UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HAROLDO ALEXANDRE DE ARAUJO

ALGORITMO *SIMULATED ANNEALING*: UMA NOVA ABORDAGEM

Dissertação submetida à Universidade
Federal de Santa Catarina como parte
dos requisitos para a obtenção do grau de
Mestre em Ciência da Computação

Prof. José Mazzucco Junior, Dr.
Orientador

Florianópolis, Maio de 2001

Algoritmo *Simulated Annealing*: uma nova abordagem

Haroldo Alexandre de Araújo

Esta dissertação foi julgada adequada para a obtenção do Título de

Mestre em Ciência da Computação

Área de Concentração: Sistemas de Computação, e aprovada em sua forma final pelo Curso de Pós-Graduação em Ciência da Computação – CPGCC

_____

Prof. Fernando A. Osthuni Gauthier, Dr. – Coordenador

Banca Examinadora

_____

Prof. José Mazzucco Junior, Dr. – Orientador

_____

Prof. Luis Fernando Friedrich, Dr. – INE, UFSC

_____

Prof. Thadeu Botteri Corso, Dr. – INE, UFSC

Dedico este trabalho a
Bernardo e Rodolfo Nejur meus filhos;
Hugo e Raifa meus pais;
Claudete minha esposa e a seus pais
Germano (in memoriam) e Luiza.

AGRADECIMENTOS

A DEUS por tudo que me é proporcionado todos os dias e por ter permitido a conclusão deste trabalho.

Ao Prof. Dr. José Mazzucco Junior, pela orientação, mas também pela paciência, compreensão e amizade que demonstrou nas diversas fases de elaboração deste trabalho, virtudes dos verdadeiros mestres.

À Claudete, minha esposa, presença importante na minha vida que muito contribuiu, com seu carinho, seu apoio, seu incentivo constante e, principalmente, por ter sabido compreender minha ausência.

Aos meus filhos Rodolfo Nejur e Bernardo presenças indispensáveis na minha vida, que entenderam minha ausência.

Aos meus pais Hugo e Raifa que apesar do pouco conhecimento acadêmico souberam nos encaminhar para o estudo.

Aos colegas de turma, os professores Laura, Luis Carlos, Wilker e Valmir pelos bons momentos de convivência durante a primeira fase desta jornada, e ao Valmir também pelas sugestões e contribuições feitas.

A Cleusa e Gláucia pelo suporte desde o inicio da caminhada.

Aos meus irmãos, incluindo os aderentes, pelo apoio quase que silencioso, muito me ajudaram.

Aos meus amigos, que me apoiaram na realização deste trabalho, especialmente aqueles que contribuíram diretamente para o mesmo.

Aos professores e funcionários do DME-UFAC pelo apoio constante recebido.

À Universidade Federal do Acre que através do Departamento de Matemática e Estatística pôde me proporcionar tais estudos.

RESUMO

A busca por soluções de problemas por meio do computador é o tema central da ciência da computação, relevante para grande parte da ciência e de suas aplicações tecnológicas. Essa busca, certamente, vai na direção de algoritmos eficientes e exatos mas que nem sempre boas soluções podem ser encontradas para muitos problemas de ordem prática, principalmente, no que diz respeito a tempo de execução. Existem problemas, dentre estes, os de otimização combinatorial que apresentam uma peculiaridade com relação aos outros, que é a grande dificuldade de se obter soluções exatas num tempo computacional aceitável.

Atualmente, as novas técnicas, especialmente as metaheurísticas, tais como: *Tabu Search*, *Simulated Annealing*, Algoritmos Genéticos e Redes Neurais, vêm conseguindo sucesso na solução de problemas de otimização combinatorial, que mesmo não apresentando soluções exatas têm mostrado bastante eficiência com suas soluções aproximadas.

Este trabalho propõe um novo método baseado no algoritmo *Simulated Annealing* (SA) através de mudanças bruscas nos valores da temperatura que são retiradas de múltiplas faixas, ao contrário do SA básico, onde esses valores são obtidos de uma faixa única, ou seja, num SA básico, os valores assumidos pela temperatura saem de um intervalo, partindo de um valor inicial, e vão diminuindo até um valor final. Tais mudanças bruscas acontecem exatamente no momento da mudança de faixa, pois o valor da temperatura que no final de uma faixa é pequeno, assume um valor correspondente a temperatura inicial da faixa seguinte, normalmente, bem maior.

Posto a prova, com instâncias euclidianas do Problema Caixeiro Viajante, que é um problema de otimização combinatorial de difícil solução, o método apresenta resultados bastante satisfatórios quando comparado com o SA básico.

ABSTRACT

The fetching for solutions of problems by means of the computer is the central subject of the computer science, relevant for great part of science and its technological applications. This fetching, certainly, goes in the direction of efficient and exact algorithms, but that nor always-good solutions can be found for many problems of practical order, mainly, in that it says respect the execution time. Problems exist, amongst these, of optimization combinatorial that present a peculiarity with relation to the others, which is the great difficulty of obtaining exact solutions in an acceptable computational time.

Currently, the new techniques, especially the metaheuristics, such as: Tabu Search, Simulated Annealing, Algorithms Genetic and Neural Networks, comes obtaining success in the solution of combinatorial optimization problems, that even not presenting exact solutions, have shown sufficiently efficiency with its approximate solutions.

This work considers a new method based on the algorithm Simulated Annealing (SA) through brusque changes in the values of the temperature that are removed of multiple bands, in contrast of the basic SA, where these values are gotten of an only band, that is to say, in a basic SA, the values assumed for the temperature leave an interval, starting with an initial value, that goes diminishing until a final value. Such brusque changes happen in the exact moment of the band change, therefore the value of the temperature that in the end of a band is small, assume a corresponding value of the initial temperature of the following band, normally, a lot bigger.

Put in test, with Euclidean instances of the Traveling Salesman Problem, that is a combinatorial optimization problem of difficult solution, the method presents resulted sufficiently satisfactory when comparative with the basic SA.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

CAPÍTULO 1

Introdução

A competitividade e a dinamicidade do mercado obrigam as empresas a se tornarem cada vez mais ágeis e ajustadas às constantes mudanças que o mercado impõe. A empresa que não conseguir gerenciar de forma eficiente todos seus recursos poderá estar perdendo mercado, com produtos/serviços de baixa qualidade, e nos piores casos, tendo prejuízo. A grande demanda por produtos/serviços de qualidade, tem levado as empresas a uma frenética busca por produtividade. Aumentar a produção sem aumentar custo é a meta principal de qualquer organização que vise o lucro. E isto tem ficado cada vez mais evidente, principalmente com a globalização, quando a concorrência tem ficado ainda mais acirrada.

A diversidade de recursos que uma empresa manipula é muito grande sendo, portanto, necessária uma análise criteriosa das diversas e possíveis combinações dos mesmos. De posse desses recursos, as empresas que prestam serviços ou fabricam produtos devem encontrar a melhor forma de usá-lo e/ou combiná-lo visando, além da qualidade, ainda atender determinados conjuntos de restrições. Por exemplo, qual o melhor itinerário que um caminhão de coleta de lixo deve fazer para que o trajeto percorrido seja mínimo, atendendo ainda os requisitos, capacidade do caminhão e quantidade de lixo em cada ponto coletor?; Qual a melhor combinação (mão-de-obra, matéria-prima, máquinas, etc) para que vários produtos sejam produzidas e que seus custos seja mínimo?; Qual a rota que o movimento do braço de um robô, que realiza milhares de pontos de solda, deve fazer para que seu deslocamento seja mínimo?. Estes são apenas três exemplos de problemas que podem ser encontrados no dia-a-dia das empresas e que envolvem processos de combinação

e otimização ao mesmo tempo, sendo, portanto, denominados problemas de otimização combinatorial.

Áreas como ciência dos computadores, estatística, engenharia, ciência da administração, ciência biológica, projeto de circuito (VSLI), entre outras, propuseram nas últimas décadas vários problemas que são inerentemente de otimização combinatorial. De todos os problemas propostos um dos mais conhecidos é o Problema do Caixeiro Viajante (PCV). Neste problema um vendedor (caixeiro), partindo de uma cidade, precisa visitar cada uma das outras cidades predefinidas exatamente uma única vez e então retornar à cidade de origem. O problema consiste em definir a seqüência de visitas que corresponda ao comprimento (custo) mínimo. Como se pode notar o PCV é um problema de fácil entendimento e estrutura simples, porém com alto grau de dificuldade de resolução à medida que o número de cidades cresce. Esta característica é inerente à maioria dos problemas de otimização combinatorial.

Resolver um problema de otimização combinatorial consiste em encontrar a “melhor” solução ou a solução “ótima” dentre um número finito ou um número infinito contábil de soluções possíveis que atenda a um objetivo específico. Tais problemas apresentam uma peculiaridade com relação aos outros, que é a grande dificuldade de obtenção de soluções exatas num tempo computacional aceitável.

Nas últimas décadas, com o desenvolvimento teórico da ciência dos computadores, várias conclusões foram tiradas em relação aos problemas de otimização combinatorial, sendo uma delas com relação à classificação. Verificou-se que cada problema apresentava uma complexidade particular, cujo esforço necessário para resolvê-lo através de algum algoritmo exato, era exponencial em relação ao tamanho do mesmo. Nesta classe de problemas ou existirá um algoritmo de complexidade polinomial capaz de resolver todos os problemas desta classe, ou então nenhum destes poderá ser resolvido em tempo polinomial. Com os resultados apresentados pelos estudos, os problemas ditos NP-completos passaram a ser de grande interesse, principalmente, depois que se constatou que a maioria dos problemas de otimização combinatorial pertence a essa classe de problemas.

O problema do caixeiro viajante (PCV) é um dos clássicos pertencente à classe dos problemas NP-Completos e, portanto um paradigma para teste de novos algoritmos. Assim,

o fato de certo problema ser identificado como NP-Completo é aceito com forte evidência contra a existência de algoritmos polinomiais e, por conseguinte, como justificativa para a utilização de heurísticas, combinadas ou não a técnicas de enumeração ou a outras técnicas de otimização, com o objetivo de obter soluções aproximadas e de boa qualidade.

Desta forma muita ênfase vem sendo dada no desenvolvimento de heurísticas que forneçam soluções de boa qualidade em tempo de processamento compatível com as necessidades das empresas. As novas técnicas, especialmente as metaheurísticas, tais como: *Tabu search, Simulated Annealing,* Computação Evolucionária (Algoritmos Genéticos, Programação Evolucionária, etc.) e Redes Neurais, são de muita importância para solução destes problemas de otimização combinatorial, e têm se mostrados bastantes eficientes em muitos problemas.

Infelizmente, ainda é pequeno o número de empresas que utilizam em seus processos alguma técnica de otimização. Isto se deve, principalmente, a uma falta de conhecimento a respeito do poder real de tais técnicas.

Durante muitos anos, as teorias e métodos desenvolvidos por matemáticos e cientistas foram arquivados em livros e periódicos especializados e muito pouco foi absorvido pelo setor empresarial. Felizmente, contudo, essa situação vem se alterando. É cada vez maior o número de companhias que adotam modelos de otimização no seu dia-a-dia, diminuindo seus custos e, por conseguinte, aumentando os lucros. Além do mais, com a onda crescente de privatizações nos diversos setores da sociedade, a concorrência se fortifica e a sobrevivência dos negócios começa a depender seriamente do desempenho de cada um com relação aos demais. Quem estiver mais bem preparado irá, sem dúvida alguma, suplantar os adversários. No Brasil, esse processo vem ganhando força, mas ainda é incipiente. Nos demais países, principalmente na Europa e nos Estados Unidos, a utilização de técnicas de otimização dentro das empresas é bem mais difundida, assumindo um papel de grande relevância.

Os sistemas de gerenciamento atuais, baseados no princípio da logística, requerem cada vez mais técnicas que propiciem uma melhor utilização dos recursos disponíveis, ocasionando uma redução nos custos logísticos, sendo que os novos métodos, são uma ferramenta importante que possibilitará o alcance deste objetivo.

Atualmente, existe um grande número de abordagens que combinam técnicas de otimização local com os novos métodos de aproximação que, normalmente, utilizam como base os algoritmos *Tabu Search*, Genético e *Simulated Annealing*, que mesmo não apresentando soluções exatas têm mostrado resultados bastante eficientes. Com estruturas bastante simples e de fácil implementação, estas abordagens vêm ganhado terreno em detrimento dos algoritmos exatos. Estas estruturas mais complexas que fazem uso de modelos matemáticos (baseado em proposições e teoremas) da programação linear através dos métodos *branch-bound e cutting-planes* e que ainda incorporam características do problema tratado, porém com resultados considerados satisfatórios, apesar do excessivo tempo computacional consumido, mesmo para instâncias consideradas pequenas.

O extraordinário sucesso alcançado por estes algoritmos e a utilização dos mesmos nas soluções de problemas gerais de otimização e, especialmente, nos combinatoriais, tem levado a intensas pesquisas. Isto se deve especialmente, a diversos fatores, dentre eles: a utilização de mecanismos de otimização modelados na imitação de processos naturais; aplicabilidade geral; flexibilidade nas adaptações às restrições específicas em casos reais; à excelente relação entre qualidade e facilidade de implementação e principalmente, tempo de processamento.

Aliando o poder computacional disponível hoje às modernas técnicas desenvolvidas, ainda não foi possível encontrar solução eficiente para um grande número de problemas, principalmente os de otimização combinatorial. Para esses, a melhor saída é se contentar com soluções aproximadas através de técnicas heurísticas.

1.1 Objetivo do trabalho

O objetivo deste trabalho é avaliar o comportamento de um método que tem como base à abordagem *Simulated Annealing* (SA), aplicado ao clássico Problema do Caixeiro Viajante que é um problema NP-completo, através de testes computacionais com instâncias públicas disponíveis via internet e instâncias construídas, também com suas soluções, *a priori*, conhecidas.

Especificamente, pretende-se fazer uma analise comparativa da performance do algoritmo gerado pelo método, que tem os valores da temperatura retirados de múltiplas faixas, em relação a um outro que tem sua implementação básica baseado no *Simulated Annealing*, isto é, os valores da temperatura são retirados de uma única faixa.

1.2 Justificativa do trabalho

A demanda crescente por aplicações em situações reais tem levado o aparecimento de métodos para solucionar instâncias para o Problema do Caixeiro Viajante com número de cidades cada vez maiores. Por tratar-se de um problema de otimização, qualquer novidade tanto em tempo de processamento, recorde em número de cidades e em soluções ótimas significa, em termos de aplicações reais, redução de custo, o que é sempre bem vindo.

Apesar de existir bastante esforço por parte dos cientistas em propor modelos matemáticos que solucionem de forma exata, problemas de otimização combinatorial, como o Problema do Caixeiro Viajante, seus tempos de processamento, para instâncias grandes, ainda é proibitivo. Entretanto, novas técnicas da computação evolutiva, tais como *Simulated Annealing*, *Tabu Search* e Genético, têm se apresentado como parte de alternativas bastante otimistas nas soluções de tais problemas, mesmo considerando resultados aproximados.

Diante disso vários estudos têm indicado que abordagens baseadas nas novas técnicas, tal como *Simulated Annealing* que combina as vantagens das técnicas de melhoramento iterativo com técnicas de randomização, resultando num poderoso instrumento de otimização que vem sendo utilizado em diversos campos como VLSI CAD, processamento de imagem e pesquisa operacional. Pode-se ainda destacar sua aplicabilidade geral para grande variedade de problemas de otimização combinatorial, produzindo de forma consistente soluções aproximadas de alta qualidade para estes problemas.

1.3 Limitações da pesquisa

Mesmo nos artigos que apresentam algoritmos relativamente eficientes para resolução do PCV, como em MARTIN & OTTO (1995) e HELSGAUN (2000), não existe uma descrição mais detalhada que possa servir de orientação para o desenvolvimento de novos algoritmos. Isso impossibilita um trabalho mais abrangente na tentativa de comparação direta dos métodos que se mostram mais eficientes.

Não é pretensão deste trabalho apresentar nenhum recorde nem em termos de número de cidades, reduzido tempo de processamento ou solução ótima, mesmo porque, para instâncias com um número muito grande de cidades o tempo de processamento pode levar dias ou até meses com um parque computacional muito mais sofisticado do que o que será utilizado aqui. Esta avaliação se dará sobre o desempenho do método aplicado ao clássico Problema do Caixeiro Viajante.

A investigação será feita de forma comparativa sobre o desempenho do método que tem como base à abordagem *Simulated Annealing*, sendo que um algoritmo terá sua temperatura retirada sobre uma única faixa de temperatura, que seria o SA básico, enquanto que o outro, que é o método proposto, terá sua temperatura retirada de múltiplas faixas.

1.4 Organização do trabalho

O presente trabalho é apresentado em seis capítulos, mais a bibliografia empregada e os anexos.

No capítulo 2 é apresentada uma fundamentação teórica sobre a complexidade dos problemas no qual o problema do caixeiro viajante se encontra inserido. Também fazem parte desse capítulo alguns conceitos sobre os principais procedimentos básicos utilizados na solução de problemas dessa classe.

No capítulo 3 é introduzida a fundamentação conceitual da abordagem *Simulated Annealing*, onde os principais elementos básicos desse método são revistos.

No capítulo 4 são feitas algumas considerações sobre duas das abordagens atuais para solucionar instâncias do Problema do Caixeiro Viajante, que reforçam as argumentações do modelo proposto, assim como as alterações introduzidas no algoritmo e sua implementação.

No capítulo 5 é apresentada a metodologia utilizada para obtenção dos resultados, bem como as argumentações sobre decisões e ajustes de parâmetros. Também faz parte desse capítulo uma análise comparativa entre os resultados computacionais obtidos com o método proposto e o outro sugerido na literatura.

Finalmente, no capítulo 6, são apresentadas as considerações finais e algumas sugestões que podem ser exploradas em futuras pesquisas.

CAPÍTULO 2

A Complexidade dos Problemas e a do Caixeiro Viajante

Neste capítulo é apresentada uma formulação do problema a ser estudado. Inicialmente uma fundamentação teórica sobre os problemas e suas complexidades é abordada. Posteriormente, é inserido o problema do caixeiro viajante dentro deste contexto. Em seguida são apresentados alguns conceitos básicos sobre os principais procedimentos utilizados que apresentam melhor desempenho na solução de problemas dessa classe.

2.1 Complexidade dos problemas

2.1.1 – Introdução

A ciência tem se deparado com uma grande quantidade de problemas, para os quais ainda não é possível apresentar uma solução satisfatória. A busca de solução para problemas de elevado nível de complexidade computacional tem sido um desafio constante para pesquisadores das mais diversas áreas. Particularmente em otimização, pesquisa operacional, ciência da computação, matemática e engenharias, que têm se defrontado freqüentemente com problemas altamente combinatórios, cuja solução ótima em muitos casos ainda está limitada somente a pequenas instâncias.

