** - demanda**[910]** - trecho**[635,78 Km]**
Agudos → (52,54 Km) → Águas de Santa Barbara → (34,13 Km) → Arandu → (154,62 Km) → Apiaí → (121,59 Km) → Angatuba → (52,87 Km) → Alambari → (29,53 Km) → Araçoiaba da Serra → (95,00 Km) → Anhembi → (95,50 Km) → **Agudos**

rota**[4]** - demanda**[780]** - trecho**[785,73 Km]**
Agudos → (122,76 Km) → Américo Brasiliense → (185,41 Km) → Aramina → (112,55 Km) → Altinópolis → (122,31 Km) → Águas da Prata → (29,83 Km) → Aguaí → (71,39 Km) → Analândia → (141,48 Km) → **Agudos**

rota**[5]** - demanda**[910]** - trecho**[587,17 Km]**
Agudos → (86,87 Km) → Álvaro de Carvalho → (71,90 Km) → Alto Alegre → (94,86 Km) → Adamantina → (45,69 Km) → Alfredo Marcondes → (14,79 Km) → Álvares Machado → (25,73 Km) → Anhumas → (167,78 Km) → Alvinlândia → (79,55 Km) → **Agudos**

Apêndice C

Exemplo de cálculo de distância a partir das coordenadas geográficas.

São Paulo (latitude=23°32'52"; longitude=46°38'07")
Americana (latitude=22°44'20"; longitude=47°19'52")

Inicialmente, encontramos as coordenadas no formato decimal

São Paulo *La*1 = LatDec(23°32'52") = 23 + 32/60 + 52/3600 = 23,55°
*Lo*1 = LonDec(46°38'07") = 46 + 38/60 + 07/3600 = 46,64°
Americana *La*2 = LatDec(22°44'20") = 22 + 44/60 + 20/3600 = 22,74°
*Lo*2 = LonDec(47°19'52") = 47 + 19/60 + 52/3600 = 47,33°

Em seguida, calculamos a distância

$d = 111,2 * \Delta$
$\Delta = \text{acos}[\text{sen}(La1) * \text{sen}(La2) + \cos(La1) * \cos(La2) * \cos(Lo1 - Lo2)]$
$\Delta = \text{acos}[\text{sen}(23,55) * \text{sen}(22,74) + \cos(23,55) * \cos(22,74) * \cos(46,64 - 47,33)]$
$\Delta = \text{acos}[0,3995492 * 0,3865500 + 0,9167118 * 0,9222685 * 0,9999275]$
$\Delta = \text{acos}[0,9998388]$
$\Delta = 1,0287882°$ (cada grau corresponde aproximadamente a um valor entre 110,5 e 111,5 Km)
$d = 111,2 * 1,0287882$
***d* = 114,40 Km**

A distância calculada apresentou um pequeno desvio da tabelada, provavelmente devido a diferença nos arredondamentos. Lembrando que em alguns ambientes, as funções trigonométricas trabalham com os valores em radianos, sendo portanto necessário realizar as devidas conversões.

Distâncias por estradas [DE] x Distâncias calculadas através das coordenadas geográficas [DG]				
Trecho	**DE**	**DG**	**DE / DG**	**Diferença %**
São Paulo – Americana	128	114,69	1,1161	11,61
São Paulo – Bauru	343	284,20	1,2069	20,69
São Paulo – Campinas	95	86,10	1,1034	10,34
São Paulo – Dracena	647	552,98	1,1700	17,00
São Paulo – Embu	25	24,85	1,0060	0,60
São Paulo – Franca	400	343,84	1,1633	16,33
São Paulo – Guaratinguetá	176	168,46	1,0448	4,48
São Paulo – Holambra	125	110,72	1,1290	12,90
São Paulo – Itu	103	75,31	1,3677	36,77
São Paulo – Jundiaí	60	48,33	1,2415	24,15
São Paulo – Limeira	150	135,03	1,1109	11,09
São Paulo – Marília	444	369,91	1,2003	20,03
São Paulo – Novo Horizonte	429	352,29	1,2177	21,77
São Paulo – Ourinhos	370	336,77	1,0987	9,87
São Paulo – Piracicaba	162	138,33	1,1711	17,11
São Paulo – Queluz	234	221,25	1,0576	5,76
São Paulo – Ribeirão Preto	314	290,21	1,0820	8,20
São Paulo – Santos	85	55,22	1,5393	53,93
São Paulo – Taubaté	130	124,57	1,0436	4,36
São Paulo – Ubatuba	224	160,17	1,3985	39,85
São Paulo – Vinhedo	79	67,22	1,1752	17,52
São Paulo – Zacarias	567	447,67	1,2666	26,66
			Diferença média	**17,77 %**

Observação: Informalmente, as distâncias por estradas são, em média, 20% maiores que as calculadas com o uso de coordenadas geográficas.

Apêndice D

Nesta parte, realizamos alguns testes com diferentes instâncias do PCV. Os resultados estão apresentados na tabela seguinte, onde a primeira coluna indica a instância que está sendo utilizada, no formato: SP-número de municípios-letra inicial dos municípios. As outras colunas, com exceção da última, possuem duas linhas, onde a primeira indica o resultado em quilômetros, e o tempo gasto em segundos utilizado pela estratégia. A segunda linha fornece os mesmos dados da primeira, após utilização da heurística de melhoria 2-Opt.

Instância	BN		IN		EC		CC		CA		CM		PC
SP-52-A	3864,80	0	4067,86	0	3485,42	3	3290,86	1	3432,98	0	3207,59	4	**3177,78**
	3350,39	6	3238,69	14	3278,77	11	3213,13	6	3277,14	5	3207,59	6	11
SP-42-B	2809,31	0	3226,63	0	2776,11	1	2741,77	0	2765,98	0	2741,77	2	**2652,80**
	2722,95	3	2789,52	3	2737,60	3	2723,86	1	2714,30	2	2723,86	3	59
SP-69-CD	4120,60	0	4201,46	2	4072,96	9	3866,24	0	3713,47	0	3682,74	11	**3535,44**
	3615,79	25	3812,10	23	3760,91	41	3760,02	19	3671,46	10	3682,74	18	51
SP-27-EF	2197,38	0	2489,07	0	2538,42	0	**2090,39**	0	**2090,39**	0	**2090,39**	0	**2090,39**
	2090,39	1	2439,05	1	**2090,39**	1	**2090,39**	0	**2090,39**	0	**2090,39**	0	1
SP-30-GH	2641,11	0	2633,86	0	2445,45	0	2238,85	0	2395,12	0	2222,82	1	**2211,71**
	2239,21	1	2486,97	1	2290,47	1	**2211,71**	1	2328,41	1	**2211,71**	1	1
SP-57-I	3559,01	0	3802,90	1	3442,48	3	3316,18	0	3562,53	0	3308,17	5	**3127,35**
	3168,49	19	3722,54	9	3406,28	8	3267,41	6	3390,52	14	3259,06	11	234
SP-40-JL	3309,29	0	3042,13	0	2578,49	0	**2468,10**	0	2652,50	0	**2468,10**	2	**2468,10**
	2469,61	3	2684,08	2	2525,48	2	**2468,10**	1	2543,92	2	**2468,10**	2	3
SP-45-M	3315,26	0	3159,91	0	2879,87	1	2547,78	0	2518,38	0	2515,70	2	**2502,46**
	2543,96	7	2902,35	4	2505,14	5	2517,29	2	2505,14	1	2515,70	4	1
SP-33-NO	3043,53	0	2830,86	0	2556,81	0	2413,72	0	2386,11	0	2378,15	1	**2364,72**
	2509,02	1	2676,02	1	2445,94	1	2378,15	1	2386,11	1	2378,15	1	1
SP-74-P	4699,59	0	4255,50	2	3617,27	13	3723,51	0	3422,15	0	3386,83	13	**3367,25**
	3463,50	43	3570,34	60	3400,89	35	3490,14	20	3372,73	12	3386,83	22	31
SP-33-QR	3636,19	0	3497,63	0	3240,25	0	3063,33	0	3151,23	0	2971,12	1	**2939,58**
	3243,07	1	3084,88	2	3029,04	2	3037,71	1	3147,89	1	2971,12	1	1
SP-86-S	4600,31	1	4299,15	4	3733,87	27	3549,73	1	3822,03	1	3523,05	24	**3408,52**
	3808,51	83	3690,80	136	3546,13	122	3462,12	93	3556,15	77	3523,05	41	2780
SP-34-T	3119,08	0	3080,61	0	2484,51	0	2441,55	0	2426,97	0	**2321,28**	1	**2321,28**
	2321,28	1	2485,57	2	2369,49	2	**2321,28**	1	2403,64	1	**2321,28**	1	1
SP-23-UVZ	2225,67	0	2080,96	0	2044,68	0	1795,34	0	1813,11	0	1795,18	0	**1783,03**
	1789,87	0	**1783,03**	0	1840,98	0	1795,34	0	1788,34	0	1795,18	0	0