Segundo LEWIS & PAPADIMITRIOU (2000), em termos computacionais, os problemas podem ser classificados em duas categorias: aqueles que podem ser resolvidos

por algoritmos e aqueles que não podem. Com tanto poder computacional disponível hoje, haveria de se supor que todos os problemas da primeira classe pudessem ser resolvidos de uma maneira satisfatória.

Infelizmente, a prática da computação revela que muitos problemas, apesar de solúveis a princípio, não podem ser resolvidos em qualquer sentido prático por computadores, mesmo pelos mais velozes ou por combinação desses (em redes, em paralelo, etc.), devido às excessivas exigências de tempo e/ou de espaço de memória.

Os problemas que não podem ser resolvidos são aqueles para os quais a teoria atual da ciência da computação não consegue produzir um algoritmo. É desnecessário dizer que as técnicas tradicionais disponíveis são insuficientes para resolver esse tipo de problema. Como problema não computável podemos citar a própria matemática, “uma vez que já foi provado que não se pode criar um algoritmo que gere as deduções para todas as verdades da matemática” (LEWIS & PAPADIMITRIOU, 2000).

Procurando dar respostas concretas sobre as grandes incertezas que pairam sobre os problemas de elevado nível de complexidade, a ciência da computação tem conseguido, nos últimos anos, apontar várias alternativas no sentido de possibilitar que pesquisadores e cientistas da computação disponham de fundamentação teórica suficientes para decidir se para um determinado problema é possível desenvolver um algoritmo que forneça uma solução exata em tempo aceitável ou se deve contentar-se apenas com solução aproximada.

A ciência da computação tem estudado, como se vê mais na frente, um grande número de problemas considerados como de difícil solução e tem procurado fazer, dentre os já conhecidos, uma classificação de modo a não permitir que determinados esforços sejam direcionados a procura de um algoritmo que forneça uma solução exata para um problema quando isso não for possível.

Computacionalmente, a solução para qualquer problema é um algoritmo ou uma composição de algoritmos (ROUTO, 1991). Doravante o termo complexidade estará relacionado com algoritmo, o que não necessariamente é sempre verdadeiro já que também é possível analisar complexidade em termos de linguagens que estão associadas com problemas.

Portanto uma análise de complexidade é fundamental no processo de definição de algoritmos mais eficientes para a solução de um problema. Apesar de parecer contraditório, com o aumento da velocidade dos computadores, torna-se cada vez mais importante desenvolver algoritmos mais eficientes, devido ao aumento constante do "tamanho" dos problemas a serem resolvidos. No mundo real nada vem a custo zero. Assim, é raro encontrar soluções para problemas de porte que estejam no ponto ótimo de menor tempo e espaço. Invariavelmente é de praxe aceitar compromissos, colocando um peso maior sobre um ou outro desses componentes.

A complexidade computacional de algoritmo diz respeito aos recursos computacionais – espaço de memória e tempo de máquina – necessários para se chegar a uma solução para o problema. Como o tempo de processamento de um algoritmo varia de um computador para outro, sua complexidade não é medida em termos reais, ou seja, não se analisa algoritmo, por exemplo, em termos de tempo real de execução. Esse tipo de avaliação não é adequado, pois em geral existem muitos fatores influenciando o resultado, tais como eficiência do computador, compilador, recursos da linguagem, sistema operacional, entre outros. Embora o espaço de memória seja também passível de análise, a complexidade aqui vai ser considerada apenas em termos de tempo.

Naturalmente, é possível determinar o tempo de um algoritmo através de métodos empíricos, isto é, obter o tempo de execução por meio da execução propriamente dita do algoritmo, considerando-se entradas diversas. Entretanto, é possível obter uma ordem de grandeza do tempo de execução através de métodos analíticos. O objetivo desses métodos é determinar uma expressão (função) matemática que traduza o comportamento do tempo do algoritmo que, ao contrário do método empírico, visa aferir o tempo de execução de forma independente do ambiente utilizado.

Para um dado problema, a complexidade é uma função que relaciona o tamanho de uma instância ao tempo necessário para resolvê-la. Essa função normalmente expressa a quantidade das operações elementares realizadas. A medida da complexidade é então o crescimento assintótico dessa contagem de operações. Por exemplo, num algoritmo para achar o elemento máximo entre n objetos, a operação elementar seria a comparação das grandezas dos objetos, a operação elementar seria a comparação das grandezas efetuadas

pelo algoritmo, para valores grandes de n. A noção de complexidade de tempo é descrita a seguir (ROUTO, 1991).

Seja A um algoritmo, $\{E_1, ..., E_m\}$ o conjunto de todas as entradas possíveis de A. Denote por t_i, o número de passos efetuados por A, quando a entrada for E_i . Definem-se

$$\text{complexidade do } \textit{pior caso} = \max\ E_i \in E\ \{ t_i \},$$
$$\text{complexidade do } \textit{melhor caso} = \min\ E_i \in E\ \{ t_i \},$$
$$\text{complexidade do } \textit{caso médio} = \sum_{1 = i = m} p_i \cdot t_i ,$$

onde p_i é a probabilidade de ocorrência da entrada E_i.

A complexidade de tempo de pior caso corresponde ao número de passos que o algoritmo efetua no seu pior caso de execução, isto é, para a entrada mais desfavorável. De certa forma, a complexidade de pior caso é a mais importante das três mencionadas. Ela fornece um limite superior para o número de passos que o algoritmo pode efetuar, em qualquer caso. Por isso mesmo é a mais empregada, o que faz com que o termo complexidade se refira à complexidade do pior caso.

2.1.1 Eficiência dos algoritmos

A eficiência de algoritmo não tem nenhuma relação direta com o rápido avanço tecnológico experimentados pelos computadores, mas com a quantidade de operações elementares que serão realizadas. Essas operações logicamente dependem do tamanho da entrada do problema. Ordenar um bilhão de números é bem diferente de ordenar 10, a idéia é expressar o número de operações elementares como uma função que caracterize o volume do trabalho que será feito. Para um algoritmo de ordenação, esse número é simplesmente o número n de elementos a serem ordenados.

É comum expressar a complexidade de um algoritmo através da ordem de grandeza da função que o modela. Quando se expressa a complexidade através de sua ordem de grandeza, pode-se desprezar constantes aditivas ou multiplicativas. Por exemplo, um valor

de número de passos igual a 3n será aproximado por n. Além disso, como o interesse é restrito a valores assintóticos, termos de menor grau também podem ser desprezados. Assim, um valor de número de passos igual a $n^2 + n$ será aproximado para n^2. O valor de $6n^3 + 4n - 9$ será transformado em n^3.

Uma notação comum para simbolizar essa ordem de grandeza é a letra O, seguida por uma função entre parênteses que representa a ordem de grandeza. Veja a tabela 2.1 que representa algumas ordens de grandezas de funções conhecidas.

Função	Ordem de Grandeza
Constante	O(1)
Logarítmica	O(log n)
Linear	O(n)
Quadrática	$O(n^2)$
Cúbica	$O(n^3)$
Polinomial	$O(n^c)$, c real
Exponencial	$O(c^n)$, c real e > 1
Fatorial	O(n!)

Tabela 2.1 - Ordens de grandeza de algumas funções

Para se entender melhor a ordem de grandeza de uma função de complexidade de um problema, vamos introduzir a seguinte definição:

> **Definição:** Dizemos que um algoritmo tem ordem de grandeza ou complexidade O(f(n)) quando o número de operações elementares executadas para obter a solução à instância do problema de tamanho n não exceder uma constante vezes f(n), para n suficientemente grande, ou seja, dizemos que g(n) é O(f(n)) se existirem duas constantes K e n' tal que $|g(n)| = K\,|f(n))|$, para todo n = n'.

A ordem de grandeza das funções é importante na análise de complexidade de algoritmos. Ao invés de computarmos a função exata do número de operações necessário,

f(n), é mais fácil e freqüentemente válido trabalhar com a sua ordem de grandeza. É claro que para uma mesma função f(n) pode existir uma infinidade de funções limitantes superiores; mas o que se procura é a menor função limitante superior para caracterizar a sua ordem de grandeza da complexidade.

> **Definição:** Diz que um algoritmo tem complexidade de tempo polinomial se sua função de complexidade g(n) é um polinômio ou se é limitada por outra função polinomial.

Algoritmos com complexidade polinomial são aceitos como de origem de problemas solúveis na prática e, portanto algoritmos computacionalmente viáveis LEWIS & PAPADIMITRIOU (2000). Infelizmente nem todos os algoritmos têm comportamento suave e de solução previsível como os polinomiais, existe um grande número de problemas para os quais seus algoritmos não são limitados por um polinômio, suas taxas de crescimento são explosivas à medida que n cresce, o que tornam esses algoritmos inviáveis e que nem sempre contém fatores exponenciais, como é o caso da função fatorial.

> **Definição:** Diz que um algoritmo tem complexidade de tempo exponencial quando sua função de complexidade g(n) não é limitada por uma função polinomial.

Apesar de parecerem bem comportadas, algumas funções da Tabela 2.1 que representam as respectivas complexidades, podem apresentar surpresas, e por vezes desagradáveis, quando o tamanho de n fica suficientemente grande. Na tabela 2.2 pode observar as diferenças na rapidez de crescimento de várias funções típicas de complexidade. Os valores expressam os tempos de execução quando uma operação elementar no algoritmo é executável em um décimo de microssegundo.

	Valores para n		
Função	20	40	60
n	0,0002 s	0,0004 s	0,0006 s
n log n	0,0009 s	0,0021 s	0,0035 s
n^2	0,004 s	0,016 s	0,036 s
n^3	0,08 s	0,64 s	2,26 s
2^n	10 s	127 dias	3.660 séculos
3^n	580 min	38.550 séculos	$1,3 \times 10^{14}$ séculos

Tabela 2.2 - Comparação de várias funções de complexidade
Fonte ROUTO (1991).

Como se pode observar na tabela 2.2, o crescimento extremamente rápido das funções de complexidade exponenciais inviabiliza qualquer algoritmo de encontrar uma solução quando n cresce, independentemente da velocidade do computador que o execute. Devido ao desempenho negativo dos algoritmos de complexidade exponencial, os cientistas de computação consideram tais algoritmos inaceitavelmente ineficientes. Ao contrário, os algoritmos de complexidade polinomial são considerados eficientes.

2.1.3 – Classificação dos problemas

A grande descoberta da ciência da computação relacionada à complexidade computacional, que estuda a eficiência dos algoritmos e as dificuldades inerentes aos problemas de importância práticas e/ou teóricas, foi que o esforço requerido para resolver precisamente um problema no computador pode variar tremendamente. Provar a existência ou não de algoritmos que solucione problema de elevado nível de complexidade computacional tem sido um desafio constante para os pesquisadores de diversas áreas.

A grande contribuição da ciência da computação foi a de propor métodos matemáticos que irão ajudar a provar que um problema de interesse pertence ou não a uma determinada classe. Segundo LEWIS & PAPADIMITRIOU (2000), isso tem poupado os cientistas de tentativas inúteis e fadadas ao fracasso.

Teoricamente é produtivo pensar em um problema como associado a uma linguagem, ou seja, uma mesma coisa sendo tratada em dois aspectos diferentes. Linguagens são mais apropriadas em conexão à máquina de Turing, enquanto problemas são instruções mais claras de práticas computacionais como tarefas de interesse prático, para as quais devemos desenvolver algoritmos. E é esta associação que tem levado aos grandes avanços conseguidos pela ciência da computação em relação à complexidade dos problemas.

A princípio, a grande maioria dos problemas pode ser resolvida por um algoritmo em tempo polinomial deste que não se faça exigência ao tipo de máquina que irá executa-lo. Pode até parecer uma heresia essa afirmação, uma vez que ainda não se tem definido precisamente quais problemas pertencem a que classe de complexidade. A idéia de que o algoritmo pode ser executado em tempo polinomial vem da relação existente entre os problemas de decisão e os algoritmos, através da teoria das linguagens, onde sempre é possível exibir um algoritmo que o resolve.

Na teoria das linguagens existem modelos matemáticos com características, por exemplo, não-determinísticas, que não podem ser executados por máquinas reais. Isso significa dizer que algoritmos com características não-determinísticas podem resolver a grande maioria dos problemas, inclusive aqueles que apresentam complexidade com crescimento exponencial. O fato é que existem máquinas reais e máquinas que não são reais, que existem apenas a nível teórico. As máquinas reais, como os computadores, só podem executar algoritmos, onde os passos são determinados, mais precisamente, os computadores só podem executar algoritmos determinísticos, ao contrário, dos modelos, como a máquina de Turing, que podem também executar algoritmos não-determinísticos. Embora a relação seja entre algoritmos e problemas de decisão, e que nem todo problema é um problema de decisão, é possível modelar um problema que não seja de decisão como tal, sem com isto perder sua característica de complexidade.

Um grande número de problemas é viável computacionalmente, ou seja, existem algoritmos que são capazes de resolve-lo, no entanto, nem todo o problema se pode chegar a uma solução em um tempo razoável. Existem problemas com ordem de grandeza exponencial que, em termos computacionais, são intratáveis. Um problema é tratável se for possível exibir um algoritmo de complexidade polinomial que o resolva. Por outro lado,

para verificar se um problema é intratável, há a necessidade de se provar que todo possível algoritmo que o resolva não possui complexidade polinomial.

A computação limitada polinomialmente é um conceito atraente e produtivo para uma teoria elegante e útil, já que inclui a maioria dos algoritmos viáveis e exclui a maioria dos impraticáveis. Essa fronteira entre os algoritmos viáveis e os impraticáveis ainda é bastante cinzenta, uma vez que é possível encontrar algoritmos polinomiais que são impraticáveis em termos práticos em função de seus índices elevados, mas que sem sempre ocorrem na prática. Não ficando, portanto prejudicada a definição para a classe.

> **Definição**: A classe P é formada pelo conjunto de todos os problemas que pode ser resolvido por um algoritmo determinístico em tempo polinomial.

Apesar de ainda existir algumas argumentações contestando a definição acima, parece ser esta, segundo LEWIS & PAPADIMITRIOU (2000), a única proposta séria da teoria da computação. O determinismo neste contexto significa que a cada passo de execução só há um determinado comando a ser executado. Mas o melhor modo de familiarizar-se com os tempos de computação polinomiais, seu escopo real e suas limitações é introduzir uma variedade de interessantes problemas computacionais que são conhecidos como pertencentes a P, sem, no entanto, se esquecer de encontrar exemplos de problemas que não parecem estar em P. Muitas vezes, os problemas difíceis e os fáceis são muito similares.

Há uma classe de problemas que apesar dos prolongados e intensos esforços dos matemáticos e cientistas da computação para descobrir um algoritmo de tempo polinomial para cada um desses problemas, nenhum algoritmo desse tipo foi encontrado. Esses problemas são, em sua grande maioria, os chamados problemas de decisão para os quais todos os algoritmos conhecidos são de complexidade exponencial (LEWIS & PAPADIMITRIOU, 2000). Entretanto, por outro lado**,** não se conhecem provas, até o momento, de que o fato seja comum a todos os possíveis algoritmos para esses problemas. Existem problemas que apesar de pertencerem a essa classe não são problemas de decisão, como os de otimização, mas que é possível através de mecanismos adequados transforma-los em tal.

Antes de definir a classe NP, descreve-se a noção de não-determinismo. Diz-se que um algoritmo é não determinístico se além de todas as regras de um algoritmo determinístico, ele pode fazer escolhas de forma não determinística. Informalmente, pode-se definir a classe NP em termos de um algoritmo não determinístico. Tal algoritmo é dividido em duas partes. A primeira parte corresponde à escolha da provável solução (podendo fazer uso de escolhas não determinísticas). A outra parte consiste de verificar de forma determinística a viabilidade de uma solução.

> **Definição**: A classe NP é formada pelo conjunto de todos os problemas que pode ser resolvido por um algoritmo não-determinístico em tempo polinomial.

Obviamente, todo problema que admite um algoritmo de rapidez polinomial pertence a NP, isto é, $P \subseteq NP$, pois todo algoritmo determinístico é equivalente a um não-determinístico, onde uma outra possível alternativa não existe. Outro entendimento para a classe NP é a de que ela consiste de todos os problemas de decisão, para os quais existe uma justificativa à resposta SIM, cujo passo de verificação pode ser realizado por um algoritmo polinomial.

Não se sabe se P = NP, embora a maioria dos pesquisadores acredite que esta igualdade não seja válida. Esta é talvez a maior conjectura na área de ciência da computação. Uma das maiores razões do porque se acredita que $P \neq NP$, é a existência da classe de problemas NP-completo. Um problema é NP-completo se é NP e se existe uma redução polinomial de qualquer problema NP para este problema. Veja mais sobre redução polinomial em LEWIS & PAPADIMITRIOU (2000).

Dessa maneira, se existir um algoritmo polinomial para um problema NP-completo, então todos os problemas de NP podem ser resolvidos polinomialmente(e consequentemente, nessa hipótese, P = NP). Parece intrigante a existência de tal conjunto, mas de fato, COOK (1971), mostrou que o problema da satisfatibilidade (SAT) pertence à classe NP-completo. Pode-se dizer sem rigor que P é a subclasse de "menor dificuldade" de NP, e que a subclasse de "maior dificuldade" de NP é a classe dos NP-completos. A figura 2.1 ilustra este relacionamento.

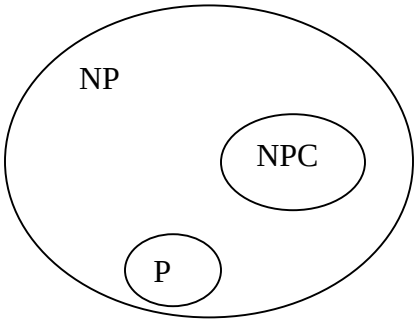

Figura 2.1 - Relacionamento entre P, NP e NPC

2.2 Alguns conceitos básicos

Antes de formular uma definição e os métodos básicos de solução para o problema do caixeiro viajante, é conveniente apresentar algumas conceituações básicas relacionadas ao escopo da formulação e da resolução do mesmo.

2.2.1 Espaço de busca e seus métodos

Em geral, problemas do mundo real que são modelados como problemas de otimização combinatorial apresentam grande dificuldade em encontrar soluções. Esta dificuldade está relacionada ao grande número de possíveis soluções, que no caso do PCV, chega a número astronômico, impossível de enumerá-las. Em termos técnicos, o conjunto de todas as possíveis soluções constitui o *espaço de busca*. Normalmente sua notação é feita por $S \subseteq \Re^n$. A solução para um determinado problema pode ser encontrar um ponto ou um conjunto de pontos onde a função $f : S \rightarrow \Re$ tem valores máximos ou mínimos, que em termos de um problema de busca, pode ser escrito:

$$\text{encontrar } x^* \mid f(x^*) = f(x), \text{ para todo } x \in S, \text{ no caso de mínimo.}^1$$

[1] Problemas de minimização podem ser convertidos em problemas de maximização e vice-versa, através de artifícios bem simples como, por exemplo, multiplicar a função por -1.

A topologia de um *espaço de busca* S pode ser contínuo ou discreto, finito ou infinito, côncavo ou convexo, que determina a aplicação desse ou daquele método, bem como as características da função f que pode ser derivável ou não, unidimensional ou multidimensional, além de estacionária ou não-estacionária (variável com o tempo). Quando um problema apresenta somente um ponto de máximo ou de mínimo no *espaço de busca*, este é chamado *unimodal*, caso contrário *multimodal*.

Existem três métodos, segundo TANOMARU (1995), que podem ser utilizados para pesquisar nesses espaços: numéricos, enumerativos e métodos probabilísticos, todos com seus respectivos derivados e ainda um número grande de métodos híbridos. Os *métodos numéricos* são bastante utilizados e apresentam soluções exatas em muitos tipos de problemas, entretanto são ineficientes para otimização de funções multimodais. Além disso, em problemas de otimização combinatorial em espaços discretos, geralmente a função a ser otimizada é não somente multimodal, mas também não diferenciável e/ou não contínua. *Métodos enumerativos* examinam cada ponto do *espaço de busca*, um por um, em busca de pontos ótimos. A idéia pode ser intuitiva, mas é obviamente impraticável quando há um número infinito ou extremamente grande de pontos a examinar. Com o advento da computação evolutiva, os *métodos probabilísticos* vêm ganhando espaços e estão presentes nas abordagens mais modernas. Este método emprega a idéia de busca probabilística, o que não quer dizer que sejam métodos totalmente baseados em sorte, como é o caso dos chamados métodos aleatórios. Dentre as abordagens mais conhecidas pode-se citar os algoritmos *Simulated Annealing* e genético.

2.2.2 Otimização combinatorial

Uma grande parte dos problemas científicos pode ser formulado como problemas de otimização combinatorial. Por exemplo, considere um sistema onde uma série de fatores influencia seu desempenho. Tais fatores podem assumir um número limitado ou ilimitado de valores, e podem ser sujeitos a certas restrições. O objetivo é encontrar a melhor combinação (*combinatorial*) dos fatores, ou seja, a combinação de fatores que proporcione

o melhor desempenho (*otimização*) possível para o sistema. Um problema de Otimização Combinatorial pode ser um problema de minimização ou de maximização e é especificado por um conjunto de instâncias do problema.

Uma instância é definida pelo par (S,f), onde o espaço S denota o conjunto finito de todas as soluções possíveis e f a função custo que é um mapeamento definido por $f : S \rightarrow \Re$. No caso de minimização, o problema é encontrar uma solução $x_{opt} \in S$ que satisfaz $f(x_{opt}) = f(x)$, para todo $x \in S$. No caso de maximização, o problema é encontrar uma solução $x_{opt} \in S$ que satisfaz $f(x_{opt}) \geq f(x)$, para todo $x \in S$. A solução x_{opt} é chamada uma solução ótima global, mínima ou máxima e $f_{opt} = f(x_{opt})$ denota o custo ótimo. Podem existir mais que uma solução com custo ótimo, definindo dessa forma S_{opt} como o conjunto de soluções ótimas .

Não é difícil perceber que existe uma dualidade entre os conceitos de *espaço de busca e otimização combinatorial*, de tal modo que todo problema de busca pode ser considerado um problema de otimização e vice-versa. Portanto, as soluções para tais problemas podem utilizar os mesmos métodos já mencionados na secção 2.2.1, onde os métodos exatos são baseados em resultados de teoremas e proposições. Estes utilizam conceitos de programação matemática (programação linear – inteira) e têm como meta obter uma solução ótima de um problema. Os outros métodos, chamados de aproximados, cujo objetivo é o de encontrar soluções próximas a uma solução ótima, mas num tempo computacional reduzido quando comparado como os métodos exatos.

2.2.3 Pesquisa local e vizinhança

Segundo GU & HUANG (1994), *pesquisa local*, também referenciada como *neighborhood search ou Hill climbing* é uma das técnicas mais eficientes para solucionar problemas de otimização combinatorial e base para muitos métodos heurísticos. Isoladamente é uma técnica iterativa simples para encontrar boas soluções. Pesquisa local foi uma das primeiras técnicas propostas durante os meados dos anos 60 para enfrentar com relativo sucesso a intratabilidade computacional dos problemas de otimização

combinatorial de difícil solução. Devido à irregularidade[2] no espaço de pesquisa a mesma freqüentemente fica preso a configurações localmente ótimas.

Considerando-se as definições e notações anteriores (seções 2.2.1 e 2.2.2) sobre espaço de busca e a função objetiva, pesquisa local consiste em mover-se de uma solução para outra, na vizinhança dessa de acordo com algumas regras bem definidas. Dada uma solução corrente $x_i \in S$ e a cada passo i, uma nova solução x_{i+1} é escolhida na vizinhança $N(x_i)$ da solução corrente x_i, de tal modo que, no caso de minimização, $f(x_{i+1}) = f(x_i)$. O processo se repete e x_{i+1} passa a ser a solução corrente, se essa solução existir, caso contrário x_i é a solução ótima local.

Para aplicar a técnica de pesquisa local a um problema particular, o procedimento é bastante simples, segundo GU & HUANG (1994), basta apenas especificar a estrutura da vizinhança, definir a função objetiva e um procedimento para obter uma possível solução inicial. Especificar a estrutura de *vizinhança* é fundamental para qualquer método que utilize pesquisa local. Definir uma vizinhança adequada para um dado problema é certamente o fator mais crítico para o sucesso de qualquer tipo de pesquisa local. A vizinhança de uma solução *x (N(x))* contém todas as soluções que podem ser alcançadas de x por um único movimento. Aqui, o significado de movimento é aquele no qual um operador transforma uma solução em outra através de algum mecanismo de perturbação.

No caso do PCV, por exemplo, duas possíveis rotas são consideradas como vizinhas se uma puder ser derivado da outra via um número definido de mudanças de arcos. Existem diferentes esquemas de mudanças de arcos, tais como troca de cidades (*city-swap*), r-opt e or-opt. Estes esquemas afetam a estrutura da vizinhança no espaço de pesquisa com o surgimento de terrenos[3] com superfícies diferentes, e como conseqüência produz diferentes números de soluções de mínimos locais.

Recentes estudos sobre métodos de pesquisa local para o PCV têm feito maior uso do poder da estrutura de vizinhança. GUTIN & YEO (1998) usando a vizinhança de PUNNEN (1996) construíram vizinhanças de tamanho exponencial polinomialmente alcançável

[2] Considerando o espaço de pesquisa como a superfície de um terreno, irregularidade significa a presença de locais planos, locais com cumes e locais com depressões.
[3] Superfície de terreno é uma analogia com o espaço de busca, já que este também possui, segundos estudos, planícies, vales e cumes.

utilizando grafos para representá-las. Segundo eles, quando todas vizinhanças são polinomialmente alcançáveis, o gráfico correspondente é polinomialmente alcançável. O diâmetro do gráfico da vizinhança é uma característica importante da estrutura da vizinhança no esquema de pesquisa local correspondente. Claramente, vizinhança com gráfico de diâmetro pequeno parece mais potente que aqueles com gráfico de diâmetro grande, ficando, logicamente, de fora aqueles cujos gráficos têm diâmetro infinito. Segundo GUTIN & YEO (1998), a principal propriedade de uma vizinhança *N(t)* imposta usualmente é quando a melhor entre todas as rotas da vizinhança possa ser alcançada em tempo polinomial.

PAPADIMITRIOU & STEIGLITZ (1982) *apud* LEWIS & PAPADIMITRIOU (2000) também fazem algumas considerações sobre estruturas de vizinhança, segundo eles se o número de vizinhos para uma dada solução é muito grande, a qualidade de uma solução especifica não diz muito sobre a qualidade de seus vizinhos e assim o processo de pesquisa não aprende para move-se para regiões mais promissoras do espaço de pesquisa. Por outro lado, também argumentam que se o número de vizinhos for muito pequeno, a relação de vizinhança não será exata. Exatidão, neste caso, significa que de qualquer solução s_i há pelos menos um caminho (bilateral) finito de soluções vizinhas [x_i, x_{i+1}, x_{i+2},..., x_{opt}] para uma solução globalmente ótima x_{opt} que vêm com uma seqüência monotonicamente decrescente dos valores da função objetiva [$f(x_i)$, $f(x_{i+1})$, $f(x_{i+2})$,..., $f(x_{opt})$]. Se isso não for o caso, uma pesquisa de melhoramento local pode ficar presa numa solução localmente ótima, sem obter o ótimo global.

2.2.4 Noções sobre heurística

O estudo da heurística tem sua inspiração na observação de como pessoas executam tarefas com esta simples e duvidosa informação conhecida como *intuição*. A ciência da heurística realiza estudos teóricos e empíricos para entender o funcionamento heurístico: como ele é adquirido, armazenado e usado pelas pessoas, como ele pode ser representado e utilizado por máquinas e ainda, o que faz com que certas heurísticas obtenham sucesso e

outras falhem. Dentro deste contexto, são os sistemas inteligentes que mais têm se beneficiados desses avanços.

REEVES (1995) *apud* JOHNSON & MCGEOCH (1997) define heurística (quando aplicada a problema de otimização combinatorial) como uma técnica que procura boas (próximas à ótima) soluções a um razoável custo computacional, sem ser capaz de garantir a viabilidade do ótimo, ou mesmo, em muitos casos, quão próxima do ótimo está uma particular solução viável encontrada.

Na própria definição percebe-se os dois parâmetros básicos, e conflitantes, de um método heurístico: a qualidade da solução gerada e o esforço computacional despendido. E é exatamente na possibilidade de minimização deste conflito que se encontra a causa para a explosão de interesse neste campo de estudo.

O desenvolvimento de conceitos de complexidade computacional e a classificação de problemas difíceis, para os quais não parece existir algoritmo em tempo polinomial, têm fundamentado a utilização de técnicas heurísticas, sem causar grande constrangimento à não garantia da solução ótima. Por outro lado, a disponibilidade de recursos computacionais mais poderosos tem propiciado muita pesquisa e conseqüente desenvolvimento de métodos heurísticos muito eficientes, o que tem lhes conferido um considerável respeito.

Problemas, em geral, como o do Caixeiro Viajante requerem a avaliação de um número imenso de possibilidades para determinar uma solução exata. Em muitos casos, o tempo requerido para esta avaliação pode ultrapassar anos. Heurísticas têm o papel de indicar uma maneira de reduzir o número de possibilidades e obter uma solução num tempo razoável.

As heurísticas costumam ser procedimentos mais simples e flexíveis que os métodos exatos de solução, permitindo-lhes então abordar modelos mais complexos. Este é outro argumento a seu favor, tendo em vista que, com freqüência, os métodos exatos exigem simplificações nos modelos, tornando-os menos representativos do problema real.

A avaliação de desempenho de heurísticas no pior caso ou no caso médio é quase sempre impraticável. Além do mais, este tipo geral de análise não fornece informações precisas sobre performance em instâncias particulares do problema. Por isso mesmo, uma das metodologias mais utilizadas para avaliação de heurísticas é a experimentação empírica

sobre problemas testes. Entretanto, para garantir a significância dos resultados é preciso ter critérios e parâmetros bem estabelecidos.

2.3 O problema do caixeiro viajante

2.3.1 Introdução

Desde que foi citado pela primeira vez por DANTZIG (1954) *apud* JOHNSON & MCGEOCH (1997) o qual relatava uma solução para 49 cidades através de métodos de programação linear, que o Problema do Caixeiro Viajante ou PCV (em inglês TSP - *the Traveling Salesman Problem*) tem se mantido no topo das pesquisas dos problemas de otimização combinatorial, e que hoje pode ser considerado como sendo *o mais estudado de todos os tempos*[4].

O estudo desse problema tem atraído muitos pesquisadores de diferentes áreas, como por exemplo, Matemática, Física, Biologia, Inteligência Artificial e a existência de uma vasta literatura sobre este problema.

A importância do PCV vem do fato de que é um representante de uma classe muito vasta de outros problemas de otimização combinatorial e contém dois ingredientes necessários para atrair qualquer cientista: simplicidade na formulação e difícil solução. Este problema serviu e continua servindo como referência no desenvolvimento, teste e demonstração de novas técnicas de otimização.

Por outro lado, o Problema do Caixeiro Viajante é interessante não somente sobre o ponto de vista teórico. Muitas aplicações práticas, que vão desde a fabricação de *chips*

[4] Citação feita pelos organizadores do 8th DIMACS Implementation Challenge: The Traveling Salesman Problem realizado em setembro de 2000.

VLSI (KORTE, 1988) *apud* JOHNSON & MCGEOCH (1997) até cristalografia de raio-X[5] (BLAND & SHALLCROSS, 1989) *apud* JOHNSON & MCGEOCH (1997), podem ser modeladas como o PCV ou como suas variações e portanto há uma necessidade de algoritmos. O número de cidades em aplicações práticas pode variar de algumas dezenas até alguns milhões. LAWLER et al. (1985) fazem excelentes abordagens sobre uma vasta coleção de publicações sobre este problema.

Desde que GAREY & JOHNSON (1979) mostraram que o PCV é um problema NP-completo e que, portanto, qualquer algoritmo para encontrar rotas ótimas tem seu tempo de complexidade aumentando mais rápido que qualquer polinômio (desde que $P \neq NP$), que poucas alternativas têm restado aos pesquisadores. Ou procurar heurística que simplesmente encontre soluções com rotas sub-ótimas, mas fazendo isso rapidamente, ou tentar desenvolver algoritmos de otimização que trabalhe bem com instâncias do mundo real, ao invés de com instâncias com o pior caso.

O mais famoso problema de otimização combinatorial tem sido nos últimos anos a base para teste de numerosas técnicas das mais variadas origens. Décadas de pesquisas em técnicas de otimização, combinado com o contínuo e rápido aumento na velocidade dos computadores e na capacidade de memória, tem levado um novo recorde atrás do outro. Nos últimos quinze anos, o registro de solução ótima para instância não trivial do PCV tem aumentado de 318 cidades (CROWDER & PADBERG, 1980), para 2.392 cidades (PADBERG & RINALDI, 1987) *apud* JOHNSON & MCGEOCH (1997), para 7.397 cidades (APPLEGATE et al., 1994) *apud* JOHNSON & MCGEOCH (1997), e mais recentemente para 13.509 cidades (APPLEGATE et al., 1998).

2.3.2 Formulação do PCV

O mais proeminente membro do rico conjunto de problemas de otimização combinatorial que mais tem atraído estudiosos de todos os tempos é sem dúvida o Problema

[5] Traduzido X-ray crystallography, ciência que se ocupa dos cristais, suas formas e estruturas – definição do dicionário do Aurélio – editora Nova Fronteira.

do Caixeiro Viajante que pode ser formulado como: Um caixeiro deve visitar **n** cidades diferentes iniciando e terminando o seu percurso na mesma cidade, não importando a ordem na qual as cidades são visitadas.

Considerando que todas as cidades são interligadas, o problema consiste em se determinar uma rota que torne mínima a viagem total, ou seja, achar uma seqüência correta de cidades a visitar de modo que essa seqüência tenha o menor custo possível. Essa rota mínima ou melhor solução é geralmente definida por algum critério específico, como por exemplo, a distância. A distância pode ter, dependendo do contexto, outras denominações tais como, tempo ou dinheiro. Neste trabalho será usado o termo *custo* para representar tal situação.

Para formular uma notação que represente o número R(n) de rotas para o caso de **n** cidades basta seguir a seguinte lógica: Considerando que o Caixeiro parte de uma cidade determinada, portanto a primeira não afeta no cálculo, a próxima escolhida deve se retirada do conjunto das (n –1) cidades restantes. A seguinte, obviamente, do conjunto das (n-2) cidades restantes e, assim sucessivamente. Dessa forma, através de um raciocínio combinatorial simples e clássico, concluí-se que o conjunto de rotas possíveis, do qual o Caixeiro deve escolher a menor, possui cardinalidade dada por:

$$(n - 1) \times (n - 2) \times (n - 3) \times \ldots \times 2 \times 1$$

ou seja, o Caixeiro deve escolher sua melhor rota de um conjunto de (n –1)! possibilidades.

Seguindo uma lógica prática, por exemplo, no caso de n = 4 cidades (A, B, C e D), a primeira e última posição é fixa, de modo que elas não afetam o cálculo; na segunda posição pode-se colocar qualquer uma das 3 cidades restantes B, C e D, e uma vez escolhida uma delas, pode-se colocar qualquer uma das 2 restantes na terceira posição; na quarta posição não se tem nenhuma escolha, pois sobrou apenas uma cidade; conseqüentemente, o número de rotas é 3 x 2 x 1 = 6, o que corresponde a 3!. Considerando que não importa o sentido que se percorre o roteiro, o número total de rotas fica diminuído

pela metade, ou seja 3 rotas. De modo que, por dedução simples pode-se fazer a seguinte notação fatorial: R(n) = (n - 1)!/2.

2.3.3 A complexidade do PCV

A primeira vista, como exemplificado acima, pode-se pensar que se trata de um problema, simples que basta se achar todas as possíveis rotas e, usando um computador, calcular o custo de cada uma delas e então ver qual a de menor custo, e com isso se estaria reduzindo-o um problema de otimização a um de enumeração. Entretanto, essa estratégia reducionista (força bruta), aparentemente viável, não é recomendável para a maioria dos casos, principalmente quando o número de cidades é muito grande.

Suponha-se que se tem um computador muito veloz, capaz de fazer um bilhão de adições por segundo. Isso parece uma velocidade imensa, capaz de tudo. Com efeito, no caso de 20 cidades, o computador precisa apenas de 19 adições para dizer qual o comprimento de uma rota, logo será capaz de calcular 10^9 / 19 = 53 milhões de rotas por segundo. Contudo, essa astronômica velocidade é um nada frente à imensidão do (19! / 2) de rotas que precisará examinar. Com efeito, o valor aproximado desse número é $6{,}0 \times 10^{16}$ em notação científica. Conseqüentemente, o computador precisaria de

$$6{,}0 \times 10^{16} / 53{,}0 \times 10^{6} = 1{,}13 \times 10^{9} \text{ segundos}$$

para completar sua tarefa, o que equivale à cerca de 36 anos, aproximadamente.

Observa-se que o aumento no valor do n provoca uma muito lenta diminuição na velocidade com que o computador calcula o tempo de cada rota (ela diminui apenas de um sexto quando n aumenta de 5 para 25), entretanto provoca um aumento imensamente grande no tempo total de calculo. Em outras palavras: a inviabilidade computacional é devida à presença da fatorial na medida do esforço computacional do método. Com efeito, se essa complexidade fosse expressa em termos de um polinômio em n, como por exemplo,

n^5, o computador proposto acima seria perfeitamente capaz de suportar o aumento de n, conforme mostra a tabela 2.3.

n	Rotas/s	(n – 1)! / 2	cálculo total	n^5	cálculo total
5	250 milhões	12	insignificante	3.125	insignificante
10	110 milhões	181.440	0,0015 seg.	100.000	Insignificante
15	71 milhões	$4{,}35 \times 10^8$	10 min.	759.375	0,01 seg.
20	53 milhões	$6{,}0 \times 10^{16}$	36 anos	3.200.000	0,06 seg.
25	42 milhões	$6{,}2 \times 10^{23}$	235×10^6anos	9.765.625	0,23 seg.