Legenda
BN – Heurística de Bellmore e Nemhauser
IN – Heurística de Inserção
EC – Heurística das Economias
CC – Heurística do Casco Convexo + Inserção mais barata
CA – Heurística do Casco Convexo + Inserção do maior ângulo
CM – Heurística do Casco Convexo Mista
PC – Método de Planos de Corte

Obs: Os valores em negrito correspondem ao **valor ótimo** encontrado para a instância.

Na seqüência, realizamos 3 testes para o PRV, utilizando o mesmo cenário proposto no capítulo 4. Entretanto, alteramos os valores das demandas dos municípios. Para o primeiro teste, as demandas não ultrapassaram o valor de 1/5 da capacidade dos veículos. Os valores obtidos, aleatoriamente, para cada município foram os seguintes:

Município	Demanda	Município	Demanda	Município	Demanda
Adamantina	68	Alto Alegre	109	Angatuba	198
Adolfo	179	Alumínio	59	Anhembi	146
Aguaí	187	Álvares Florence	43	Anhumas	51
Águas da Prata	34	Álvares Machado	29	Aparecida	141
Águas de Lindóia	66	Álvaro de Carvalho	148	Aparecida d'Oeste	146
Águas de Santa Bárbara	76	Alvinlândia	100	Apiaí	48
Águas de São Pedro	71	Americana	101	Araçariguama	179
Agudos (depósito)	0	Américo Brasiliense	160	Araçatuba	128
Alambari	95	Américo de Campos	82	Araçoiaba da Serra	46
Alfredo Marcondes	137	Amparo	157	Aramina	193
Altair	55	Analândia	13	Arandu	114
Altinópolis	9	Andradina	103		

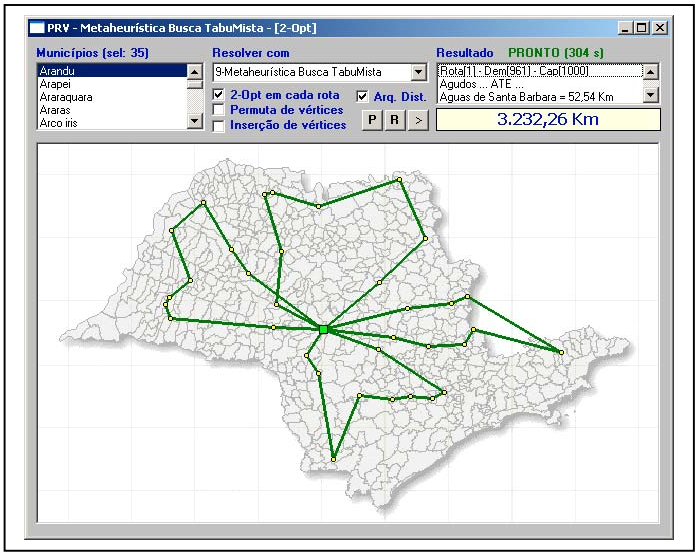
Os resultados estão apresentados nas tabelas seguintes, onde os valores contidos nas células correspondem à distância total a ser percorrida (em quilômetros), e o tempo gasto pela estratégia (em segundos) para obter a solução. Os resultados foram dispostos seguindo a mesma ordem de apresentação das estratégias no trabalho. Para as variações da Busca Tabu, utilizamos a solução de partida da primeira fase de testes do capítulo 5, adotando o tamanho das listas tabu igual a 7.

CW	CW*	CW**	MJ	GM
3505,75 1	3505,75 1	3322,66 1	3348,66 1	3313,35 139

BT1	BT2	BT3	BT4	BT5	BT6
3276,93 312	3239,79 301	3232,26 304	3268,19 307	3232,26 307	3232,26 308

Legenda

CW – Heurística de Clarke e Wright
CW* - CW com constante de Gaskell θ=1.0
CW** - CW com constante de Gaskell θ=3.6 + procedimentos de melhoria.
MJ – Heurística de Mole e Jameson com (μ=1, λ=0) + procedimentos de melhoria.
GM – Heurística de Gillett e Miller
BT1 – Busca Tabu Custo (capítulo 4)
BT2 – Busca Tabu Ângulo
BT3 – Busca Tabu (combinação 1)
BT4 – Busca Tabu (combinação 2)
BT5 – Busca Tabu (combinação 3)
BT6 – Busca Tabu (combinação aleatória)



Aspecto das rotas obtidas com a estratégia **BT3**

Para o segundo teste, as demandas utilizadas possuem valores entre ¼ e ½ da capacidade dos veículos. Os valores obtidos, aleatoriamente, para cada município foram os seguintes:

Município	Demanda	Município	Demanda	Município	Demanda
Adamantina	382	Alto Alegre	379	Angatuba	453
Adolfo	306	Alumínio	495	Anhembi	327
Aguaí	468	Álvares Florence	411	Anhumas	336
Águas da Prata	381	Álvares Machado	435	Aparecida	259
Águas de Lindóia	370	Álvaro de Carvalho	402	Aparecida d'Oeste	395
Águas de Santa Bárbara	348	Alvinlândia	324	Apiaí	316
Águas de São Pedro	483	Americana	423	Araçariguama	342
Agudos (depósito)	0	Américo Brasiliense	250	Araçatuba	273
Alambari	251	Américo de Campos	476	Araçoiaba da Serra	265
Alfredo Marcondes	268	Amparo	284	Aramina	354
Altair	446	Analândia	315	Arandu	371
Altinópolis	292	Andradina	282		

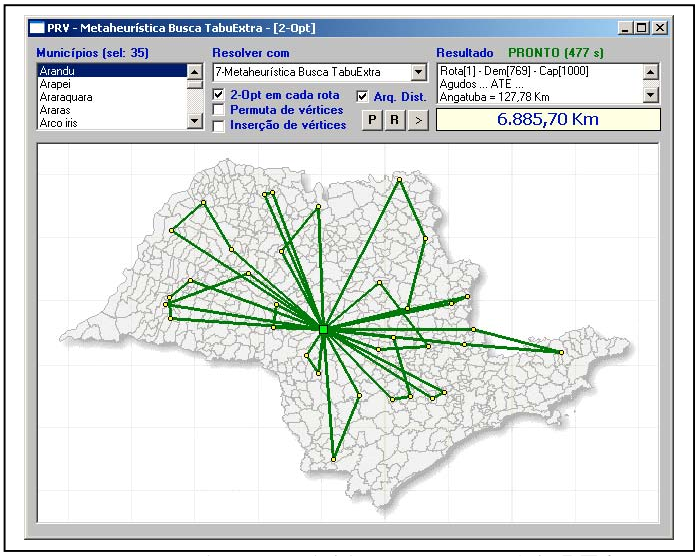
Os resultados estão apresentados nas tabelas seguintes, onde os valores contidos nas células correspondem à distância total a ser percorrida (em quilômetros), e o tempo gasto pela estratégia (em segundos) para obter a solução. Os resultados foram dispostos seguindo a mesma ordem de apresentação das estratégias no trabalho. Para as variações da Busca Tabu, utilizamos a solução de partida da primeira fase de testes do capítulo 5, adotando o tamanho das listas tabu igual a 7.