Tabela 2.3[6] - Esforço computacional

2.3.4 Procedimentos básicos

Por ser considerado um problema de difícil solução (NP-completo), o PCV tem despertado interesse de estudiosos das mais diversas áreas. A princípio qualquer problema difícil pode ser resolvido tanto método que apresente solução exata quando não exata, dependendo da conveniência. Várias soluções têm sido propostas para ambos os métodos.

Dentre os resultados de métodos exatos, salientam-se o *Branch and Bound* de MILLER & PENKY (1991) *apud* LAPORTE (1992) rodando instâncias de ATSP com até 500.000 cidades, enquanto o *Branch and Bound* de CARPANETO & TOTH (1980) *apud* LAPORTE (1992) apresentam resultados para instâncias de ATSP de até 1000 cidades. Fischetti e Toth (1997) *apud* JOHNSON & MCGEOCH (1997) apresentaram um *Branch and Cut* rodando instâncias de ATSP com até 100 cidades geradas aleatoriamente e todas as instâncias da TSPLIB entre 34 e 171 cidades. Para o caso simétrico, APPLEGATE et al. (1998) apresentaram resultados com um *Branch and Cut* para algumas instâncias da TSPLIB, dentre as quais, a maior instância da TSPLIB, correspondendo as 13.509 cidades dos Estados Unidos com mais de 500 habitantes.

[6] Os dados da terceira coluna foram alterados, já que na original era computado apenas (n-1)!.

Apesar de todo poder computacional disponível atualmente e, dos grandes avanços conseguidos, os métodos exatos ainda se apresentam bastantes proibitivos para grandes instâncias[7]. O que tem levado a procura por métodos não exatos, que apesar de não garantir a solução ótima, garante boas soluções aproximadas. Estes algoritmos são usualmente muito simples e têm (relativamente) pequeno tempo de processamento. Apresentam solução que diferem em média somente por pouco ponto percentual da solução ótima. Portanto, para problemas onde pequenas diferenças do ótimo podem ser aceitas, os algoritmos aproximados são métodos bastante apropriados.

As abordagens aproximadas para o PCV podem atuar sobre uma solução de duas maneiras, construindo-a passo a passo uma possível solução ou tenta sistematicamente melhorar uma solução existente. Para a primeira categoria, o conjunto de heurísticas é chamado de procedimentos de construção de rotas, enquanto que para a segunda categoria, tem-se procedimentos de melhoramento de rotas. Em geral, abordagens que combinam os dois procedimentos têm sido bastante recomendadas.

2.3.4.1 Procedimentos de construção de rotas

Um algoritmo para construção de rota inicia-se com uma sub-rota e tenta expandi-la numa rota completa que seja aproximadamente ótima. Esta expansão se dar gradualmente pela adição de uma cidade a cada passo. Usualmente, a sub-rota inicial é simplesmente uma escolha randômica de uma cidade ou ao acaso. LAWLER et al. (1985) dão alguns ingredientes considerados chaves para procedimentos de construção de rotas.

JOHNSON & MCGEOCH (1997) fazem uma análise de quatro importantes algoritmos utilizados em heurísticas para construção de rotas. Para todos, é apresentado uma descrição formal, além de uma amostra de seu desempenho tanto do ponto de vista teórico como em termos prático. Embora não apresente melhor performance, a heurística,

[7] O tempo para calcular a instância com 13.509 cidades levou três meses para se conseguir a solução ótima pelo método exato.

Vizinho mais Próximo, é a mais simples de ser implementada, portanto a mais utilizada como forma de obter uma solução inicial.

a) Vizinho mais próximo (*Neaest Neighbor*)

Talvez a heurística mais natural para PCV. Ela imita um viajante, cuja regra prática é a de ir sempre para a cidade mais próxima que ainda não foi visitada. Partindo de uma cidade qualquer c_i, visita-se a cidade c_k que é a mais próxima da cidade c_i, ou seja, a de menor custo. Depois, estende-se o roteiro visitando a cidade mais próxima de c_k. Continua-se assim até que todas cidades tenham sido visitadas, retornando-se então à cidade c_i. A complexidade deste procedimento é $O(n^2)$. Uma possível modificação é ter como ponto de partida todas as cidades. A complexidade aumenta ($O(n^3)$), mas em compensação resulta, geralmente numa rota melhor.

b) Algoritmos de inserção

Existem várias heurísticas baseadas em critérios de inserção. Dado uma sub-rota, uma distinção é feita para selecionar qual cidade deve ser inserida na sub-rota corrente e onde a cidade deve ser inserida. Enquanto tais decisões podem ser feitas ao mesmo tempo (procedimento de inserção mais barato), vários outros da mesma família, empregam critérios diferentes para decidir *qual* versus *onde*. A escolha dos critérios de seleção e inserção pode ser crítica para o sucesso de tal procedimento (LAPORTE, 1992). Dependendo do critério que é usado, a complexidade desses tipos de procedimentos podem variar entre $O(n^2)$ e $O(n \log n)$.

2.3.4.2 Procedimentos de melhoramento de rotas

Os procedimentos de melhoramento de rotas, usualmente iniciam-se com uma rota completa e tentam reduzir o custo da mesma de tal modo que a rota se mantenha completa. Uma heurística freqüentemente usada nos procedimentos de melhoramento de rotas é a mudança de arcos ou de links. Essas técnicas são as mais simples e as mais comuns para esses tipos de procedimentos.

Um método bastante direto nos procedimentos de melhoramento é a troca (*swap*) entre duas cidades na rota. Se a troca reduz o custo total da rota, ela é simplesmente mantida e essa nova rota é o novo ponto de partida para o próximo melhoramento, caso contrário, tenta-se uma nova troca. Este método é chamado de heurística de troca de cidades. Apesar de simples e direto, empiricamente ele é muito menos efetivo que outro método de melhoramento de rota.

A melhor e mais efetiva heurística de mudança de arcos conhecida para melhoramento de rota aplicada ao PCV é aquela que usa o conceito *r-optimal* (ou simplesmente r-opt) popularizado por LIN (1965).

> **Definição:** Uma rota é dita ser *r-optimal* (ou *r-opt*) se for impossível obter uma outra rota com custo menor trocando qualquer conjunto de r links entre cidades por um outro conjunto de r links diferentes.

Como se pode ver *r-opt* como heurística de melhoramento de rota é também um procedimento interativo, onde em cada iteração r mudanças de arcos são efetuadas. Ambas as heurísticas (*swap e r-opt*) podem ser vista como um processo simples de pesquisa em vizinhança, onde cada rota tem uma vizinhança associada que são as rotas adjacentes, que podem ser alcançadas numa única operação uma vizinha melhor e, continuamente até que não exista nenhuma vizinha melhor.

Os movimentos *r-opt* são básicos para os principais procedimentos de pesquisa local para o PCV. Diversos valores para r para a heurística *r-opt* foram testados[8]. Entre os algoritmos de pesquisa local mais simples, as heurísticas *2-opt e 3-opt* estão entre as mais utilizadas. Cada heurística define uma estrutura de vizinhança a qual pode ser pesquisada por diferentes esquemas de pesquisa na vizinhança (VOUDOURIS & TSANG, 1999).

i) Heurística 2-opt

Embora o conceito *r-opt* tenha sido apresentado por LIN (1965), o algoritmo *2-opt* foi primeiro proposto por CROES (1958) *apud* JOHNSON & MCGEOCH (1997) depois

[8] CHRISTOFIDES & EILON (1972) implementaram sem sucesso este método com r = 4 e 5 *apud* LAPORTE (1992).

que o movimento básico já tinha sido sugerido por FLOOD (1956) *apud* JOHNSON & MCGEOCH (1997). Uma solução vizinha é obtida da solução corrente pela remoção de dois arcos da rota e reconectando-os de modo que o caminho resultante fique invertido, veja figura 2.2. A complexidade do pior caso para uma pesquisa na vizinhança definida por *2-opt* é $O(n^2)$.

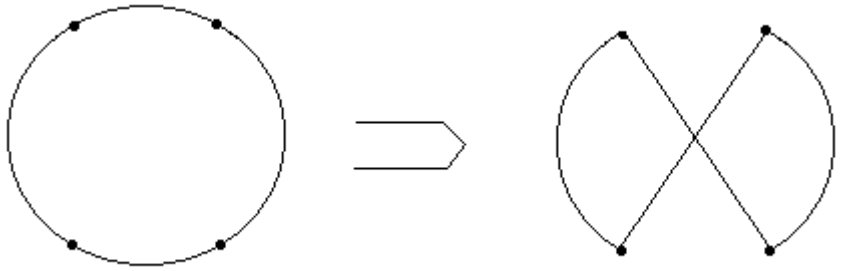

Figura 2.2 - Movimento *2-opt*

ii) Heurística 3-opt

Utilizando por LIN (1965), agora a solução vizinha é obtida pela remoção de três arcos. Os três caminhos resultantes são colocados juntos de uma outra forma, possibilitando a inversão de um ou mais desses caminhos, conforme pode ser visto na figura 2.3. As heurísticas *3-opt*, segundo VOUDOURIS & TSANG (1999), são muito mais efetivas que as *2-opt*, muito embora o tamanho da vizinhança seja maior e com isso consumir mais tempo de pesquisa. A complexidade do pior caso para uma pesquisa na vizinhança definida por *3-opt* é $O(n^3)$.

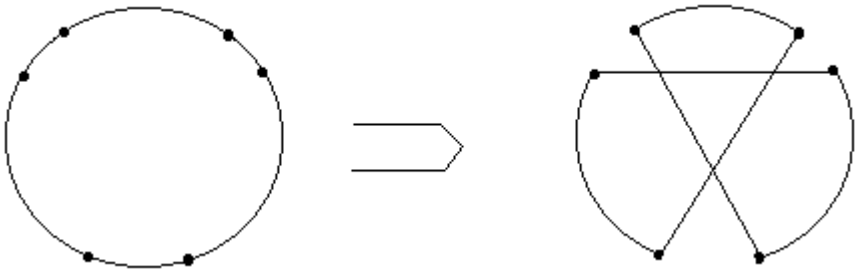

Figura 2.3 - Um possível movimento *3-opt*

2.3.5 Avaliação de performance

Avaliar a performance é uma questão relevante para qualquer modelo que utilize algoritmos aproximados. Segundo LAWLER et al. (1985), existem três maneiras de avaliar a performance de uma heurística.

A primeira forma é estritamente empírica, *analise empírica*, a heurística é aplicada a um conjunto de problemas testes e os valores das soluções obtidas são comparados aos valores das soluções ótimas, ou a valores limites, ou ainda, a valores produzidos por outras heurísticas. A dificuldade com esta abordagem está na geração de problemas testes corretos, no sentido de ser representativo e na interpretação estatística correta dos resultados. Todavia, por muitos anos, foi esta a única abordagem disponível. Foi com ela que LIN & KERNIGHAN (1973) apresentaram sua heurística que tinha como atrativo a combinação com solução de alta qualidade e tempo de processamento baixo.

A segunda maneira, *analise do pior caso*, tornou-se popular no final dos anos 60, a idéia é obter uma heurística que produza solução afastada o quanto possível da solução ótima, ou seja, no pior limite possível e inventar instâncias para problemas ruins para os quais a heurística realmente alcance esse desvio. Tal análise fornece a garantia da performance que mesmo sendo pessimista pelos menos nunca será violada. A dificuldade obvia da análise do pior caso pode ser baseado na sua fraca performance para instâncias de problemas patológicos ou com número infinito de cidades. Entretanto, evidências empíricas dão conta que tal comportamento ruim é extremamente raro (LAWLER et al.,1985).

Ao contrário dos outros métodos onde suas propriedades são estabelecidas empiricamente, simplesmente por métodos de tentativas e observando-se a qualidade dos resultados, a *análise probabilística* é uma abordagem teórica, na qual assume que as instâncias do problema são projetadas de uma distribuição probabilística simples e então provada matematicamente que o método heurístico para uma solução particular tem alta chance de produzir soluções próximas da ótima quando o número de cidades é grande. A principal dificuldade obvia encontrada pela abordagem é que ela pressupõe a especificação de uma distribuição probabilística sobre o conjunto de todas as instâncias do problema em

questão. O problema de decidir se tal distribuição é razoável, é comparável ao problema de encontrar um conjunto de problemas teste correto.

2.3.6 Aplicações

O PCV é sedutivelmente fácil de formular (LAWLER et al., 1985). Ele não exige nenhum fundamento matemático para entendê-lo e nenhum grande talento para encontrar boas soluções para problemas grandes, o que torna o trabalho de seus estudiosos quase que um divertimento recreativo.

Por outro lado, o PCV tem resistido a todos os esforços de encontrar bons algoritmos de otimização ou ainda de algoritmos aproximados que tenha garantia de serem efetivos. Desse modo, o PCV tem dois elementos que tem atraído matemáticos para problemas particulares por séculos: simplicidade na formulação e dificuldade na solução. Por tudo isso, não há razão para acreditar que essa atração irá diminuir daqui para frente.

Além das de pesquisas, há também razões práticas para a importância do PCV. Muitos problemas relevantes do mundo real podem ser formulados como instâncias dele. A despeito do fato de que o modelo do caixeiro viajante, onde um viajante deseja minimizar sua distância viajada, é aplicado diretamente a diversas situações práticas, muitas das aplicações relatadas a seguir são bastante diferentes, isto é, problemas aparentemente sem relação são resolvidos através de formulações com instâncias do PCV. Descreve-se abaixo algumas aplicações que são modeladas pelo PCV para que se possa ter uma idéia da versatilidade do mesmo.

2.3.6.1 Roteamento de veículo

Em geral, o problema de roteamento de veículo consiste em determinar para uma frota de carros quais clientes seriam atendidos por quais veículos e qual a ordem que cada veículo visitaria seus clientes. Outros problemas podem ser derivados dessa idéia.

a) Distribuição de Bens de Consumo. Dado um conjunto de fregueses que precisam receber mercadorias, a fábrica tem que decidir a quantidade de carga a ser colocada em cada caminhão e quais caminhões irá atender quais clientes. Além disso, é preciso otimizar as rotas dos veículos e, em alguns casos, levar em consideração a eventual necessidade de reabastecimento da carga por parte de alguns caminhões. Por exemplo, os clientes podem ser bares e a fábrica pode ser uma fábrica de bebidas.

b) Localização de Facilidades. Dado um conjunto de clientes que precisam ser atendidos e um conjunto de possíveis locais para instalação de facilidades, deseja-se determinar quais os melhores locais para instalação das facilidades de forma que todos os clientes sejam atendidos a um custo mínimo. Por exemplo, os clientes podem ser um conjunto de casas e as facilidades podem ser instalar postos de pronto-socorro. O custo pode estar relacionado com a distância das casas ao pronto-socorro mais próximo.

2.3.6.2 Projeto de circuitos integrados

Os seguintes tipos de problemas ocorrem repetidamente no projeto de sistemas de *hardware.* Tais sistemas possuem muitos módulos, cada um contendo sua própria pinagem, isto é, uma certa quantidade de pinos. A posição física de cada módulo é pré-determinada. Para interconectar um dado conjunto de pinos por fios, sujeito a determinadas condições: (i) no máximo dois fios podem ligar dois pinos (devido ao seu tamanho e possíveis mudanças futuras no *layout*); (ii) o comprimento total dos fios deve ser minimizada (para facilidade e organização da fiação). Desse modo, o problema da fiação torna-se um PCV com (n+1) cidades.

2.3.6.3 Seqüênciamento de tarefas

Em certas fábricas, um produto final é criado a partir da execução de uma seqüência de pequenas tarefas. Essas tarefas podem possuem regras de precedência entre si e

particularidades que exigem um ou outro tipo de máquina em determinado estado para sua execução. Considere, particularmente, uma seqüência de n tarefas numa única máquina. As tarefas podem ser feitas em qualquer ordem e o objetivo é completá-las no menor tempo possível. Assumindo que a máquina deve está no estado S_j (que pode ser: rotação, temperatura, pressão, espessura de corte, cor de pintura, ou qualquer outro estado) para fazer a tarefa j e que o estado inicial e final para a máquina é S_0.

Considere que o tempo utilizado para completar a tarefa j diretamente após a tarefa i seja,

$$t_{ij} = c_{ij} + p_j,$$

onde c_{ij} é o tempo requerido para transformar a máquina do estado S_i para S_j e p_j é o tempo real para realizar a tarefa j (com $p_0 = 0$). Para um dado ciclo de permutação p de {0, ..., n}, o tempo requerido para completar todos as tarefas é

$$\sum_{i=0}^{n} \left(c_i p_{(i)} + p p_{(i)} \right) = \sum_{i=0}^{n} c_i p_{(i)} + \sum_{j=1}^{n} p_j$$

Visto que a soma de todos os p_j‘s é uma constante, segue que o problema de seqüênciamento de tarefas é um PCV.

2.3.6.4 Aplicações mais recentes (APPLEGATE et al., 1998).

Além das aplicações relatadas acima, novas necessidades têm surgido modeladas pelo PCV.

- Ligações mais rápidas de computadores a um custo menor, também pode ser de telefones ou de usuários de eletricidade;
- Projeto de fibra óptica em rede de comunicação;

- Projetar planos de vôos para helicópteros que fiscalizam torres de perfuração petróleo ou de alta tensão e,
- Programar a coleta das moedas das cabines de telefones.

CAPÍTULO 3

Simulated Annealing

Este capítulo descreve um novo método dos chamados métodos computacionais inteligentes (TANOMARU, 1995), que são inspiradas em observações nos complexos sistemas presentes na natureza, o *Simulated Annealing*, onde é dada uma descrição geral sobre os fundamentos. Apresenta-se o contexto no qual o mesmo está inserido como participante das novas abordagens, para então, introduzir sua estrutura e seus elementos básicos com argumentações a respeito da influência dos parâmetros na sua eficácia. Faz-se algumas considerações teóricas sobre a garantia da convergência desse método e finaliza abordando algumas técnicas que podem aumentar a eficiência do algoritmo.

3.1 Introdução

Simulated Annealing é um método para encontrar soluções satisfatórias para problemas de otimização difíceis. Desde que KIRKPATRICK et al. (1983) e posteriormente CERNY (1985) mostraram a analogia entre o comportamento dos problemas de otimização combinatorial e os grandes sistemas físicos estudados em mecânica estatística, que o método vem recebendo muita atenção dos pesquisadores. Eles mostraram que o modelo da mecânica estatística para simular processos de *annealing* (recozimento), proposto inicialmente por METROPOLIS et al. (1953), podia ser estendido para resolver problemas de otimização em geral, especialmente problemas de otimização combinatorial.

O funcionamento do algoritmo *Simulated Annealing* talvez seja mais facilmente explicado como sendo uma extensão do simples e familiar método de busca local ou pesquisa na vizinhança (seção 2.2.3). A busca local somente requer a definição de um esquema de vizinhança e um método de avaliação do custo de uma solução particular. O algoritmo sempre termine numa solução ótima, que dependendo da solução inicial pode ser muito pobre.

A armadilha do ótimo local faz da busca local uma heurística pobre para muitos problemas de otimização combinatorial. A figura 3.1. ilustra uma estratégia de aproximação do algoritmo de busca local. Uma propriedade desejável de qualquer algoritmo é a habilidade de achar uma boa solução independente do ponto de partida.

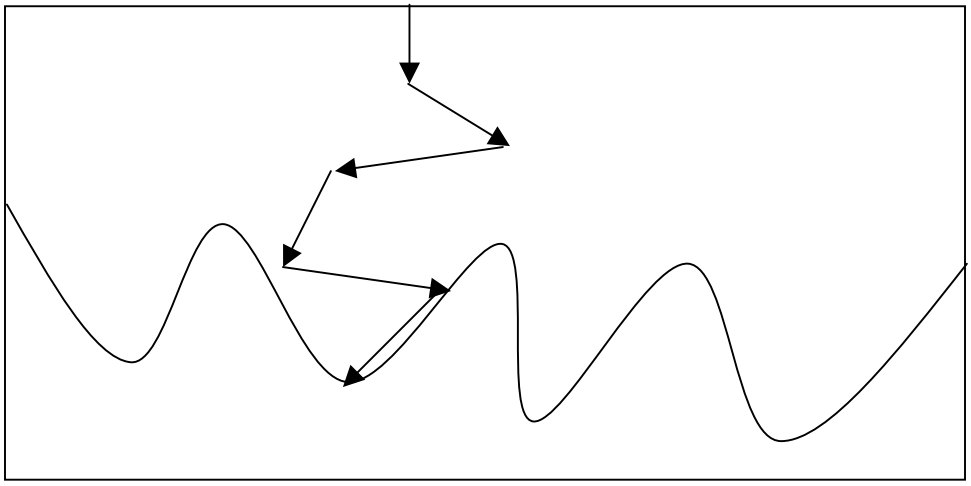

Figura 3.1 - Ilustração da estratégia da pesquisa local

Uma forma de escapar da armadilha do ótimo local é reiniciar a busca local de várias soluções iniciais diferentes e utilizar a melhor solução encontrada como solução do algoritmo. Tal estratégia reduz a dependência da solução inicial, mas apresenta um outro problema que consiste em determinar quando parar o algoritmo. Observe que repetidas busca locais convergem assintoticamente para a solução ótima usando todas as soluções como ponto de partida, o que não é viável em problemas grandes.

3.2 Analogia física

Toda motivação do *Simulated Annealing* teve sua origem na mecânica estatística que congrega métodos que analisam as propriedades de agregação dos átomos presentes tanto em substâncias sólidas quanto em substâncias líquidas. Devido à quantidade extremamente elevada de átomos (da ordem de 10^{23} átomos por centímetro cúbico), a análise de um sistema para observar seu comportamento em relação à variação da temperatura é caracterizada através do comportamento mais provável dos seus elementos (átomos) tendo em vista as flutuações randômicas, ou seja, o comportamento médio nas várias temperaturas.

Uma questão fundamental da mecânica estatística diz respeito a seguinte pergunta: O que acontece quando um sistema tem sua temperatura reduzida ao extremo? Dependendo de como ocorre esse resfriamento o sistema pode atingir um estado que é referenciado como um estado fundamental ou estado de energia mínima. Normalmente, não é o estado natural das substâncias, sendo portanto obtido somente através de experimentos. O processo para obter um estado de energia mínima é chamado de *annealing* (recozimento) e possui os seguintes passos:

a) Aumento da temperatura do banho térmico até um valor máximo no qual o sólido derrete;

b) Redução cuidadosa da temperatura do banho térmico até que as partículas se formem por si mesmas um arranjo que é o estado fundamental do sólido.

Dando uma descrição pode-se dizer que *annealing* (recozimento) é o processo de aquecimento de um sólido, por exemplo, até seu ponto de fusão, seguido de um resfriamento gradual e vagaroso, até que se alcance novamente seu enrijecimento. Nesse processo, o resfriamento vagaroso é essencial para se manter um equilíbrio térmico no qual os átomos encontrarão tempo suficiente para se reorganizarem em estrutura uniforme com energia mínima. Se o sólido é resfriado bruscamente, seus átomos formarão uma estrutura irregular e fraca, com alta energia em conseqüência do esforço interno gasto.

Na fase líquida todas partículas do sólido arranjam-se aleatoriamente. No estado de baixa energia as partículas formam um arranjo altamente estruturado conhecido como

cristal onde a energia do sistema é mínima. O estado fundamental do sólido é obtido somente se a temperatura máxima for suficientemente alta e o resfriamento for feito suficientemente lento. Caso contrário o verdadeiro estado de energia mínima não é alcançado ficando o sólido num estado meta-estável.

O processo físico de recozimento pode ser modelado com sucesso por métodos de simulação da física da matéria condensada. Por volta de 1953, METROPOLIS et al. (1953) introduziram um algoritmo simples para simular a evolução da temperatura de um sólido em um banho quente até o equilíbrio térmico. Segundo eles, quando os átomos se encontram em equilíbrio, em uma temperatura *T*, a probabilidade de que a energia do sistema seja *E*, é proporcional à $e^{-E/kT}$, onde *k* é conhecida como constante de *Bolzmann*.

Desta forma, a probabilidade de que a energia de um sistema seja (*E* + *dE*) pode ser expressa por:

$$prob\,(E + dE) = prob\,(E)\;prob\,(dE) = prob\,(E)\;e^{-dE/kT}$$

Em outras palavras, a probabilidade de que a energia de um sistema passe de *E* para *E*+*dE* é dada por $e^{-dE/kT}$. Na expressão $e^{-dE/kT}$, como *k* é uma constante, observa-se que a medida que *T* diminui, a probabilidade da energia do sistema se alterar é cada vez menor. Nota-se, também, que, nessas condições, quanto menor o valor de *dE*, maior a probabilidade de a mudança ocorrer.

O algoritmo introduzido por esses autores é baseado em técnicas de Monte Carlo e gera uma seqüência de estados do sólido como descrito a seguir. Dado um estado corrente i do sólido com energia E_i, então o estado subseqüente j é gerado aplicando um mecanismo de perturbação que transforma o estado corrente no próximo estado por uma pequena distorção. A energia do próximo estado é E_j. Se a diferença de energia $E_j - E_i$, é menor ou igual a 0, o estado j é aceito como estado corrente. Se a diferença de energia é maior que 0, o estado j é aceito com uma certa probabilidade que é dada por

$$\exp\left(\frac{E_i - E_j}{k_B T}\right),$$

onde T denota a temperatura do banho quente e k_B uma constante física conhecida como constante de *Boltzmann.* A regra de aceite descrita acima é conhecida como critério de *Metropolis* e o algoritmo que a usa é conhecido como algoritmo de *Metropolis*.

Se a redução da temperatura é feita suficientemente lenta, o sólido pode encontrar o equilíbrio térmico em cada temperatura. No algoritmo de Metropolis isto é obtido gerando um grande número de transições em uma determinada temperatura. O equilíbrio térmico é caracterizado pela distribuição de *Boltzmann*. Esta distribuição dá a probabilidade de sólidos estando no estado i com energia E_i na temperatura T, e é dado por

$$P_T\{X=i\} = \frac{1}{Z(T)}\exp\left(\frac{-E_i}{k_B T}\right),$$

onde X é uma variável estocástica denotando o estado corrente do sólido. Z(T) é a *função partição*, que é definida como

$$Z(T) = \sum_j \exp\left(\frac{-E_j}{k_B T}\right),$$

onde o somatório abrange todos os possíveis estados.

O método computacional que imita esse processo de recozimento de um sólido é chamado de *Simulated Annealing.*

3.3 Descrição do algoritmo

O algoritmo *Simulated Annealing,* oferece uma forma de escapar do ótimo local analisando a vizinhança da solução corrente e aceitando a solução que traga melhora mas, aceitando também solução que piore a solução corrente com uma probabilidade que é menor quanto maior for à distância entre essa solução e a solução corrente. A condição para

aceitar ou rejeitar um movimento que aumente a função de custo (ou seja solução pior) é determinado por uma seqüência de números randômicos, mas com uma probabilidade controlada. A probabilidade de aceitar um movimento que aumente a função de custo é chamada de função de aceite e é normalmente representada por exp(- Δ / T), onde Δ é a diferença entre as soluções e T é um parâmetro de controle que corresponde a temperatura fazendo uma analogia com *annealing* físico. Esta função de aceite implica que os pequenos aumentos da função de custo são provavelmente mais aceitos que os grandes, e que quando a temperatura T é alta, mais movimentos são aceitos, mas quando a temperatura T se aproxima de zero, muitos movimentos que aumentam a função de custo serão rejeitados.

Assim, o algoritmo *Simulated Annealing* é iniciado com um valor de temperatura T relativamente alto para evitar que prematuramente fique preso a um mínimo local . O algoritmo então procede tentando um certo número de movimento na vizinhança em cada temperatura, enquanto o parâmetro temperatura é gradualmente reduzido. Veja na figura 3.2 uma ilustração de como o algoritmo *Simulated Annealing* caminha para fugir de um mínimo local.

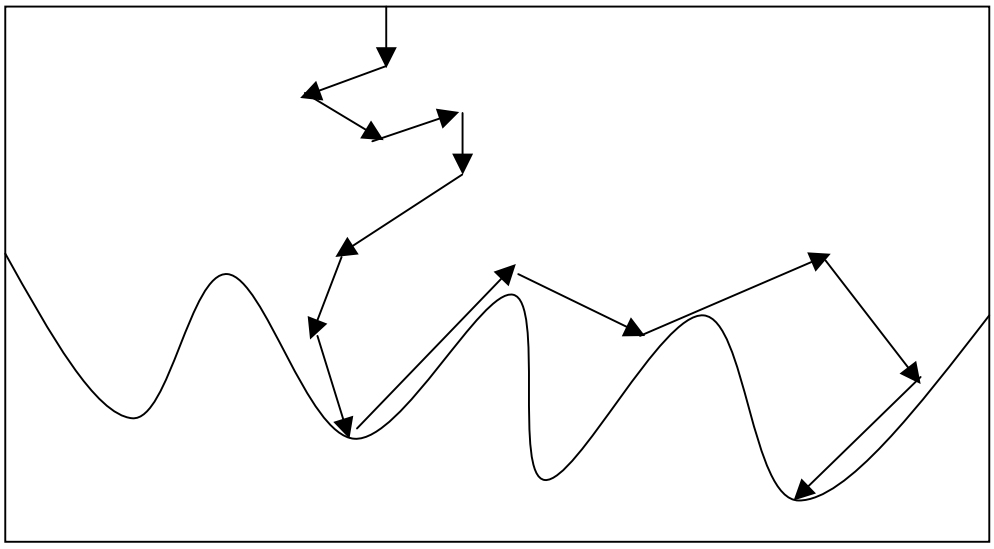

Figura 3.2- Ilustração da estratégia do *Simulated Annealing*

Computacionalmente, o *annealing* pode ser visto como um processo estocástico de determinação de uma organização dos átomos de um sólido, que apresente energia mínima. Em temperatura alta, os átomos se movem livremente e, com grande probabilidade, podem mover-se para posições que incrementarão a energia total do sistema. Quando se baixa a temperatura, os átomos gradualmente se movem em direção a uma estrutura regular, e somente com pequena probabilidade, incrementarão suas energias.

A sua semelhança com o método original no qual foi inspirado é muito grande. Na sua apresentação, nos trabalhos independentemente de KIRKPATRICK et al. (1983) e CERNY (1985), é mostrado como um modelo de simulação de *annealing* de sólidos, como proposto por METROPOLIS et al. (1953), pode ser utilizado em problemas de otimização, onde a função objetivo, a ser minimizada, corresponde à energia dos estados do sólido. Para esse propósito eles assumiram uma analogia entre um sistema físico de partículas e um problema de otimização combinatorial:

a) Soluções de problema de otimização são equivalentes a estados do sistema físico.

b) O custo de uma solução é equivalente à energia de um estado.

c) A seleção de uma solução vizinha em um problema de otimização é equivalente à perturbação de um estado físico.

d) O ótimo global de um problema combinatório é equivalente ao estado fundamental de um sistema de partículas.

e) Um ótimo local de um problema combinatório é equivalente a resfriamento rápido no sistema físico.

Através dessa analogia e aplicando sucessivamente o algoritmo de Metropolis para um problema de otimização combinatorial em temperaturas decrescentes uma nova técnica de otimização conhecida como *Simulated Annealing* foi definida.

Antes de passar para a descrição do algoritmo propriamente dito, a título de exemplo será definido o problema de otimização que se pretende resolver. Para tanto, seja *S* o espaço total de soluções de um problema combinatorial. Ou seja, *S* é o conjunto finito que contém todas as combinações possíveis que representam as soluções viáveis para o problema. Seja *f* uma função de valores reais definida sobre *S,* ou seja, $f : S \rightarrow R$. O problema se constitui em encontrar uma solução (ou estado) $i \in S$, tal que $f(i)$ seja mínimo.

Uma das formas mais simples de tentar resolver o problema, como já aludido anteriormente, é utilizar uma busca local em *S*, onde o processo se inicia com uma solução, normalmente, tomada de forma aleatória. Uma outra solução *j* é então gerada, na vizinhança desta, por meio de um mecanismo apropriado e dependente do problema. Caso uma redução do custo dessa nova solução seja verificada, ou seja, $f(j) < f(i)$, a mesma passa a ser considerada a solução corrente e o processo se repete. Caso contrário, a nova solução é rejeitada e uma outra gerada. Esse processo se repete até que nenhum melhoramento possa ser obtido na vizinhança da solução corrente, após um número determinado de insistências. O algoritmo retorna, então, o valor da última solução corrente, considerada uma solução de mínimo local.

O grande problema desse método, muito simples e rápido, é que o mínimo local encontrado pode estar longe de ser um mínimo global, o que se traduziria em uma solução inaceitável para o problema. Uma estratégia muito simples de aprimorar a solução obtida através desse tipo de algoritmo, seria escolher a menor solução, de um conjunto de soluções obtidas de execuções sucessivas, realizadas a partir de diferentes soluções iniciais.

Simulated annealing não utiliza essa estratégia. Esse método tenta evitar a convergência para um mínimo local, aceitando, às vezes, uma nova solução gerada, mesmo que essa incremente o valor de *f*. O aceite ou a rejeição de uma nova solução, que causará um incremento de δ em *f*, em uma temperatura *T*, é determinado por um critério probabilístico, através de uma função *g* conhecida por função de aceite. Normalmente, essa função é expressa por

$$g(\delta, T) = e^{-\delta / T}$$

Caso $\delta = f(j) - f(i)$, for menor que zero, a solução *j* será aceita como a nova solução corrente. Caso contrário, a nova solução somente será aceita se

$$g(\delta, T) > random(0, 1)$$

A semelhança com o método original de simulação de recozimento na termodinâmica, como já aludido, é grande, pois o parâmetro δ, corresponde à variação da energia de um estado para outro *(dE)* e o parâmetro de controle *T*, corresponde à

temperatura. Uma vez que, agora *T* é imaginário, a constante *K*, que aparecia na expressão original multiplicando *T*, é considerada igual a *1*.

Da mesma forma que no processo físico, a função *g* (δ, *T*) implica em:

i) a probabilidade de aceite de uma nova solução é inversamente proporcional ao incremento de δ e;

ii) quando *T* é alto, a maioria dos movimentos (de um estado para outro ou de uma solução para outra) é aceita, entretanto, a medida que *T* se aproxima de zero, a grande maioria das soluções são rejeitadas.

Procurando evitar uma convergência precoce para um mínimo local, o algoritmo inicia com um valor de *T* relativamente alto. Esse parâmetro é gradualmente diminuído e, para cada um dos seus valores, são realizadas várias tentativas de se alcançar uma melhor solução, nas vizinhanças da solução corrente, como pode ser observado no pseudo-código apresentado na figura 3.3.

```
Procedimento Simulated Annealing
  início
        S ← S0;
        T ← T0;
        enquanto temperatura elevada faça
        para interações para equilíbrio faça
              gerar uma solução S' de N(S);
                    avaliar a variação de energia – ΔE = f(S') – f(S);
                    se ΔE < 0 then
                          S ← S';
                    senão
                          gerar u ∈ random[0,1];
                          se u < exp(-ΔE/KB . T) then
                                S ← S';
                          fimse
                    fimse
              fimpara
                    reduzir a temperatura T;
        fimenquanto
        fimprocedimento
```

Figura 3.3 – Pseudo-código do algoritmo *Simulated Annealing*

Em termos puramente teóricos tem sido mostrado que o algoritmo converge para um conjunto de soluções ótimas globais quando o tempo tende para o infinito. Esta propriedade de convergência assintótica tem sido provada por meio de um modelo de cadeia de Markov. Entretanto quando implementando o algoritmo, considerações práticas devem ser dadas ao programa de resfriamento para assegurar convergência para uma solução de boa qualidade em um tempo razoável.

3.4 Programas de resfriamento

É de grande interesse o desenvolvimento de uma heurística com tempo polinomial e que nos dê uma boa aproximação do ótimo global. Uma implementação do *Simulated Annealing* com tempo finito pode ser realizada gerando cadeias de Markov homogêneas de tamanho finito para uma seqüência de valores descendentes de parâmetros de controle. O conjunto de parâmetros a serem determinado compõe o que se chama *programa de resfriamento* (*cooling schedule*).

Um programa de resfriamento especifica uma seqüência finita de valores dos parâmetros de controle, isto é:

a) uma temperatura inicial,

b) uma função de redução da temperatura,

c) um critério de parada que especifica a temperatura final,

d) um número finito de transições em cada temperatura,

e) um tamanho finito de cada cadeia de Markov homogênea,

Para limitar o esforço computacional do algoritmo precisa-se aumentar a queda da temperatura o que implica em menos iterações com diferentes temperaturas, mas exige cadeia de Markov maiores. Então há a necessidade de uma solução de compromisso entre o decremento da temperatura e cadeias de Markov pequenas. A busca por programas de resfriamento adequados tem sido assunto de estudo muito explorado.

3.4.1 Programa de resfriamento geométrico

Esse programa foi proposto por KIRKPATRICK et al. (1983) e permanece largamente utilizado. Esse programa não é baseado em qualquer justificativa teórica bem como não se preocupa com a analogia física. Cabe salientar que há muitos parâmetros a serem ajustados para um determinado problema, mas uma vez encontrados bons parâmetros para o problema, poucos ajustes são necessários para outras instâncias do mesmo.