CW	CW*	CW**	MJ	GM
7272,04 0	7258,74 0	6972,75 2	7018,73 1	7653,26 548

BT1	BT2	BT3	BT4	BT5	BT6
6899,07 480	7015,54 473	6952,54 477	6917,40 479	6899,07 483	6885,70 477

Legenda

CW – Heurística de Clarke e Wright
CW* - CW com constante de Gaskell θ=0.4
CW** - CW com constante de Gaskell θ=1.7 + procedimentos de melhoria.
MJ – Heurística de Mole e Jameson com (μ=1, λ=2) + procedimentos de melhoria.
GM – Heurística de Gillett e Miller
BT1 – Busca Tabu Custo (capítulo 4)
BT2 – Busca Tabu Ângulo
BT3 – Busca Tabu (combinação 1)
BT4 – Busca Tabu (combinação 2)
BT5 – Busca Tabu (combinação 3)
BT6 – Busca Tabu (combinação aleatória)



Aspecto das rotas obtidas com a estratégia **BT6**

Para o terceiro teste, as demandas possuem valores que variam de uma unidade até a capacidade do veículo. Os valores obtidos, aleatoriamente, para cada município foram os seguintes:

Município	Demanda	Município	Demanda	Município	Demanda
Adamantina	721	Alto Alegre	130	Angatuba	422
Adolfo	512	Alumínio	231	Anhembi	430
Aguaí	32	Álvares Florence	150	Anhumas	23
Águas da Prata	910	Álvares Machado	313	Aparecida	342
Águas de Lindóia	59	Álvaro de Carvalho	220	Aparecida d'Oeste	531
Águas de Santa Bárbara	61	Alvinlândia	142	Apiaí	618
Águas de São Pedro	73	Americana	249	Araçariguama	626
Agudos (depósito)	0	Américo Brasiliense	211	Araçatuba	812
Alambari	80	Américo de Campos	324	Araçoiaba da Serra	19
Alfredo Marcondes	121	Amparo	336	Aramina	711
Altair	114	Analândia	521	Arandu	41
Altinópolis	92	Andradina	415		

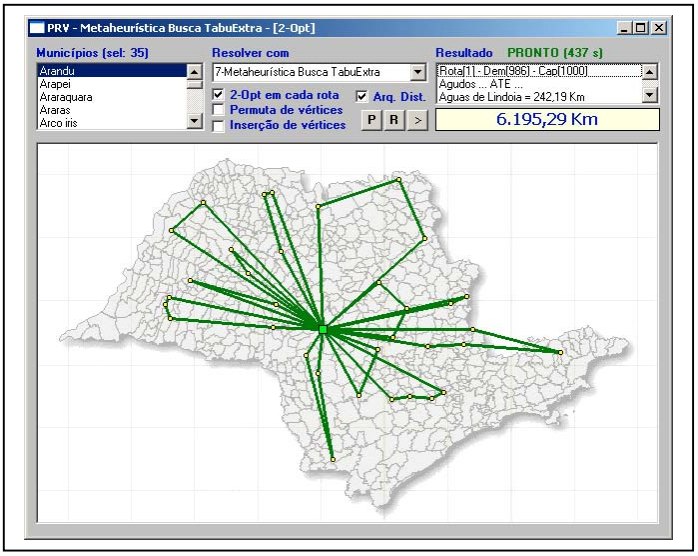
Os resultados estão apresentados nas tabelas seguintes, onde os valores contidos nas células correspondem à distância total a ser percorrida (em quilômetros), e o tempo gasto pela estratégia (em segundos) para obter a solução. Os resultados foram dispostos seguindo a mesma ordem de apresentação das estratégias no trabalho. Para as variações da Busca Tabu, utilizamos a solução de partida da primeira fase de testes do capítulo 5, adotando o tamanho das listas tabu igual a 7.

CW	CW*	CW**	MJ	GM
6325,55 1	6325,55 0	6259,87 1	6596,97 1	7276,12 355

BT1	BT2	BT3	BT4	BT5	BT6
6206,67 440	6378,79 432	6195,29 437	6195,29 437	6195,29 439	6195,29 438

Legenda

CW – Heurística de Clarke e Wright
CW* - CW com constante de Gaskell θ=0.8
CW** - CW com constante de Gaskell θ=0.4 + procedimentos de melhoria.
MJ – Heurística de Mole e Jameson com (μ=0.5, λ=1.5) + procedimentos de melhoria.
GM – Heurística de Gillett e Miller
BT1 – Busca Tabu Custo (capítulo 4)
BT2 – Busca Tabu Ângulo
BT3 – Busca Tabu (combinação 1)
BT4 – Busca Tabu (combinação 2)
BT5 – Busca Tabu (combinação 3)
BT6 – Busca Tabu (combinação aleatória)



Aspecto das rotas obtidas com a estratégia **BT4**

Obs: Os testes deste apêndice foram realizados numa estação com processador de 2.4 GHz e 512 MB de RAM.

Anexo A

Tabela de coordenadas geográficas dos municípios do estado de São Paulo.

Município	Coordenadas Geográficas		Município	Coordenadas Geográficas	
	Latitude S	Longitude O		Latitude S	Longitude O
Adamantina	21°41'16"	51°04'30"	Adolfo	21°14'01"	49°38'44"
Aguaí	22°03'30"	46°58'30"	Águas da Prata	21°56'15"	46°43'00"
Águas de Lindóia	22°28'26"	46°37'55"	Águas de Santa Bárbara	22°52'45"	49°14'25"
Águas de São Pedro	22°35'55"	47°52'30"	Agudos	22°28'03"	48°59'20"
Alambari	23°33'04"	47°54'03"	Alfredo Marcondes	21°57'16"	51°24'42"
Altair	20°31'23"	49°03'34"	Altinópolis	21°01'25"	47°22'28"
Alto Alegre	21°34'47"	50°09'53"	Alumínio	23°32'16"	47°15'29"
Álvares Florence	20°19'08"	49°54'43"	Álvares Machado	22°04'35"	51°28'08"
Álvaro de Carvalho	22°04'40"	49°43'14"	Alvinlândia	22°26'37"	49°45'45"
Americana	22°44'20"	47°19'52"	Américo Brasiliense	21°43'41"	48°06'15"
Américo de Campos	20°17'53"	49°47'07"	Amparo	22°42'09"	46°45'52"
Analândia	22°07'38"	47°39'50"	Andradina	20°53'47"	51°22'35"
Angatuba	23°29'16"	48°24'53"	Anhembi	22°47'25"	48°07'36"
Anhumas	22°17'41"	51°23'10"	Aparecida	22°50'25"	45°13'37"
Aparecida d'Oeste	20°27'05"	50°52'52"	Apiaí	24°30'36"	48°50'23"
Araçariguama	23°26'24"	47°03'51"	Araçatuba	21°11'50"	50°25'52"
Araçoiaba da Serra	23°30'14"	47°36'57"	Aramina	20°05'15"	47°47'08"
Arandu	23°08'00"	49°03'12"	Arapeí	22°40'24"	44°26'29"
Araraquara	21°47'36"	48°10'49"	Araras	22°21'20"	47°23'12"
Arco Íris	21°46'24"	50°28'00"	Arealva	22°01'39"	48°54'41"
Areias	22°34'51"	44°41'48"	Areiópolis	22°40'07"	48°39'45"
Ariranha	21°11'12"	48°47'15"	Artur Nogueira	22°34'26"	47°10'15"
Arujá	23°23'45"	46°19'20"	Aspásia	20°09'45"	50°43'37"
Assis	22°39'39"	50°25'13"	Atibaia	23°06'59"	46°33'06"
Auriflama	20°41'13"	50°33'20"	Avaí	22°09'18"	49°19'53"
Avanhandava	21°27'37"	49°57'03"	Avaré	23°05'47"	48°55'01"
Bady Bassitt	20°55'05"	49°26'42"	Balbinos	21°54'05"	49°21'17"
Bálsamo	20°44'05"	49°34'59"	Bananal	22°40'44"	44°19'07"
Barão de Antonina	23°37'38"	49°33'45"	Barbosa	21°15'59"	49°57'02"
Bariri	22°04'31"	48°44'22"	Barra Bonita	22°29'39"	48°33'51"
Barra do Chapéu	24°28'20"	49°01'24"	Barra do Turvo	24°45'14"	48°30'19"
Barretos	20°33'18"	48°34'26"	Barrinha	21°11'30"	48°09'35"
Barueri	23°30'45"	46°52'25"	Bastos	21°55'13"	50°44'07"