Valor da temperatura inicial

O parâmetro temperatura T_0 deve ser suficientemente grande para permitir virtualmente que todas as transições sejam aceitas. Desde que o valor apropriado de T_0 varia de problema para problema e de instância para instância, uma metodologia genérica é proposta para obter-se uma temperatura suficientemente quente para todos problemas. Executando o *Simulated Annealing* inicialmente com uma temperatura pequena, a taxa de aceite $\chi(T)$ pode ser determinada pela proporção entre o número de transições aceitas e o número l de transições propostas usando o teste usual de Metropolis. Se o valor de $\chi(T)$ é maior que a taxa de aceite inicial desejada χ_0, então a temperatura inicial T_0=T; caso contrário T é incrementado por um fator $\beta > 1$, $T = \beta T$. O processo é repetido k vezes até que o valor de T forneça uma taxa de aceite $\chi(T) > \chi_0$, ponto em que T_0 recebe o valor corrente de T.

Uma sugestão para o valor inicial de T é: $T = \ln f(x_0)$, onde $f(x_0)$ é o valor da função objetivo da solução inicial. O valor de l sugerido é: $l = \lceil h \times |N(x, \sigma)| \rceil$, onde $|N(x,\sigma)|$ é o tamanho da vizinhança e h é uma constante multiplicativa. Se o valor de l for muito pequeno, então a medida da taxa de aceite torna-se muito imprecisa; cada contrário, a taxa de aceite terá grande precisão mas poderá haver um grande desperdício de tempo. Se o valor de β for muito grande há o risco de ultrapassar em muito a temperatura inicial ideal, caso contrário também pode haver um gasto excessivo de tempo.

Redução da temperatura

A maneira pela qual a temperatura é reduzida da o nome ao programa de resfriamento. A temperatura é reduzida pela multiplicação por um fator fixo $\alpha < 1$:

$$T_{k+1} = \alpha T_k$$

Desde que, de acordo com a teoria, a temperatura dever ser reduzida lentamente, o valor de α é usualmente selecionado entre 0.8 e 0.99 com tendência a valores próximos de 1(dando uma redução de temperatura mais lenta). O valor de α usado no algoritmo afeta o tempo de execução e a qualidade desejada para a solução. Se o valor dele for pequeno, então a temperatura resfria rapidamente levando o algoritmo a parar rapidamente mas levando a soluções pobres. Se α estiver próximo de 1 então a temperatura reduz lentamente, exigindo maior tempo de execução e obtendo soluções de melhor qualidade.

Tamanho da cadeia

O programa de resfriamento geométrico tenta restabelecer o equilíbrio após cada redução de temperatura executando um número L fixo de iterações de movimentos pela vizinhança. O número de transições requeridas para restabelecer esse equilíbrio depende da variação da temperatura e do tamanho da instância do problema. O programa de resfriamento geométrico ignora a variação da temperatura e estabelece um tamanho suficiente que pode ser atrelado ao tamanho do problema:

$$L = \lceil c \times |N(x, \sigma)| \rceil$$

onde $|N(x,\sigma)|$ é o tamanho da vizinhança e c é uma constante multiplicativa.

O valor de c afeta o algoritmo de duas maneiras. Se c é pequeno então cada cadeia consistirá de um pequeno número de transições propostas. Então a temperatura irá resfriar rapidamente e o algoritmo terminará em um pequeno intervalo de tempo. Entretanto cadeias pequenas não permitem o sistema restabelecer o equilíbrio. Por outro lado, valores

de c grandes exigem muito tempo de execução e, em conseqüência, espera-se soluções de melhor qualidade.

Condição de parada

O algoritmo *Simulated Annealing* deve parar quando estiver congelado. Como determinar quando o sistema está congelado não é claro mas deve significar que há uma pequena ou nenhuma possibilidade de escapar da solução corrente. O método aqui utilizado é parar quando a solução corrente permanece a mesma para um número s de cadeias consecutivas. Esta condição de parada deve ocorrer somente em baixas temperaturas. O valor de s pode afetar o algoritmo provocando parada prematura, resultando em uma solução que não é a melhor que o algoritmo pode encontrar. Se o valor de s não afeta o algoritmo, então a condição de parada usada não está atuando o suficiente para reconhecer que o sistema já está congelado.

3.4.2 Programa de resfriamento com tempo polinomial

Esse programa foi proposto inicialmente por AARTS & KORST (1990).Tem sido mostrado teoricamente que o mesmo produz tempo de execução limitado por uma função polinomial do tamanho da instância do problema. Como a maioria do programas de resfriamento, ele não garante qualidade da solução final produzida. A preocupação básica é atingir um equilíbrio aproximado em uma dada temperatura

Valor da temperatura inicial

Se forem gerados alguns ensaios com uma dada temperatura T, se obtém o valor m_1 denotando o número de transições propostas de i para j com $f(j) \leq f(i)$, e m_2 o número de transições para o qual $f(j) > f(i)$. Seja $-\overline{\Delta f}^{(+)}$, a variação média do custo para as m_2 transições onde o custo aumenta. Então a taxa de aceite χ pode ser aproximada pela seguinte expressão:

$$\chi \approx \frac{m_1 + m_2 . \exp\left(\frac{-\overline{\Delta f}^{(+)}}{T} \right)}{m_1 + m_2},$$

do qual se obtém:

$$T = \frac{\overline{\Delta f}^{(+)}}{\ln\left(\frac{m_2}{m_2 . \chi - m_1 (1 - \chi)} \right)}.$$

Sabendo que a temperatura a ser calculada deve ser suficientemente alta para se obter uma taxa de aceite próxima de 1.0, então pode se realizar m ensaios aceitando todas as transições e computa-se m_1 e m_2 sem a aplicação do critério de Metropolis e calcular $-\overline{\Delta f}^{(+)}$. De posse desses valores e da taxa de aceite desejada χ, usar a fórmula 2.1 para a obtenção de T_0.

Redução da temperatura

Deseja-se uma função de redução da temperatura, tal que os passos de redução sejam pequenos o suficiente para permitir que as cadeias de Markov sejam as menores possíveis para permitir a condição de "quase equilíbrio" dada abaixo:

$$\forall k \geq 0 : ||q(T_k) - q(T_{k+1})|| < \varepsilon,$$

para algum valor positivo ε. Evidentemente assume-se que a condição de quase equilíbrio existe em T_0.

Então para duas temperaturas sucessivas esperamos que as distribuições estacionárias estejam próximas. Isto pode ser quantificado por:

$$\forall i \in S: \frac{1}{1+\delta} < \frac{q_i(T_k)}{q_i(T_{k+1})} < 1+\delta \text{, k=0,1,...,}$$

para algum número pequeno positivo que pode correspondente a ε da condição de “quase equilíbrio”.

A partir dai, AARTS & KORST (1990) chegam à seguinte expressão:

$$T_{k+1} > \frac{T_k}{1+\frac{T_k \cdot \ln(1+\delta)}{< f >_{T_k} - f_{opt} + 3\sigma_{T_k}}} \text{, k=0,1,..}$$

Como nem sempre se conhece a f_{opt}, a solução ótima, pode-se omitir o termo $<f>_{Tk}$ - f_{opt} deixando que sua ausência seja contrabalançada pelo parâmetro δ.

Condição de parada

O algoritmo deve terminar quando o custo médio para a temperatura tendendo a zero tender para o valor ótimo.

O algoritmo deve terminar na k-ésima iteração que satisfaça a condição:

$$\frac{T_k}{\langle f \rangle_\infty} \left. \frac{\partial \langle f \rangle_T}{\partial T} \right|_{T=T_K} < \varepsilon_S$$

onde ε_s é um número positivo pequeno, chamado *parâmetro de parada*.

Tamanho da cadeia de Markov

Se em nosso processo de resfriamento usarmos cadeias de Markov de tamanho fixo igual ao tamanho da vizinhança, visitaremos cerca de 2/3 das diferentes soluções dessa vizinhança, segundo AARTS & KORST (1990).

Usando a função de redução de temperatura dada anteriormente da seguinte forma:

$$T_{k+1} = \frac{T_k}{1 + \alpha_k T_k}, \text{ k=0,1,..}$$

onde $\alpha_k = \frac{\ln(1+\delta)}{3\sigma_{T_k}}$, k=0,1, ...

Seja K é o primeiro inteiro que satisfaz o critério de parada. Então K= O(ln(|S|)

Este resultado é muito importante e sua prova pode ser encontrada em . Um passo da prova consiste em provar que K é expresso como função de T_k e o outro passo consiste em estabelecer um limite inferior para T_k.

Assim o algoritmo requer O(τ.L.ln|S|) passos, onde L é o tamanho de cada cadeia de Markov, ln|S| denota o limite superior do número de cadeias de Markov e τ denota o tempo computacional para um transição. Para a maioria dos problemas combinatórios L e τ podem ser escolhido para serem polinomiais. Como ln|S| é polinomial, então o *Simulated Annealing* executa em tempo polinomial.

3.4.3 Programa de resfriamento com tempo linear

LUNDY & MEES (1986) sugerem uma função de redução de temperatura da seguinte forma:

$$t_k = \frac{t_k}{1 + \beta . t_k}$$

onde β é um número pequeno que controla a redução de temperatura. Quando ele for pequeno, a queda é mais lenta, quando β aumenta a queda é mais rápida, gerando soluções de qualidade inferior às geradas para um β menor.

Como o critério de parada pode ser expresso em termos de um valor mínimo para a temperatura ou em termos do "congelamento" do sistema, a proposta deste programa é:

$$t \leq \frac{\varepsilon}{\ln\left[(|S|-1)/\theta\right]}$$

onde S é o espaço de soluções. Deseja-se produzir soluções que estejam a ε da solução ótima com probabilidade θ.

Uma regra de parada simples consiste em se estabelecer um percentual de iterações a ser executado sem que haja melhora na solução, assim não haverá desperdício de tempo. No entanto, qualquer regra deve ser bem ajustada de forma a permitir uma queda suficiente da temperatura a fim de que a convergência seja assegurada.

Nesta situação, uma condição que se considera importante não é a temperatura, mas a taxa de aceitação. Quando a média da taxa de aceitação atinge valor zero, significa que a variação do valor da função objetivo, visitadas naquela iteração, é muito superior ao valor da temperatura. A partir daí o algoritmo está em busca local o que pode ser prorrogado por um certo tempo. Um valor que pode ser experimentado é executar *n.ln n* reduções de temperatura após a taxa de aceitação média atingir zero em uma dada temperatura. O critério adotado no programa de tempo polinomial pode ser colocado simultaneamente, ou seja, quando a média dos valores da função objetivo de uma temperatura para outra sofrer redução inferior a um parâmetro ε_t, o que permite o algoritmo parar antes da *n.ln n* estabelecidas.

3.5 Comentários teóricos

Apesar de todo esforço dispensado com a publicação de vários trabalhos sobre a convergência do algoritmo *Simulated Annealing* ainda não foi possível se ter uma provar

matematicamente definitiva que justifique o sucesso empírico desse algoritmo. LEWIS & PAPADIMITRIOU (2000) argumentam sobre uma nova e desafiadora fronteira da teoria da computação de hoje que é de demonstrar teoricamente os impressionantes resultados práticos que tem se obtidos com o emprego dessas novas heurísticas, incluindo o *Simulated Annealing*.

Matematicamente, o algoritmo *Simulated Annealing* pode ser modelado através da teoria de cadeias de Markov. Utilizando esse modelo, vários resultados importantes, tratando de condições suficientes para a convergência, têm surgido na literatura. A seguir, um resumo da discussão apresentada por AARTS AND KORST (1990) em sua análise teórica da convergência assintótica do algoritmo *Simulated Annealing*.

Como já comentado anteriormente, da teoria da física da matéria condensada, no processo físico de recozimento se a temperatura é reduzida cuidadosamente então o sólido atinge o equilíbrio térmico em cada temperatura, onde o equilíbrio térmico é expresso pela distribuição de *Boltzmann*. Estendendo a analogia entre otimização combinatorial e o processo físico de recozimento podemos conjecturar que para um problema P, com um espaço de soluções S e uma função de custo f(x) $\forall x \in S$, após um suficientemente grande número de movimentos na vizinhança usando o critério de aceite de Metropolis em uma temperatura T fixa, a probabilidade de estarmos em um estado x é dada por:

$$P_T(X = x) \overset{def}{=} q_T(x) = \frac{1}{Z(T)} \exp\left(\frac{-f(x)}{T}\right)$$

onde Z(T) é uma função de partição normalizada definida como:

$$Z(T) = \sum_{y \in S} \exp\left(\frac{-f(y)}{T}\right)$$

A partir dessas expressões, e considerando-se que a distribuição é estacionária, obtém-se:

$$\lim_{T \to 0} P_T(X = x) = \frac{I_S.(x)}{|S_*|} \qquad (1)$$

onde S^* é o conjunto de soluções ótimas com custo f^* e $I_{S*}(x)$ é a função indicador tal que:

$$I_A(a) = \begin{cases} 1, \forall a \in A \\ 0, \forall a \notin A \end{cases}$$

A prova da convergência assintótica é imediata e assume a minimização do problema P. A prova baseia-se no fato de que:

$$\forall a \leq 0, \lim_{T \to 0} \exp\left(\frac{a}{T}\right) = \begin{cases} 1, a = 0 \\ 0, cc \end{cases}$$

Pelo resultado apresentado em (1) demonstra que se a distribuição estacionária para o processo de *Simulated Annealing* ocorre segundo a distribuição de Boltzmann, então o mesmo converge assintoticamente para um conjunto de soluções ótimas globais.

Os resultados acima são baseados no fato de que o comportamento de algoritmo s*imulated annealing* poder ser modelado usando cadeias de Markov. A probabilidade de mover de um estado para outro pode ser representada por uma matriz, e para uma dada temperatura, a probabilidade de transição $P_{i,j}$ de mover da solução i para a solução j depende somente de i e j. Este tipo de matriz de transição é conhecida como cadeia de Markov homogênea. Quanto mais longa para obter uma seqüência de trocas que transforme qualquer solução em qualquer outra com probabilidade diferente de zero, o processo converge através de uma distribuição estacionária que é independente da solução inicial. Como a temperatura tende para zero, assim a distribuição estacionária tende para uma distribuição uniforme sobre o conjunto ótimo de estados ou soluções. Embora seja um resultado bastante interessante, nada é garantido sobre o número de iterações necessárias para alcançar a convergência.

Considerando que no *Simulated Annealing* a temperatura não é constante, mas é reduzida após um número de iterações. Isto pode ser considerado também como um número de diferentes cadeias homogêneas, ou como uma simples cadeia não-homogênea em que a probabilidade de transição $P_{i,j}$ não depende somente dos estados i e j, mas também da temperatura, isto é, as probabilidades são dependentes do número de iterações. AARTS & VAN LAARHOVEN (1985) mostraram que se uma cadeia homogênea está próxima de sua distribuição estacionária então o número de iterações em uma dada temperatura será no mínimo o quadrado da cardinalidade do espaço de soluções. Visto que o espaço solução é usualmente exponencial em relação ao tamanho do problema, significa que desta forma o *Simulated Annealing* gastará mais que uma busca exaustiva, o que certamente não é interessante.

Outro resultado muito importante é fornecido no trabalho publicado por HAJEK (1988), no qual, não só as condições necessárias e suficientes para a convergência assintótica do algoritmo para um conjunto de soluções ótimas globais são fornecidas, mas também, o número de iterações necessárias para que essa convergência possa ocorrer. Entretanto algumas considerações referentes à aplicabilidade dessas condições devem ser consideradas, já que a maioria dos algoritmos implementados usa programa de resfriamento exponencial e não logarítmica como proposta por HAJEK (1988), o que foi comprovado na avaliação experimental de JOHNSON et al. (1989) no problema de particionamento de grafos. O uso de programas de resfriamento logaritmo, na prática resulta em tempos computacionais impraticáveis. Na realidade, a discrepância entre os resultados da teoria e os da prática em relação ao *Simulated Annealing* ainda é um problema insolúvel.

3.6 Considerações finais

O algoritmo *Simulated Annealing* tem sido aplicado em muitos problemas de otimização que ocorrem nas mais diversas áreas, tais como projeto físico de computador (VLSI), processamento de imagem, física e química molecular, agendamento de tarefas em máquinas, roteamento de veículo, etc. Muitos problemas de otimização combinatorial têm

sido mostrados pertencerem à classe NP-completo, o que significa que o tempo de execução para qualquer algoritmo conhecido atualmente para garantir a solução ótima é uma função exponencial do tamanho do problema. É um das muitas abordagens heurísticas projetadas para dá uma boa, embora não necessariamente ótima solução num tempo de computação aceitável. Pertence à classe das novas técnicas que surgiram recentemente inspirados nos processos observados da natureza, chamados modelos computacionais inteligentes (TANOMARU, 1995), que quando modelados podem ser aplicados a problemas que a ciência atual ainda não consegue resolver satisfatoriamente.

Dentre os novos modelos, além do *Simulated Annealing* outros como *Tabu Search*, Redes Neurais e Algoritmos Genéticos, também têm recebido muita atenção dos pesquisadores em função de sua grande aplicabilidade em problemas práticos. Essas abordagens têm conseguido grandes resultados em problemas de otimização, especialmente combinatorial já que se trata de estratégia geral de pesquisa e que portanto não são vistas como um simples algoritmo, mas como métodos ou meta-algoritmo. O extraordinário sucesso desses métodos é devido vários fatores :

a) Faz referencia a mecanismo de otimização da natureza;

b) Aplicabilidade geral das abordagens;

c) Flexibilidade, já que levam em conta restrições específicas em casos reais;

d) Excelente relação entre qualidade da solução e facilidade de implementação.

O esforço requerido pelo *Simulated Annealing* pode variar muito, dependendo do problema e do tamanho da instância, de alguns segundos para um problema pequeno a muitas horas para um problema grande. Com isto em mente, há a necessidade de se investigar várias modificações no algoritmo básico para acelerar sua convergência para uma solução de boa qualidade. AARTS & KORST (1990) identificaram três categorias gerais de abordagens para acelerar o algoritmo *Simulated Annealing*, que são:

a) um algoritmo seqüencial mais eficiente,

b) aceleração do hardware,

c) projeto de algoritmo paralelo.

A seguir, vários métodos para criar algoritmos seqüenciais mais eficientes e uma rápida discussão sobre a implementação de algoritmos paralelos.

REEVES (1995) afirma que há evidências empíricas e a pesquisa teórica sugere que a maneira como o resfriamento é feito não é tão importante quanto à taxa de resfriamento. Ele sugere que quando usando *annealing* para novas aplicações é melhor começar com o programa geométrico ou com o sugerido por LUNDY & MEES (1986), tendo em vista que os mesmos trabalham sobre uma mesma faixa de temperatura e com taxas de resfriamentos semelhantes e por apresentarem bons resultados na literatura. Logo o esforço para acelerar o *Simulated Annealing* reside na obtenção de procedimentos eficientes para exploração da vizinhança. Como por exemplo, onde a avaliação da função objetivo requer $O(n^2)$ operações usando a estrutura de vizinhança pela troca de posição de um par de elementos, para atualizarmos o valor da função objetivo, basta calcularmos a variação introduzida por essa troca, o que pode ser feito em O(n). O ajuste de parâmetros é o ponto mais sensível para o desempenho do algoritmo, tanto em tempo de execução quanto à qualidade da solução.