Batatais	20°53'34"	47°34'09"	Bauru	22°19'18"	49°04'13"
Bebedouro	20°56'21"	48°29'39"	Bento de Abreu	21°16'15"	50°48'43"
Bernardino de Campos	23°00'36"	49°28'44"	Bertioga	23°50'47"	46°08'21"
Bilac	21°24'14"	50°28'30"	Birigui	21°17'25"	50°20'20"
Biritiba Mirim	23°34'20"	46°02'15"	Boa Esperança do Sul	21°59'25"	48°23'15"
Bocaina	22°08'08"	48°31'10"	Bofete	23°06'05"	48°15'26"
Boituva	23°17'02"	47°40'25"	Bom Jesus dos Perdões	23°08'05"	46°27'53"
Bom Sucesso de Itararé	24°18'59"	49°08'26"	Borá	22°16'09"	50°32'39"
Boracéia	22°11'40"	48°46'41"	Borborema	21°37'10"	49°04'30"
Borebi	22°34'10"	48°58'16"	Botucatu	22°52'20"	48°26'37"
Bragança Paulista	22°57'53"	46°32'10"	Braúna	21°30'01"	50°19'05"
Brejo Alegre	21°10'00"	50°11'02"	Brodowski	20°59'19"	47°39'30"
Brotas	22°10'01"	48°07'37"	Buri	23°47'40"	48°35'35"
Buritama	21°04'01"	50°08'53"	Buritizal	20°11'25"	47°42'25"
Cabrália Paulista	22°27'20"	49°20'20"	Cabreúva	23°18'25"	47°07'55"
Caçapava	23°04'50"	45°42'37"	Cachoeira Paulista	22°39'41"	45°00'32"
Caconde	21°31'46"	46°38'45"	Cafelândia	21°48'04"	49°36'36"
Caiabu	22°00'46"	51°09'05"	Caieiras	23°21'35"	46°44'27"
Caiuá	21°49'50"	51°59'16"	Cajamar	23°21'25"	46°52'40"
Cajati	24°43'39"	48°07'30"	Cajobi	20°52'40"	48°48'31"
Cajuru	21°16'30"	47°18'23"	Campina do Monte Alegre	23°35'30"	48°28'35"
Campinas	22°53'20"	47°04'40"	Campo Limpo Paulista	23°12'20"	46°47'13"
Campos do Jordão	22°43'45"	45°34'47"	Campos Novos Paulista	22°36'05"	50°00'16"
Cananéia	25°00'56"	47°55'33"	Canas	22°42'13"	45°03'18"
Cândido Mota	22°44'52"	50°23'08"	Cândido Rodrigues	21°19'38"	48°37'53"
Canitar	23°00'22"	49°46'27"	Capão Bonito	24°00'14"	48°20'54"
Capela do Alto	23°28'15"	47°44'17"	Capivari	22°59'56"	47°30'16"
Caraguatatuba	23°37'31"	45°24'44"	Carapicuíba	23°31'25"	46°50'10"
Cardoso	20°04'43"	49°54'56"	Casa Branca	21°46'30"	47°05'15"
Cássia dos Coqueiros	21°16'53"	47°10'05"	Castilho	20°52'05"	51°29'25"
Catanduva	21°08'05"	48°58'27"	Catiguá	21°03'07"	49°03'33"
Cedral	20°54'12"	49°16'10"	Cerqueira César	23°01'58"	49°09'52"
Cerquilho	23°09'38"	47°44'35"	Cesário Lange	23°13'30"	47°57'12"
Charqueada	22°30'30"	47°46'42"	Chavantes	23°02'05"	49°42'34"
Clementina	21°33'33"	50°26'54"	Colina	20°43'04"	48°32'38"
Colômbia	20°10'33"	48°41'19"	Conchal	22°20'00"	47°10'22"
Conchas	23°00'47"	48°00'39"	Cordeirópolis	22°28'50"	47°27'27"
Coroados	21°21'06"	50°16'53"	Coronel Macedo	23°37'50"	49°18'45"
Corumbataí	22°13'10"	47°37'27"	Cosmópolis	22°38'35"	47°11'40"
Cosmorama	20°28'39"	49°46'40"	Cotia	23°36'09"	46°55'52"
Cravinhos	21°20'27"	47°43'50"	Cristais Paulista	20°24'22"	47°25'13"
Cruzália	22°44'28"	50°47'35"	Cruzeiro	22°34'38"	44°57'30"
Cubatão	23°53'30"	46°25'30"	Cunha	23°04'27"	44°57'35"
Descalvado	21°54'25"	47°37'10"	Diadema	23°41'10"	46°37'25"
Dirce Reis	20°27'57"	50°36'31"	Divinolândia	21°39'40"	46°44'15"

Dobrada	21°31'05"	48°23'43"	Dois Córregos	22°21'53"	48°22'50"
Dolcinópolis	20°07'18"	50°30'52"	Dourado	22°06'53"	48°19'05"
Dracena	21°29'10"	51°32'05"	Duartina	22°24'51"	49°24'16"
Dumont	21°14'18"	47°58'25"	Echaporã	22°25'43"	50°12'12"
Eldorado	24°31'07"	48°06'28"	Elias Fausto	23°02'32"	47°22'25"
Elisiário	21°10'06"	49°06'35"	Embaúba	20°58'50"	48°49'55"
Embu	23°39'05"	46°51'05"	Embu-Guaçu	23°50'00"	46°48'45"
Emilianópolis	21°49'54"	51°28'50"	Engenheiro Coelho	22°29'01"	47°12'47"
Espírito Santo do Pinhal	22°11'35"	46°44'47"	Espírito Santo do Turvo	22°41'23"	49°25'48"
Estiva Gerbi	22°16'22"	46°57'10"	Estrela d' Oeste	20°17'10"	50°24'03"
Estrela do Norte	22°29'15"	51°39'40"	Euclides da Cunha Paulista	22°33'23"	52°35'45"
Fartura	23°23'16"	49°39'41"	Fernando Prestes	21°16'00"	48°41'15"
Fernandópolis	20°16'51"	50°15'01"	Fernão	22°21'32"	49°31'17"
Ferraz de Vasconcelos	23°32'30"	46°22'00"	Flora Rica	21°40'31"	51°22'53"
Floreal	20°40'38"	50°08'45"	Flórida Paulista	21°36'47"	51°10'24"
Florínia	22°54'10"	50°43'43"	Franca	20°32'18"	47°24'06"
Francisco Morato	23°16'53"	46°44'35"	Franco da Rocha	23°19'48"	46°43'20"
Gabriel Monteiro	21°31'51"	50°33'24"	Gália	22°17'35"	49°33'05"
Garça	22°12'55"	49°39'04"	Gastão Vidigal	20°47'56"	50°11'23"
Gavião Peixoto	21°50'19"	48°29'41"	General Salgado	20°38'49"	50°21'42"
Getulina	21°47'52"	49°55'48"	Glicério	21°22'50"	50°12'30"
Guaiçara	21°37'18"	49°47'50"	Guaimbê	21°54'40"	49°53'48"
Guaíra	20°19'03"	48°18'48"	Guapiaçu	20°47'41"	49°13'05"
Guapiara	24°11'03"	48°31'48"	Guará	20°25'40"	47°49'35"
Guaraçaí	21°01'46"	51°12'42"	Guaraci	20°29'55"	48°56'38"
Guarani d' Oeste	20°04'25"	50°20'25"	Guarantã	21°53'47"	49°35'25"
Guararapes	21°15'12"	50°38'41"	Guararema	23°24'50"	46°02'05"
Guaratinguetá	22°48'43"	45°11'41"	Guareí	23°22'17"	48°11'10"
Guariba	21°21'40"	48°13'43"	Guarujá	23°59'14"	46°13'49"
Guarulhos	23°28'12"	46°31'35"	Guatapará	21°29'47"	48°02'16"
Guzolândia	20°39'03"	50°39'51"	Herculândia	22°00'09"	50°23'18"
Holambra	22°37'55"	47°03'36"	Hortolândia	22°51'22"	47°13'05"
Iacanga	21°53'20"	49°01'06"	Iacri	21°51'35"	50°41'25"
Iaras	22°52'14"	49°09'45"	Ibaté	21°57'18"	47°59'45"
Ibirá	21°04'50"	49°14'28"	Ibirarema	22°49'05"	50°04'18"
Ibitinga	21°45'23"	48°49'08"	Ibiúna	23°39'20"	47°13'30"
Icém	20°20'28"	49°11'54"	Iepê	22°39'50"	51°04'32"
Igaraçu do Tietê	22°30'35"	48°33'18"	Igarapava	20°02'11"	47°44'54"
Igaratá	23°12'20"	46°09'27"	Iguape	24°42'28"	47°33'20"
Ilha Comprida	24°44'14"	47°32'58"	Ilha Solteira	20°25'42"	51°20'34"
Ilhabela	23°46'40"	45°21'28"	Indaiatuba	23°05'12"	47°13'06"
Indiana	22°10'38"	51°15'20"	Indiaporã	19°58'45"	50°17'28"
Inúbia Paulista	21°46'08"	50°57'35"	Ipaussu	23°03'10"	49°37'30"
Iperó	23°20'55"	47°41'15"	Ipeúna	22°26'08"	47°43'05"
Ipiguá	20°39'28"	49°23'13"	Iporanga	24°35'03"	48°35'24"

Ipuã	20°26'30"	48°00'50"	Iracemápolis	22°34'55"	47°31'12"
Irapuã	21°16'46"	49°24'32"	Irapuru	21°34'12"	51°20'47"
Itaberá	23°51'34"	49°08'14"	Itaí	23°24'55"	49°05'24"
Itajobi	21°19'05"	49°03'23"	Itaju	21°58'50"	48°48'15"
Itanhaém	24°11'01"	46°47'18"	Itaóca	24°38'31"	48°50'20"
Itapecerica da Serra	23°42'50"	46°50'56"	Itapetininga	23°35'08"	48°02'51"
Itapeva	23°58'53"	48°52'34"	Itapevi	23°32'45"	46°56'05"
Itapira	22°26'20"	46°49'32"	Itapirapuã Paulista	24°34'25"	49°10'02"
Itápolis	21°35'41"	48°48'49"	Itaporanga	23°42'13"	49°29'02"
Itapuí	22°14'00"	48°43'05"	Itapura	20°38'52"	51°30'32"
Itaquaquecetuba	23°29'12"	46°20'45"	Itararé	24°06'34"	49°19'56"
Itariri	24°17'25"	47°10'38"	Itatiba	23°00'18"	46°50'28"
Itatinga	23°06'04"	48°36'58"	Itirapina	22°13'04"	47°47'57"
Itirapuã	20°38'30"	47°13'09"	Itobi	21°43'52"	46°58'30"
Itu	23°15'34"	47°18'11"	Itupeva	23°09'12"	47°03'28"
Ituverava	20°20'16"	47°46'58"	Jaborandi	20°41'25"	48°24'47"
Jaboticabal	21°15'20"	48°19'16"	Jacareí	23°17'49"	45°58'09"
Jaci	20°53'00"	49°34'20"	Jacupiranga	24°41'23"	48°00'09"
Jaguariúna	22°42'18"	46°59'22"	Jales	20°16'17"	50°32'54"
Jambeiro	23°15'15"	45°41'24"	Jandira	23°31'45"	46°54'25"
Jardinópolis	21°01'12"	47°45'50"	Jarinu	23°06'05"	46°43'40"
Jaú	22°17'42"	48°33'39"	Jeriquara	20°18'42"	47°35'20"
Joanópolis	22°55'45"	46°16'24"	João Ramalho	22°15'00"	50°46'02"
José Bonifácio	21°03'10"	49°41'23"	Júlio Mesquita	22°00'43"	49°47'32"
Jumirim	23°05'13"	47°47'02"	Jundiaí	23°10'56"	46°53'29"
Junqueirópolis	21°30'51"	51°26'00"	Juquiá	24°19'14"	47°37'40"
Juquitiba	23°55'59"	47°04'10"	Lagoinha	23°05'20"	45°11'25"
Laranjal Paulista	23°02'50"	47°50'10"	Lavínia	21°10'00"	51°02'25"
Lavrinhas	22°34'13"	44°54'08"	Leme	22°11'04"	47°23'12"
Lençóis Paulista	22°35'49"	48°48'00"	Limeira	22°33'37"	47°24'12"
Lindóia	22°31'20"	46°38'55"	Lins	21°40'25"	49°45'23"
Lorena	22°44'02"	45°07'21"	Lourdes	20°58'04"	50°13'26"
Louveira	23°05'05"	46°56'48"	Lucélia	21°44'05"	51°00'19"
Lucianópolis	22°25'50"	49°31'20"	Luís Antônio	21°33'12"	47°42'08"
Luiziânia	21°40'25"	50°19'45"	Lupércio	22°24'52"	49°48'55"
Lutécia	22°20'24"	50°23'32"	Macatuba	22°30'02"	48°42'48"
Macaubal	20°48'25"	49°58'00"	Macedônia	20°08'45"	50°11'35"
Magda	20°38'35"	50°13'30"	Mairinque	23°32'35"	47°11'02"
Mairiporã	23°19'12"	46°35'18"	Manduri	23°00'08"	49°19'24"
Marabá Paulista	22°06'28"	51°57'47"	Maracaí	22°36'39"	50°40'01"
Marapoama	21°15'36"	49°07'44"	Mariápolis	21°47'45"	51°10'50"
Marília	22°13'10"	49°56'45"	Marinópolis	20°26'30"	50°49'30"
Martinópolis	22°08'56"	51°10'16"	Matão	21°36'10"	48°22'03"
Mauá	23°40'20"	46°27'10"	Mendonça	21°11'00"	49°34'52"
Meridiano	20°21'25"	50°10'20"	Mesópolis	19°58'01"	50°37'12"