JOHNSON et al. (1989) sugerem a substituição da função $\exp(-\delta/T)$ no critério de aceite, pela função $1 - (-\delta/T)$. Tal substituição, segundo eles, aumentaria a velocidade do algoritmo em cerca de 30%. A aceleração do hardware sugerida na literatura consiste em um hardware dedicado e específico para processamento das operações mais pesadas do algoritmo, ou alteração do microcódigo do computador, o que normalmente é muito difícil e de alto custo. A aceleração mais acessível consiste na utilização de máquinas de uso geral com maior poder de processamento.

A outra alternativa consiste no desenvolvimento de *Simulated Annealing* paralelo para tirar proveito da disponibilidade de diversas máquinas em ambiente de redes. Tal alternativa tem sido muito pesquisada e AARTS & KORST (1990) sugerem três estratégias: algoritmo divisão, algoritmo agrupamento, algoritmo erro.

CAPÍTULO 4

Método proposto

Neste capítulo é descrito o método proposto que tem como base o *Simulated Annealing*(SA). Inicialmente é feita uma definição formal do PCV bem como sua modelagem. Algumas considerações sobre o atual estágio de abordagens existentes para solucionar instâncias do problema tanto em termos de tempo de processamento, de solução ótima e número de cidades. Por fim, o método proposto é apresentado.

4.1 Introdução

A importância de encontrar um algoritmo aproximado para o PCV reside no fato de muitos problemas que ocorrem na ciência da computação, e em muitas outras áreas, poderem ser modelados a partir do mesmo. Desta forma, nas últimas décadas, ênfase vem sendo dada no desenvolvimento de algoritmos aproximados, geralmente utilizando métodos heurísticos que forneçam soluções de boa qualidade em tempo de processamento compatível com as necessidades.

Atualmente, observa-se um aumento considerado no interesse por abordagem baseadas em técnicas de otimização de busca local para o tratamento de problemas NP-Completos. São métodos aproximados que normalmente utilizam como base os algoritmos: genético, *Simulated Annealing*, *tabu search* e redes neurais.

Esses algoritmos têm sido intensivamente pesquisados em soluções de problemas gerais de otimização e, especialmente, nos combinatoriais. O grande sucesso alcançado por

esses algoritmos, principalmente nos últimos anos, é conseqüência de diversas características que os mesmo têm, dentre eles: a utilização de mecanismos modelados diretamente da natureza; aplicabilidade geral; flexibilidade nas adaptações às características específicas em casos reais; a excelente relação qualidade e facilidade de implementar e o

4.2 Definição formal do problema

Devido a sua simplicidade e aplicabilidade, o PCV tem há décadas servido de base para novas idéias, e atualmente tem um papel importante no desenvolvimento, teste e demonstração de novas propostas de técnicas de otimização.

Matematicamente, o PCV pode ser formulado como segue:

Dada uma matriz de custo C = (c_{ij}), onde c_{ij} representa o custo de ir da cidade i para a cidade j, (i, j = 1, ... , n), encontrar uma permutação π (i_1, i_2, i_3, ..., i_n) de inteiros de 1 até n que minimize a quantidade obtida pela soma.

$$\text{Custo} = c_{i1,i2} + c_{i2,i3} + ... + c_{in,i1}$$

Em geral algumas restrições são colocadas quando da obtenção de alguma solução para o PCV. A mais comum é dizer que o problema é euclidiano, ou seja, as distâncias estão distribuídas num plano e, portanto obedecem à desigualdade triangular e a simetria.

i) Se $c_{ij} = c_{ji}$, para todo i e j então o problema é dito simétrico, senão ele é assimétrico.

ii) Se para todo i, j, k com $c_{ik} = c_{ij} + c_{jk}$, o problema obedece a desigualdade triangular.

Além da análise combinatorial, outra área tem contribuído sobremaneira para os avanços nos estudos sobre soluções para PCV, emprestando suas ferramentas para modela-lo, a teoria dos grafos. É com o uso das ferramentas da teoria dos gráficos que tem se conseguido as propostas mais eficientes. A solução consiste em encontrar um circuito

hamiltoniano[9], isto é, um circuito que visita todas as cidades exatamente uma só vez, de custo mínimo.

Desta forma o PCV pode ser representado por um grafo G(V,C), onde V é conjunto de vértices que são as n cidades a visitar e uma matriz $C = [c_{ij}]$ de distâncias (custos) entre cada par de vértices (cidades) (i,j). A função custo é definida por:

$$f(v) = \sum_{i=1}^{n-1} c_{i,\,i+1} + c_{n,\,1}\,,$$

onde a solução v é um conjunto de vértices que forma o circuito hamiltoniano.

Por não exigir o emprego de ferramentas mais sofisticadas, o modelo de representação utilizado neste trabalho está baseado na definição matemática feita anteriormente, que é o de permutação.

4.3 Modelagem do problema

Numa aplicação que envolva SA na resolução de qualquer problema de otimização combinatorial, é fundamental que se defina primeiramente a forma como serão representadas as configurações ou soluções do problema. Uma representação natural do espaço de soluções para o problema em questão é, portanto, aquela que contém o conjunto de todas as permutações possíveis, como já visto, que para n cidades é (n-1)!/2. Define-se, assim, uma solução *i* para o problema, como sendo um elemento do conjunto de todas as permutações que é uma possível rota.

$$\pi_i = \{c_{i1}, c_{i2}, \ldots , c_{in}\}$$

onde cada elemento desse conjunto, c_{ik}, deve ser interpretado como as cidades naquela solução π_i e a ordem na qual as **n** cidades devem ser visitadas.

[9] Termo comum que na teoria dos grafos é definido como um caminho simples fechado em um gráfico conexo G, que passa em cada vértice de G exatamente uma vez, exceto naturalmente no vértice inicial que é considerado também vértice terminal. Homenagem ao famoso matemático Sir William Rowan Hamilton.

Dessa forma, utilizando esse modelo de representação, o problema de PCV passa a ter seu espaço de busca como sendo o conjunto de todas as permutações de rotas, isto é, determina-se, para cada permutação, o custo da rota correspondente. A melhor solução será aquela que apresentar a rota de menor valor.

Adicionalmente à forma de representação, deve-se definir uma estrutura de vizinhança através da qual, especialmente no SA, é possível se passar de uma solução a outra por meio de pequenas mudanças, através de algum mecanismo de perturbação. Existem vários métodos para se fazer essa perturbação, entretanto, o mais empregado está baseado na proposta de LIN (1965) que é o conceito de *λ-opt*.

Neste contexto uma estrutura de vizinhança pode ser definida pelo conjunto de todos as rotas que se obtém removendo k ligações e substituindo-as por k outras ligações, de tal modo que se mantenha uma rota válida. Para $K \geq 3$ existem diferentes maneiras de, depois de removidos as k ligações, se voltar a ligar as rotas. Para K = 2, existe apenas uma maneira de se fazer essas duas ligações de modo que a rota continue válida. Se retirarem as ligações (i, i + 1) e (j, j + 1) então a única maneira admissível de criar uma outra rota é ligando (i, j) e (i + 1, j + 1).

Considere o exemplo do conjunto de cidades mostrado na figura abaixo, correspondendo à representação de uma rota para o caso de n = 8 cidades como sendo {1, 2, 3, 4, 5, 6, 7, 8, 1}. Se as ligações (2, 3) ou seja i = 2 e (6, 7) ou seja j = 6, forem removidas, então as únicas ligações possíveis, de modo a manter-se um circuito, são (2, 6) e (3, 7). Com isso se inverte a direção das ligações entre 3 e 4, entre 4 e 5 e entre 5 e 6.

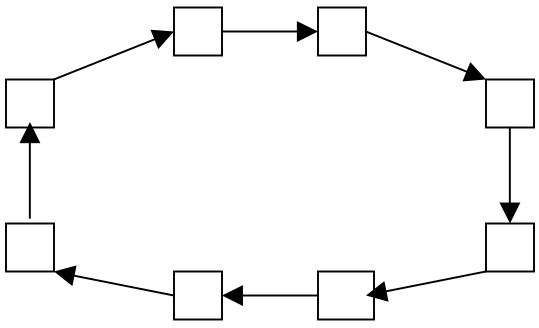

Figura 4.1 – Representação de uma rota ligando as 8 cidades

A nova rota assim obtida, mostrada na figura abaixo, é representada pela seqüência {1, 2, 6, 5, 4, 3, 7, 8, 1}.

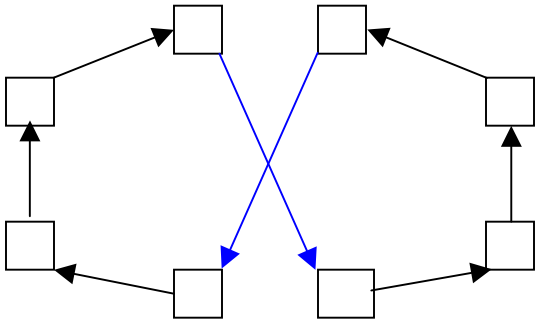

Figura 4.2 – Representação da rota após a troca

Deste modo a rota inicial {1, 2, 3, 4, 5, 6, 7, 8, 1} tem a rota {1, 2, 6, 5, 4, 3, 7, 8, 1} como sua vizinha e vice-versa. Assim, a estrutura da vizinhança, de modo mais simples, pode ser vistas como todas as possíveis 2-ligações que podem ser removidas de uma rota e religadas de modo a formar uma nova rota. Onde a solução obtida dessa forma combina com o conceito *λ-opt* para *λ*=2, ou melhor *2-opt*, onde a cardinalidade do conjunto vizinhança é n(n-1)/2, ou seja, combinação de **n** dois a dois.

Assim, dada uma seqüência, um vizinho é obtido por troca de lugar de dois números nessa seqüência. Além disso, qualquer rota pode ser obtida pela aplicação de uma ou mais vezes (número finito) desta operação a partir de outra rota qualquer. Por outras palavras, qualquer ponto do espaço é atingível a partir de qualquer outro ponto do espaço pela aplicação sucessiva do operador de vizinhança.

Adicionalmente esta estrutura de vizinhança é fácil de ser implementada, pois consiste apenas na troca de valores entre duas posições, posições essa que podem ser escolhidas aleatoriamente extraindo sem reposição dois números *i* e *j* entre 1 e *n*.

Retornando à figura 4.2, a solução para o problema ilustrado é:

{1, 2, 6, 5, 4, 3, 7, 8, 1}

Esta expressão deve ser interpretada como: o caixeiro parte da cidade 1 viaja até a cidade 2 segue até a cidade 6, e assim por diante. Tendo em vista que a abordagem proposta no presente trabalho envolve o algoritmo SA, essa definição de solução e vizinhança é importante, pois pode ser perfeitamente utilizada, já que não produz soluções inviáveis.

4.4 Abordagens existentes

A resolução de problemas de otimização combinatória tem sido tratada na literatura basicamente por dois tipos de métodos: aqueles ditos exatos e os chamados métodos aproximados. Os métodos exatos garantem que a solução ótima para o problema é encontrada, com a desvantagem que os algoritmos gerados a partir desses métodos são, em geral, computacionalmente custosos, o que limita suas utilizações a instâncias pequenas. Alguns métodos dessa classe que ainda são usados, como por exemplo, *branch-and-bound* e planos de cortes (APPLEGATE et al., 1998), têm suas fundamentações dentro da programação linear e suas variantes.

Os métodos aproximados apesar de não garantirem soluções ótimas, possibilitam que instâncias maiores sejam tratadas em menor tempo. Por isso são que têm mais atraído a atenção dos pesquisadores e, portanto conseguido os avanços mais expressivos. Portanto, neste trabalho serão considerados apenas estes métodos.

Desde que o PCV (e suas variantes) passou a ter aplicação no mudo real, que muitas propostas de soluções aproximadas, as únicas que são possíveis para instâncias grandes, começaram a surgir. Isto fez com que aparecessem inúmeros trabalhos, publicados recentemente na literatura, que propõem soluções para o PCV, sendo que cada um dos modelos proposto apresenta recordes em números de cidades, com algumas restrições sobre as instâncias, ou com tempo recorde de processamento para instâncias presentes na literatura com suas soluções ótimas conhecidas, ou ainda com novo recorde de otimalidade para instâncias ainda em estudo.

Apresentar resultado novo, especialmente para as instâncias ainda em aberto é uma tarefa difícil. A diversidade de aplicações cada vez mais complexas presentes na solução

desse problema, dificulta em parte esta tarefa. Esta dificuldade está associada à demanda crescente por soluções de instâncias cada vez maiores, uma vez que grande parte dos problemas de interesse prático é grande, e até muito grande, além de exigir uma demanda com crescimento proporcional ao tamanho da instância por recursos computacionais, que ainda é limitado.

Matematicamente o que se tentou até hoje sem sucesso, foi estabelecer um método que transformasse este e outros problemas do tipo fatorial (e exponencial) em problemas do tipo polinomial considerado de solução fácil. Enquanto, não se consegue o método na matemática exata, ou pelo menos saber se o mesmo existe para resolver os problemas do tipo NP-Completos, desviou-se o interasse por soluções aproximadas, trabalhando com busca heurística, juntando (hibridização) dois ou mais algoritmos na tentativa de encontrar uma combinação que resulte em solução cada vez mais perfeita, isto é, com garantia de 100% para a ótima, principalmente para instâncias grandes.

Procurar essa combinação perfeita para garantia do ótimo, até parece um problema isolado, em que a solução interessa apenas a este problema em si. No entanto, soluções embora que aproximadas do problema do caixeiro viajante, levará à solução de diversos outros problemas de minimização de custo nas áreas de engenharia, economia, biologia, etc. A seguir, serão tecidos alguns comentários sobre métodos aproximados que, utilizando heurísticas obtiveram sucessos quando aplicados ao PCV.

LIN (1965) foi o primeiro a apresentar solução (embora para instâncias pequenas) para o PCV utilizando o conceito λ-*optimal* (ou simplesmente λ-*opt*, conforme notação original) e também o primeiro, juntamente com KERNIGHAN (LIN & KERNIGHAN, 1973) a propor um algoritmo heurístico para encontrar soluções ótimas ou próximas da ótima para o PCV simétrico, que é considerado um dos métodos mais efetivos, sendo comumente referenciado na literatura como heurística de Lin-Kernighan ou heurística L-K.

Por muito tempo, a heurística L-K foi considerado o melhor método conhecido para solucionar instâncias simétricas do PCV. Mesmo hoje, com o surgimento de tantos novos métodos, a heurística de L-K ainda é bastante empregada como componente básico, ou às vezes até como principal, das novas abordagens, como é o caso da de HELSGAUN (1999), que apenas faz algumas modificações na heurística original.

Muitos problemas de otimização combinatorial como o PCV, segundo LIN & KERNIGHAN (1973) podem ser formulados resumidamente como "encontrar de um conjunto *S*, um subconjunto *T* que satisfaça algum critério *C* e que minimize uma função objetivo *f* ". Para encontrar tal subconjunto é necessário empregar algum mecanismo de transformação que possa alterar a rota corrente. Um deles pode ser mudança de arco (considerando o uso de ferramenta da teoria dos grafos) proposto por CROES (1958), que tem sido aplicado em muitos problemas e que consiste em mudar um número fixo de *k* arcos de *T* com *k* arcos de $\underline{S - T}$, de tal modo que o resultado obtido *T'* é possível e melhor.

Para se entender melhor as abordagens aqui apresentadas, é interessante ter alguma compreensão sobre a idéia básica para a construção da heurística de Lin-Kernighan que é baseado na estratégia λ-*opt*.

LIN & KERNIGHAN (1973) perceberam que: se *T* é uma possível solução e não é ótima, é porque existem *k* elementos (ligações) $x_1, x_2, x_3,..., x_k$ 'fora do lugar'. Para que esta *T* se torne ótima, os x's deveriam ser substituídos por *k* elementos (ligações) $y_1, y_2, y_3,..., y_k$ do conjunto $S - T$ (*S* conjunto de todas as rotas e que *T* e $S - T$ têm o mesmo número de ligações). O problema se resume simplesmente em encontrar *k* e os x's e os y's. Visto que não se conhece o valor de *k*, parece artificial fixar um valor para *k* e então considerar todo possível *k*-subconjuntos de *T* para este *k* fixo.

A heurística L-K não especifica o valor de *k* com antecedência, já que considera desvantajoso, pois o esforço computacional aumenta rapidamente com o aumento de *k* e também a dificuldade em encontrar um *k* que proporcione o máximo de equilibre entre o tempo de execução e a qualidade da solução. Este talvez seja o grande diferencial, ou até o principal fator de sucesso, entre a heurística L-K e as demais que em sua maioria fixam o valor de *k*. Como não se sabe antecipadamente nem o valor de *k* nem de x's e y's, tenta-se encontrá-los um a um. Assim procura-se primeiro pelo par (x_1, y_1) que esteja 'mais fora do lugar'. Então com x_1 e y_1 selecionados, procura-se pelo par (x_2, y_2) que também esteja 'mais fora do lugar' e, assim por diante.

Transportando esta idéia para o PCV. Considere uma rota *T* com custo *f(T)* e qualquer outra rota *T'* com custo $f(T') < f(T)$. Suponha que *T* e *T'* (como conjuntos de *n* links) diferem por *k* links. A idéia é tentar transformar *T* em *T'* pela identificação seqüencial dos *k*

pares de links que podem ser mudados entre *T* e *S* – *T*. Isto é, tentar encontrar dois conjuntos de links X = {x_1, x_2, x_3, ..., x_k} e Y = {y_1, y_2, y_3, ..., y_k} tal que, se os links X são retirados e substituídos pelos links Y, o resultado é uma rota de custo menor.

A heurística de HELSGAUN (1999) gera soluções ótimas para diversas instâncias do problema do caixeiro viajante e, dentre estas, a solução considerada em tempo ótimo para o problema das 13.509 cidades. Nessa heurística, a de HELSGAUN (1999), observa-se uma derivação da heurística de Lin-Kernighan onde se tem além de alteração na estratégia de busca original, pois são usados maiores e mais complexos passos de busca, o uso de uma análise mais sensível para direcionar e restringir a busca. Os tempos obtidos para a solução do problema do caixeiro viajante para várias instancias de cidades (incluindo a das 13.509), entre esta versão e a original são bem parecidas.

No entanto, com a variante de HELSGAUN (1999) foi obtida solução ótima para todos os problemas com ótimo conhecido, além de soluções melhores para uma série de problemas grandes com ótimo ainda é desconhecido, como é o caso de 85.900 cidades. Esta variante foi avaliada em várias classes de problemas: simétrico, assimétrico, ciclo hamiltoniano e patológica. A patológica é uma classe especial de instâncias para o PCV construída por PAPADIMITRIOU & STEIGLITZ (1982), onde a heurística L-K aparece muito ineficiente.