Miguelópolis	20°10'37"	48°02'13"	Mineiros do Tietê	22°24'40"	48°27'05"
Mira Estrela	19°58'43"	50°08'23"	Miracatu	24°16'50"	47°27'40"
Mirandópolis	21°07'58"	51°06'11"	Mirante do Paranapanema	22°17'27"	51°54'27"
Mirassol	20°49'01"	49°30'47"	Mirassolândia	20°36'58"	49°27'54"
Mococa	21°28'16"	47°00'23"	Mogi das Cruzes	23°31'20"	46°11'52"
Mogi Guaçu	22°21'50"	46°56'35"	Moji Mirim	22°26'04"	46°57'32"
Mombuca	22°55'45"	47°33'48"	Monções	20°51'03"	50°05'30"
Mongaguá	24°05'35"	46°37'10"	Monte Alegre do Sul	22°40'50"	46°40'45"
Monte Alto	21°15'40"	48°29'42"	Monte Aprazível	20°46'16"	49°42'46"
Monte Azul Paulista	20°54'24"	48°38'36"	Monte Castelo	21°18'00"	51°34'10"
Monte Mor	22°56'47"	47°18'58"	Monteiro Lobato	22°57'36"	45°50'23"
Morro Agudo	20°43'55"	48°03'30"	Morungaba	22°52'48"	46°47'29"
Motuca	21°30'48"	48°09'05"	Murutinga do Sul	20°59'28"	51°16'33"
Nantes	22°37'13"	51°14'11"	Narandiba	22°24'24"	51°31'29"
Natividade da Serra	23°23'16"	45°26'13"	Nazaré Paulista	23°10'53"	46°24'00"
Neves Paulista	20°46'20"	49°42'35"	Nhandeara	20°41'42"	50°02'30"
Nipoã	20°54'49"	49°46'47"	Nova Aliança	21°00'57"	49°29'50"
Nova Campina	24°07'13"	48°54'13"	Nova Canaã Paulista	20°23'13"	50°56'53"
Nova Castilho	20°45'47"	50°20'34"	Nova Europa	21°46'41"	48°33'39"
Nova Granada	20°31'53"	49°19'05"	Nova Guataporanga	21°19'58"	51°38'40"
Nova Independência	21°06'10"	51°29'33"	Nova Luzitânia	20°51'22"	50°15'45"
Nova Odessa	22°46'38"	47°17'49"	Novais	20°59'30"	48°54'18"
Novo Horizonte	21°28'02"	49°13'18"	Nuporanga	20°43'50"	47°45'00"
Ocauçu	22°26'20"	49°55'25"	Óleo	22°56'30"	49°20'26"
Olímpia	20°44'16"	48°55'02"	Onda Verde	20°36'35"	49°17'35"
Oriente	22°09'08"	50°05'33"	Orindiúva	20°10'54"	49°21'08"
Orlândia	20°43'14"	47°53'12"	Osasco	23°31'52"	46°46'30"
Oscar Bressane	22°19'02"	50°16'54"	Osvaldo Cruz	21°47'37"	50°52'33"
Ourinhos	22°58'28"	49°52'20"	Ouro Verde	21°29'13"	51°42'06"
Ouroeste	20°00'03"	50°22'18"	Pacaembu	21°33'50"	51°15'41"
Palestina	20°23'24"	49°26'05"	Palmares Paulista	21°04'55"	48°48'02"
Palmeira d'Oeste	20°25'00"	50°45'47"	Palmital	22°47'04"	50°13'21"
Panorama	21°21'14"	51°51'49"	Paraguaçu Paulista	22°24'52"	50°34'34"
Paraibuna	23°23'09"	45°39'48"	Paraíso	21°00'57"	48°46'25"
Paranapanema	23°23'14"	48°43'24"	Paranapuã	20°06'15"	50°35'12"
Parapuã	21°46'57"	50°47'34"	Pardinho	23°04'55"	48°22'20"
Pariquera-Açu	24°43'14"	47°52'43"	Parisi	20°18'17"	50°00'46"
Patrocínio Paulista	20°38'26"	47°16'52"	Paulicéia	21°18'53"	51°49'52"
Paulínia	22°45'47"	47°09'07"	Paulistânia	22°34'42"	49°24'08"
Paulo de Faria	20°01'51"	49°24'04"	Pederneiras	22°21'04"	48°46'42"
Pedra Bela	22°47'35"	46°26'32"	Pedranópolis	20°14'55"	50°06'38"
Pedregulho	20°14'55"	47°28'48"	Pedreira	22°44'45"	46°53'51"
Pedrinhas Paulista	22°49'10"	50°47'27"	Pedro de Toledo	24°16'19"	47°14'06"
Penápolis	21°24'59"	50°04'23"	Pereira Barreto	20°38'43"	51°06'35"
Pereiras	23°04'25"	47°58'30"	Peruíbe	24°19'18"	46°59'55"

Piacatu	21°35'27"	50°35'50"	Piedade	23°42'36"	47°25'12"
Pilar do Sul	23°48'44"	47°42'59"	Pindamonhangaba	22°55'33"	45°27'40"
Pindorama	21°11'12"	48°54'20"	Pinhalzinho	22°46'48"	46°35'25"
Piquerobi	21°53'12"	51°43'40"	Piquete	22°36'56"	45°10'42"
Piracaia	23°03'12"	46°21'31"	Piracicaba	22°42'30"	47°38'01"
Piraju	23°11'44"	49°22'54"	Pirajuí	22°00'04"	49°27'24"
Pirangi	21°05'26"	48°39'34"	Pirapora do Bom Jesus	23°23'55"	47°00'05"
Pirapozinho	22°16'26"	51°29'57"	Pirassununga	21°59'52"	47°25'28"
Piratininga	22°24'45"	49°08'00"	Pitangueiras	21°00'37"	48°13'13"
Planalto	21°02'04"	49°55'29"	Platina	22°37'56"	50°12'17"
Poá	23°31'30"	46°20'20"	Poloni	20°47'05"	49°48'50"
Pompéia	22°06'29"	50°10'31"	Pongaí	21°44'10"	49°21'22"
Pontal	21°01'25"	48°02'22"	Pontalinda	20°26'24"	50°31'22"
Pontes Gestal	20°10'58"	49°42'16"	Populina	19°56'43"	50°32'18"
Porangaba	23°10'30"	48°07'20"	Porto Feliz	23°13'02"	47°31'35"
Porto Ferreira	21°51'10"	47°28'48"	Potim	22°50'12"	45°15'09"
Potirendaba	21°02'30"	49°22'32"	Pracinha	21°51'14"	51°05'12"
Pradópolis	21°21'34"	48°04'12"	Praia Grande	24°00'35"	46°24'45"
Pratânia	22°48'31"	48°39'59"	Presidente Alves	22°06'03"	49°26'15"
Presidente Bernardes	22°00'25"	51°33'16"	Presidente Epitácio	21°45'34"	52°06'12"
Presidente Prudente	22°07'04"	51°23'01"	Presidente Venceslau	21°52'19"	51°50'48"
Promissão	21°32'09"	49°51'27"	Quadra	23°17'55"	48°03'17"
Quatá	22°14'48"	50°41'52"	Queiroz	21°47'52"	50°14'30"
Queluz	22°32'23"	44°46'16"	Quintana	22°03'59"	50°18'25"
Rafard	23°00'38"	47°31'41"	Rancharia	22°13'34"	50°53'35"
Redenção da Serra	23°16'05"	45°32'29"	Regente Feijó	22°13'10"	51°18'12"
Reginópolis	21°53'13"	49°13'34"	Registro	24°29'13"	47°50'17"
Restinga	20°36'10"	47°29'00"	Ribeira	24°39'21"	49°00'23"
Ribeirão Bonito	22°03'59"	48°10'42"	Ribeirão Branco	24°13'10"	48°45'56"
Ribeirão Corrente	20°27'25"	47°35'25"	Ribeirão do Sul	22°47'02"	49°55'57"
Ribeirão dos Índios	21°50'18"	51°36'02"	Ribeirão Grande	24°06'01"	48°21'47"
Ribeirão Pires	23°42'55"	46°25'10"	Ribeirão Preto	21°10'30"	47°48'38"
Rifaina	20°04'44"	47°25'20"	Rincão	21°35'20"	48°04'20"
Rinópolis	21°43'26"	50°43'16"	Rio Claro	22°24'22"	47°31'39"
Rio das Pedras	22°50'36"	47°36'15"	Rio Grande da Serra	23°44'40"	46°23'45"
Riolândia	19°58'49"	49°40'58"	Riversul	23°49'28"	49°26'03"
Rosana	22°34'47"	53°03'32"	Roseira	22°53'55"	45°18'18"
Rubiácea	21°17'55"	50°43'50"	Rubinéia	20°11'00"	51°00'49"
Sabino	21°27'32"	49°34'48"	Sagres	21°53'00"	50°57'18"
Sales	21°20'32"	49°30'03"	Sales Oliveira	20°46'18"	47°50'12"
Salesópolis	23°31'55"	45°50'51"	Salmourão	21°37'27"	50°51'34"
Saltinho	22°50'44"	47°40'37"	Salto	23°12'10"	47°17'35"
Salto de Pirapora	23°38'50"	47°34'28"	Salto Grande	22°53'32"	49°59'13"
Sandovalina	22°27'23"	51°45'45"	Santa Adélia	21°14'33"	48°48'22"
Santa Albertina	20°02'01"	50°43'34"	Santa Bárbara d'Oeste	22°45'15"	47°24'58"

Santa Branca	23°23'51"	45°53'08"	Santa Clara d'Oeste	20°05'35"	50°55'45"
Santa Cruz da Conceição	22°08'21"	47°27'05"	Santa Cruz da Esperança	21°17'26"	47°25'43"
Santa Cruz das Palmeiras	21°49'41"	47°14'58"	Santa Cruz do Rio Pardo	22°54'12"	49°37'49"
Santa Ernestina	21°27'46"	48°23'22"	Santa Fé do Sul	20°12'25"	50°55'35"
Santa Gertrudes	22°27'17"	47°31'42"	Santa Isabel	23°19'20"	46°13'49"
Santa Lucia	21°41'03"	48°05'05"	Santa Maria da Serra	22°34'00"	48°09'36"
Santa Mercedes	21°21'05"	51°45'12"	Santa Rita d'Oeste	20°08'40"	50°49'58"
Santa Rita do Passa Quatro	21°42'20"	47°28'42"	Santa Rosa do Viterbo	21°28'46"	47°21'50"
Santa Salete	20°14'41"	50°41'12"	Santana da Ponte Pensa	20°15'15"	50°47'50"
Santana de Parnaíba	23°26'40"	46°55'05"	Santo Anastácio	21°58'17"	51°39'26"
Santo André	23°39'15"	46°31'40"	Santo Antonio da Alegria	21°05'10"	47°09'14"
Santo Antonio de Posse	22°36'05"	46°55'05"	Santo Antonio do Aracanguá	20°56'07"	50°29'41"
Santo Antonio do Jardim	22°06'50"	46°40'55"	Santo Antonio do Pinhal	22°49'29"	45°40'00"
Santo Expedito	21°51'00"	51°23'26"	Santópolis do Aguapeí	21°38'16"	50°30'02"
Santos	23°57'35"	46°19'56"	São Bento do Sapucaí	22°41'20"	45°43'55"
São Bernardo do Campo	23°41'40"	46°33'05"	São Caetano do Sul	23°37'30"	46°33'20"
São Carlos	22°01'20"	47°53'38"	São Francisco	20°21'35"	50°41'50"
São João da Boa Vista	21°58'15"	46°47'40"	São João das Duas Pontes	20°23'19"	50°22'38"
São João de Iracema	20°30'47"	50°21'17"	São João do Pau d' Alho	21°16'10"	51°40'00"
São Joaquim da Barra	20°35'00"	47°51'45"	São José da Bela Vista	20°35'35"	47°38'30"
São José do Barreiro	22°38'36"	44°34'29"	São José do Rio Pardo	21°35'53"	46°53'30"
São José do Rio Preto	20°48'36"	49°22'50"	São José dos Campos	23°11'25"	45°53'03"
São Lourenço da Serra	23°51'02"	46°56'36"	São Luís do Paraitinga	23°13'23"	45°18'38"
São Manuel	22°43'49"	48°34'11"	São Miguel Arcanjo	23°52'37"	47°59'34"
São Paulo	23°32'52"	46°38'07"	São Pedro	22°33'02"	47°55'01"
São Pedro do Turvo	22°45'01"	49°44'25"	São Roque	23°31'48"	47°08'04"
São Sebastião	23°48'12"	45°23'52"	São Sebastião da Grama	21°42'35"	46°49'25"
São Simão	21°28'48"	47°33'11"	São Vicente	23°57'30"	46°23'15"
Sarapuí	23°38'28"	47°49'38"	Sarutaiá	23°16'20"	49°28'48"
Sebastianópolis do Sul	20°39'20"	49°55'15"	Serra Azul	21°18'38"	47°34'00"
Serra Negra	22°36'35"	46°42'05"	Serrana	21°12'30"	47°35'50"
Sertãozinho	21°08'10"	47°59'20"	Sete Barras	24°23'13"	47°55'35"
Severínia	20°48'32"	48°48'15"	Silveiras	22°39'52"	44°51'17"
Socorro	22°35'30"	46°31'37"	Sorocaba	23°29'56"	47°27'30"
Sud Mennucci	20°41'23"	50°55'28"	Sumaré	22°49'13"	47°16'08"
Suzanápolis	20°29'51"	51°01'31"	Suzano	23°32'15"	46°18'40"
Tabapuã	20°57'46"	49°01'48"	Tabatinga	21°44'00"	48°41'06"
Taboão da Serra	23°36'25"	46°45'20"	Taciba	22°23'19"	51°17'03"
Taguaí	23°27'05"	49°24'35"	Taiaçu	21°08'37"	48°30'50"
Taiúva	21°07'26"	48°27'07"	Tambaú	21°42'20"	47°16'16"
Tanabi	20°37'21"	49°38'51"	Tapiraí	23°57'51"	47°30'28"
Tapiratiba	21°28'10"	46°45'00"	Taquaral	21°04'26"	48°24'40"
Taquaritinga	21°24'43"	48°29'54"	Taquarituba	23°31'54"	49°14'42"
Taquarivaí	23°55'19"	48°41'49"	Tarabaí	22°18'12"	51°33'40"
Tarumã	22°44'46"	50°34'32"	Tatuí	23°21'03"	47°50'53"

Taubaté	23°01'35"	45°33'21"	Tejupá	23°20'30"	49°22'32"
Teodoro Sampaio	22°31'52"	52°10'03"	Terra Roxa	20°47'12"	48°19'52"
Tietê	23°06'03"	47°42'55"	Timburi	23°12'11"	49°36'34"
Torre de Pedra	23°14'48"	48°11'41"	Torrinha	22°25'32"	48°10'14"
Trabiju	22°02'28"	48°20'08"	Tremembé	22°57'44"	45°33'16"
Três Fronteiras	20°14'09"	50°53'27"	Tuiuti	22°49'03"	46°41'32"
Tupã	21°56'01"	50°30'49"	Tupi Paulista	21°23'12"	51°34'25"
Turiúba	20°57'00"	50°06'28"	Turmalina	20°03'05"	50°28'32"
Ubarana	21°09'52"	49°43'03"	Ubatuba	23°26'09"	45°04'10"
Ubirajara	22°31'35"	49°39'52"	Uchôa	20°57'05"	49°10'26"
União Paulista	20°53'13"	49°53'50"	Urânia	20°19'39"	50°38'40"
Uru	21°47'02"	49°16'48"	Urupês	21°12'05"	49°17'29"
Valentim Gentil	20°25'12"	50°05'07"	Valinhos	22°58'25"	46°59'50"
Valparaíso	21°13'26"	50°51'41"	Vargem	22°53'20"	46°24'42"
Vargem Grande do Sul	21°50'00"	46°53'35"	Vargem Grande Paulista	23°36'30"	47°10'40"
Várzea Paulista	23°12'47"	46°50'02"	Vera Cruz	22°13'15"	49°49'23"
Vinhedo	23°01'47"	46°58'28"	Viradouro	20°52'18"	48°17'42"
Vista Alegre do Alto	21°10'18"	48°37'43"	Vitória Brasil	20°11'51"	50°29'05"
Votorantim	23°32'30"	47°26'40"	Votuporanga	20°25'06"	49°58'39"
Zacarias	21°03'12"	50°03'04"			

Fonte: Fundação Seade [31]

Referências

[1] M. Loparic. *Uma Aplicação do Método Branch-and-Cut a um Problema de Roteamento de Veículos*. Dissertação (Mestrado em Matemática Aplicada). Instituto de Matemática e Estatística, Universidade de São Paulo, 1996.

[2] J. V. Caixeta-Filho. Modelagem no Sistema Agroindustrial. In J. V. Caixeta-Filho e A. H. Gameiro, organizadores, *Sistemas de gerenciamento de transportes : modelagem matemática*. São Paulo: Atlas, 2001. 125 p. ISBN 85-224-2838-7.

[3] M. C. Goldbarg and H. P. L. Luna. *Otimização Combinatória e Programação Linear: modelos e algoritmos*. Rio de Janeiro: Campus, 2000. 649 p. ISBN 85-352-0541-1.

[4] N. Ziviani. *Projetos de Algoritmos: com implementações em Pascal e C*. São Paulo: Pioneira Thomson Learning, 2002. 267 p. ISBN 85-221-0174-4.

[5] C. Pelizaro. *Avaliação de Desempenho do Algoritmo de um Programa Comercial para Roteirização de Veículos*. Dissertação (Mestrado), Escola de Engenharia de São Carlos, Universidade de São Paulo, 2000. Disponível em: (http://www.teses.usp.br/teses/disponiveis/18/18137/tde-09102001-143129/publico/Pelizaro.pdf). Acesso em: 05/11/2003.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. New York: Freeman, 1979.

[7] H. R. Lewis e C. H. Papadimitriou. *Elementos de Teoria da Computação*. 2. ed. Porto Alegre: Bookman, 2000. 344 p. ISBN 85-7307-534-1.

[8] M. Bellmore and G. L. Nemhauser. The Traveling Salesman Problem: A Survey. *Operations Research*, 16:538-558, 1968.

[9] Y. N. Chong. *Heuristic Algorithms For Routing Problems*. Ph.D. thesis, School of Mathematics and Statistics, Curtin University of Technology, 2001. Disponível em: (http://adt.curtin.edu.au/theses/available/adt-WCU20030702.093906/).
Acesso em: 16/08/2003.

[10] G. Clarke and J. V. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12:568-581, 1964.

[11] S. Lin. Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal*, 44:2245-2269, 1965.

[12] E. Balas and P. Toth. Branch and bound methods. In E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, editors, *The Traveling Salesman Problem*, Wiley, Chichester, UK, 1985.

[13] W. W. Garvin, H. W. Crandall, J. B. John, and R. A. Spellman. Applications of Linear Programming in the Oil Industry. *Management Science*, 3:407-430, 1957.

[14] G. Laporte and F. Semet. Classical Heuristics for the Capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*. Philadelphia: SIAM (Society for Industrial and Applied Mathmatics), 2002. ISBN 0-89871-498-2.

[15] T. J. Gaskell. Bases for Vehicle Fleet Scheduling. *Operational Research Quarterly*, 18:281-295, 1967.

[16] P. Yellow. A Computational Modification to the Savings Method of Vehicle Scheduling. *Operational Research Quarterly*, 21:281-283, 1970.

[17] B. L. Golden, T. L. Magnanti, and H. Q. Nguyen. Implementing Vehicle Routing Algorithms. *Networks*, 7:113-148, 1977.

[18] M. D. Nelson, K. E. Nygard, J. H. Griffin, and W. E. Shreve. Implementation Techniques for the Vehicle Routing Problem. *Computers and Operations Research*, 12:273-283, 1985.

[19] M. Dror and L. Levy. Vehicle Routing Improvement Algorithms: Comparison of a "Greedy" and a Matching Implementation for Inventory Routing. *Computers and Operations Research*, 13:33-45, 1986.

[20] S. Salhi and G. K. Rand. Improvements to Vehicle Routing Heuristics. *Journal of the Operational Research Society*, 38:293-295, 1987.

[21] C. D. J. Waters. A Solution Procedure for the Vehicle-Scheduling Problem Based on Iterative Route Improvement. *Journal of the Operational Research Society*, 38:833-839, 1987.

[22] R. Fahrion and W. Wrede. On a Principle of Chain-exchange for Vehicle-routeing Problems (1-VRP). *Journal of the Operational Research Society*, 41:821-827, 1990.

[23] R. H. Mole and S. R. Jameson. A Sequential Route-building Algorithm Employing a Generalised Savings Criterion. *Operational Research Quarterly*, 27:503-511, 1976.

[24] B. E. Gillett and L. R. Miller. A Heuristic Algorithm for the Vehicle-Dispatch Problem. *Operations Research*, 22:340-349, 1974.

[25] A. Wren and A. Holliday. Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points. *Operational Research Quarterly*, 23:333-344, 1972.

[26] M. Gendreau, G. Laporte and J.-Y.Potvin. Metaheuristics for the Capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*. Philadelphia: SIAM (Society for Industrial and Applied Mathmatics), 2002. ISBN 0-89871-498-2.

[27] M. Gendreau, A. Hertz and G. Laporte. A Tabu Search Heuristic for the Vehicle Routing Problem. *Management Science*, 40:1276-1290, 1994.

[28] V. M. M. Pureza. *Problemas de Roteamento de Veículos Via Metaheurística Tabu*. Dissertação (Mestrado). Faculdade de Engenharia Elétrica, Universidade Estadual de Campinas, 1990.

[29] E. L. Lawler, J. K. Lenstra, A. H.G. Rinnooy Kan and D. B. Shmoys, editors, *The Traveling Salesman Problem: A guided tour of Combinatorial Optimization*, Wiley, Chichester, UK, 1985.

[30] IBGE - Instituto Brasileiro de Geografia e Estatística. Ministério do Planejamento, Orçamento e Gestão - BR. Disponível em: (http://www.ibge.gov.br/). Acesso em 28 Out. 2003.

[31] SEADE - Fundação Sistema Estadual de Análise de Dados. Secretaria de Estado dos Negócios de Economia e Planejamento - SP. Disponível em: (http://www.seade.gov.br/). Acesso em 28 Out. 2003.

[32] DER – Departamento de Estradas de Rodagem. Secretaria dos Transportes - SP. Disponível em: (http://www.der.sp.gov.br/). Acesso em 19 Dez. 2003.