Experimentos computacionais mostraram que o novo algoritmo apresentado na abordagem de HELSGAUN (1999) é altamente eficiente. Sua eficiência é conseguida por meio de uma eficiente estratégia de pesquisa. A pesquisa é baseada em movimentos *5-opt* restringida por uma escolha cuidadosa dos conjuntos candidatos (conjuntos de links X e Y).

4.5 Método proposto

Como já mencionado anteriormente, não é intenção do método proposto por este trabalho bater novo recorde em número de cidades, nem em tempo de processamento ou garantia da solução ótima, mesmo porque não existe uma abordagem (*Simulated Annealing*, *tabu search* ou algoritmo genético), que contemple todos esses itens ainda que modelada e

aplicada especificamente a um problema. Isso é mais afirmativamente verdadeiro se considerar que tais abordagens têm a pretensão de apresentarem-se como de propósito geral.

4.5.1 Descrição do método

Em um projeto que envolva a aplicação do algoritmo SA, além da definição do conjunto de parâmetros específicos do problema a ser tratado, com já visto anteriormente, também devem ser determinados os parâmetros associados com o esquema de controle da temperatura. A maioria dos projetos adota o esquema original, onde a temperatura T é retirada de uma faixa definida pelos limites superior (valor inicial da temperatura) e inferior (valor que a temperatura deve alcançar, normalmente zero ou próximo de zero).

Como o SA é um procedimento iterativo, uma dada temperatura é mantida constante por um número predeterminado de repetições, para então ser reduzida através de sua multiplicação por um valor constante, denominado fator de redução. O número de repetições para uma dada temperatura, que na nomenclatura do SA é para atingir o equilíbrio, pode ser visto dentro do contexto de uma solução para o problema sendo tratado, como sendo o número de vizinhos que devem ser alcançados. Após esse número de repetições, a temperatura é reduzida e um novo ciclo se inicia, até que a temperatura alcance um valor mínimo, normalmente próximo de zero.

4.5.2 Múltiplas faixas de temperatura

Qualquer algoritmo que utilize como paradigma a abordagem *Simulated Annealing* tem os valores da temperatura retirados de uma única faixa, conforme mostra a figura 4.1. O algoritmo inicia com uma temperatura T_0, geralmente alta, e vai diminuindo a cada iteração através do fator de redução, até atingir um valor de temperatura mínimo, normalmente próximo de zero.

Faixa única significa que os valores assumidos pela temperatura são retirados de uma faixa compreendida entre os valores inicial e final da temperatura que vai diminuindo cada vez que o sistema se equilibra. Com isso, é possível encontrar a quantidade de vezes que o sistema alcança o equilíbrio, já que está relacionado como os valores inicial e final da temperatura e do fator de redução, que matematicamente de forma simples pode ser calculada.

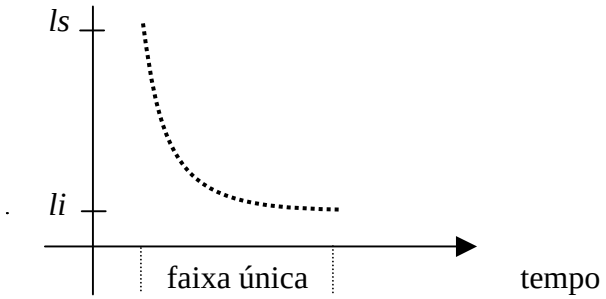

Figura 4.3 – Temperatura retirada de uma única faixa

No método proposto, o valor da temperatura continua constante para um determinado número de repetições, só que agora a temperatura não percorre somente uma faixa de valores entre as temperaturas inicial e final, mas com várias faixas compreendidas entre estes dois valores.

Para tratar com as múltiplas faixas são definidos dois intervalos. Um superior, que tem os valores *ls* e *lls* representando, respectivamente, o limite superior e inferior do intervalo, e um inferior, com os valores *li* e *lli* representando o limite superior e inferior do intervalo, respectivamente.

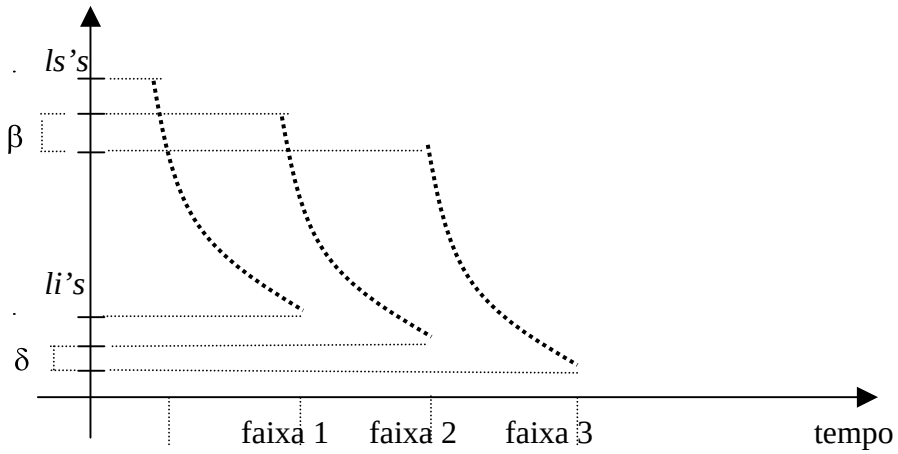

Figura 4.4 - Temperatura retirada de três faixas

No intervalo superior estão os valores iniciais que a temperatura assume em cada uma das faixas. Cada um desses valores que a temperatura assume no intervalo superior é determinado em função dos valores do limite e do número de faixas que será usada. Descrição idêntica tem o intervalo inferior, só que representando os valores mínimos que a temperatura assume em cada faixa. A figura 4.4 ilustra tal situação.

Durante a execução do algoritmo duas constantes passam a controlar a redução dos valores iniciais e finais da temperatura em cada faixa denominadas β e δ. Os valores das constantes β e δ serão determinados pelo número de faixas que é utilizada no processo, através das fórmulas 4.3.1 e 4.3.2. Em cada faixa o valor da temperatura *T* se inicia com *ls* e é reduzido até *li* utilizando a expressão $T = \alpha * T$, onde α é o fator redutor. Após concluir a varredura dos pontos ao longo de uma faixa, a temperatura salta abruptamente para o limite superior da nova faixa que é obtido através pela expressão $ls = ls * \beta$, enquanto que o limite inferior pela $ls = ls * \delta$.

$$\beta \text{ (beta)} = exp\,(\,(\,ln\,(\,lls\,) - ln\,(\,ls\,)\,)\,/\,nfaixa\,) \qquad (4.3.1)$$

$$\delta \text{ (gama)} = exp\,(\,(\,ln\,(\,lli\,) - ln\,(\,li\,)\,)\,/\,nfaixa\,) \qquad (4.3.2)$$

Na figura 4.3 é apresentado um esboço em pseudo-código do procedimento proposto pelo método que praticamente imita o procedimento original, da figura 3.3, já apresentada.

Trata-se basicamente da inclusão de mais um laço que realiza o controle das faixas de temperaturas, como também os fatores de redução para cada faixa.

O laço se inicia com os valores superior e inferior da temperatura para a primeira faixa, como no SA básico. No final da execução desse laço, uma nova iteração ocorre com novos valores superior e inferior para a temperatura, calculados pelos comandos *ls = ls* * β e *li = li* * δ, para a uma nova faixa.

Procedimento do método proposto

```
início
    ...
    T    ← ls;
    T_Min ← li;
     para FAIXA de 1 até  NFAIXAS faça
        enquanto ( T > T_Min ) faça
        para interações para equilíbrio faça
            Gerar uma solução S' da N(S);
                Avaliar variação de energia – ΔE ← f(S') – f(S);
                 se ΔE < 0 then
                        S ← S';
                 senão
                        Gerar u ∈ random[0,1];
                        se u  <  exp(- ΔE / K_B * T) then
                              S ← S';
                        fimse
                 fimse
            fimpara
            Reduzir a temperatura T;
        fimenquanto
        ls ← ls * β;
        li ← li * δ;
        T    ← ls;
            T_Min ← li;
        fimpara
fimprocedimento
```

Figura 4.5 – Pseudo-código do algoritmo proposto pelo método

Após o teste de aceitação, uma solução melhor pode surgir se tornando a solução corrente. Em cada temperatura, um equilíbrio deve ser alcançado por meio de uma pesquisa

exaustiva na vizinhança da solução corrente. Como o procedimento é interativo, após cada equilíbrio, a temperatura é reduzida até que alcance T_{Min}, quando então o procedimento se reinicia com mudanças bruscas nos novos valores da temperatura que é feita quando os valores de *ls* e *li* são calculados, já dentro da próxima faixa.

Acredita-se que essas mudanças abruptas em cada nova faixa podem conduzir a busca para região do espaço onde existe grande chance de se encontrar boas soluções. Segundo MAZZUCCO JUNIOR (1999), essas perturbações, causadas pelos aumentos bruscos nas temperaturas nos limites inferiores de cada faixa, têm como objetivo procurar evitar convergência para solução ótima local. Elas permitem a realização de modificações acentuadas nas soluções correntes. É uma estratégia simples que procura desviar a busca de regiões já intensamente explorada, dirigindo a procura para uma nova região do espaço de soluções. Como visto anteriormente, o algoritmo SA utiliza os movimentos conhecidos por *uphill* com este mesmo objetivo, porém, sem a intenção de deixar a região de busca corrente.

Em LOURENÇO (1995), é proposta uma abordagem que utiliza uma outra estratégia, cujo objetivo é também abandonar regiões do espaço de soluções intensamente exploradas, onde o passo largo é feito por métodos de otimização. O algoritmo empregado no passo largo fica responsável pela modificação acentuada realizada na solução corrente. Cabe ao método de otimização de busca local, o refinamento da solução encontrada no passo largo.

MARTIN & OTTO (1995) argumentam sobre as vantagens em aplicar *Simulated Annealing* junto com algum método de pesquisa local. Seu método chamado de otimização local em cadeia (*Chained Local Optimization – CLO*) inicia com uma solução localmente ótima, obtida pela aplicação de algum método de pesquisa local, quando é aplicada uma perturbação que resulta numa configuração que eles chamam de intermediária. É aplicado um novo método de melhoramento, para então aplicar o teste de aceita/rejeita do *Simulated Annealing*. Este método pode ser visto como uma generalização da cadeia de Markov a passos largo do SA.

Neste trabalho será feita uma avaliação comparativa do algoritmo aplicado ao PCV. Serão comparados os resultados obtidos com uma faixa, que é a implementação básica do SA, com os resultados obtidos com várias faixas que é a idéia do método proposto. Essa

diferença, em termos de faixas, em nada prejudica uma comparação em termos de desempenho. Os testes serão realizados sempre com as mesmas instâncias.

4.5.3 Implementação

A implementação do algoritmo foi feita na linguagem de programação Pascal, micro computador PC compatível, processador da família Pentium II Intel, com clock de 400 Mhz, como 64 Mb de memória e ambiente operacional Windows 98.

O algoritmo utiliza como base a abordagem *Simulated Annealing* para solucionar instâncias moderadas do PCV, dentro do conceito de soluções aproximadas, uma vez que implementa a heurística *2-opt*, proposta por LIN (1965), como mecanismo de perturbação. A implementação é bastante simples, tanto em termos de funcionalidade dos procedimentos, como pelas estruturas de dados que utiliza, o que satisfaz com um dos atributos que BALL & MAGAZINE (1981) *apud* LAWLER et al. (1985) que reconhecem como um critério válido para qualquer algoritmo que implemente heurística, *implementação fácil*.

4.5.3.1 Estrutura de dados

Utilizar a estrutura de dados adequada é uma questão crucial para o bom desempenho de qualquer *software*. A representação da rota é uma questão central da implementação. A escolha da estrutura de dados adequada pode ter um grande impacto na eficiência do algoritmo. É claro que o maior gargalo do algoritmo é a pesquisa para um possível movimento (mudança de arcos) e a execução de tal mudança na rota. Portanto, cuidados especiais devem ser tomados para escolher uma estrutura de dados que permita execução rápida dessas operações.

FREDMAN et al. (1993) fazem um extenso comentário sobre o comportamento em termos de eficiência das principais estruturas de dados que podem ser utilizadas em

algoritmos heurísticos para resolver o PCV. Examinam seus desempenhos tanto do ponto de vista teórico quanto experimental. Dentre essas estruturas de dados está a representação tradicional que é a baseada em vetor (*array)*. A representação da rota em estrutura de vetor, segundo FREDMAN et al. (1993), permite que a ordem relativa das cidades sejam determinadas em um tempo constante pequeno, mas requer tempo Ω(n) para o pior caso (onde n representa o número de cidades) para implementar o procedimento de troca de arcos, o que o torna impraticável para instâncias grandes.

O algoritmo analisado neste trabalho utiliza uma estrutura de dados bastante simples, sendo as principais as que definem uma cidade em termos de coordenados no plano e aquela que representa uma rota, conforme relatado a seguir.

a) Cidade: uma cidade é definida por duas coordenadas do tipo real que representam a localização da cidade em relação ao plano no qual as mesmas de encontram, que na nomenclatura da linguagem Pascal será representada por uma estrutura de dados do tipo registro com a seguinte notação.

type *cidade* = r**ecord**
x, y **: real**
end

b) Rota: uma rota de *n* cidades é definida como uma seqüência de cidades que devem ser visitadas pelo viajante, que na nomenclatura da linguagem Pascal será representada por uma estrutura de dados do tipo vetor de *n* elementos do tipo cidade com a seguinte notação.

type *rota* **= array [***1...n***] of** *cidade*

Embora haja conveniências na escolha da forma de representação do problema, algumas formas de representação facilitam mais o entendimento do problema, enquanto outras favorecem mais a sua solução o que deixa o programador bastante à vontade nessa escolha.

4.5.3.2 Os procedimentos

A seguir serão apresentados os principais procedimentos que merecem destaque pois são críticos dentro do contexto geral da implementação. São procedimentos pequenos e bastante funcionais, que os tornam de fácil entendimento.

Como o algoritmo proposto pelo método tem uma forma bastante simples tanto em termos de estrutura de dados como em termos de implementação dos procedimentos, já que não envolve estruturas dinâmicas, serão abordados apenas aqueles mais importantes dentro do contexto, ficando os procedimentos que sofreram alterações, comentados a seguir.

a) Procedimento *Distância*.

Este procedimento calcula a distância entre duas cidades quaisquer através de suas coordenadas utilizando à métrica euclidiana.

```
function distancia( p,q : cidade ) : double;

        var dx,dy : real;

begin
        dx := q.x - p.x;
        dy := q.y - p.y;
        distance := sqrt(dx*dx + dy*dy)
end;
```

b) Procedimento *Custo*.

O vendedor visita as cidades em ordem numérica 1,2,…,n antes de retornar para cidade de número 1. O custo da rota é a soma das distâncias entre as cidades sucessivas. O procedimento abaixo calcula este custo.

```
function custo(a : rota) : real;

        var i : integer;
          sum : real;

begin
        sum := distancia(a[n], a[1]);
        for i:= 1 to n-1 do
              sum := sum + distancia(a[i], a[i+1]);
        custo := sum
end;
```

c) Procedimento *Annealing.*

Este é o principal procedimento do algoritmo que inicia com uma temperatura alta T_{max} que é lentamente reduzida num número fixo de passos. A cada passo, a temperatura é mantida constante enquanto se pesquisa por uma rota de custo menor. A temperatura é então reduzida por um fator α (*alfa*) que deve ser ligeiramente menor que 1.

```
procedure anneal(var a:rota; Tmax,alhpa : real;
                 passos,tentativas,mudanças : integer);

        var T : real;
              k: integer;

begin
        T := Tmax;
        for k := 1 to passos do begin
              search(a,T,tentativas,mudanças);
              T := alpha * T
        end
  end;
```

d) Procedimento *Pesquisa.*

Para se chegar próximo ao equilíbrio numa determinada temperatura *T*, deve-se examinar muitas possíveis rotas antes reduzir a temperatura. A *pesquisa* continua até que um número fixo de mudanças tenha sido aceitas. Em temperaturas altas, muitas mudanças são aceitas. Em temperaturas baixas muitas mudanças aleatórias são provavelmente rejeitadas, visto que elas aumentam o custo da rota que neste momento já é bastante pequena. Para limitar a pesquisa por rotas de custo menor em baixas temperaturas o algoritmo limita o número de tentativas. Este procedimento também limita o número de mudanças que possam ocorrer numa determinada temperatura.

```
procedure Pesquisa(var a: route; T: real;
                   tentativas, mudanças : integer);

        var i,j,na,nc: integer;
                     dE: real;

begin
        na := 0;
        nc := 0;
        while (na < tentativas) and (nc < mudanças) do begin
              seleciona(a,i,j,dE);
```

```
                if aceita(dE,T) then begin
                      troca(a,i,j);
                      nc := nc + 1
                end;
                na := na + 1
          end
end;
```

e) Procedimento *Seleciona*.

O procedimento seleciona randomicamente duas cidades e considera a possibilidade de trocá-las, computando a variação da energia causada pela reversão da rota da cidade de número *si* para a cidade de número *j* (2-*opt)*. A energia trocada é computada em tempo constante usando as coordenadas das duas cidades e de suas sucessoras, não havendo necessidade de calcular o custo total de nenhuma rota, mas apenas verificar se teve ganho ou perda de energia com os arcos revertidos.

```
procedure seleciona (n : integer; var t : tour;
                        var si, j : integer;
                          var dE : double );

     var i, sj : integer;

begin
        gera ( n,  i, j );
        si := i mod n+1;
        sj := j mod n+1;
        if (  i <> j )   then
        dE  :=   distancia (  t [  i ], t [  j ] ) +
                        distancia (  t [ si ], t [ sj ] ) -
                        distancia (  t [  i ], t [ si ] ) -
                        distancia (  t [  j ], t [ sj ] )
        else
        dE  :=  0.0
end; { seleciona }
```

f) Procedimento *troca.*

Este procedimento, que faz efetivamente a mudança na rota e define o trajeto de reversão entre as cidades *i* até *j*. O número de cidades no trajeto é denotado por n_{ij}. O processo de reversão se inicia pelas extremidades do trajeto e caminha para o meio. O critério usado é uma variante da idéia de LIN (1965), onde de uma rota, por exemplo, c_1, c_2, c_3, ..., c_n são selecionadas randomicamente duas cidades c_i e c_j com suas sucessoras representadas respectivamente por c_{si} e c_{sj}. Assim a rota ficaria

..., c_i, c_{si}, ..., c_j, c_{sj}, ...

A nova rota é gerada pela reversão na ordem das cidades de c_{si} a c_j, que passa agora a ter a seguinte configuração:

..., c_i, c_j, ..., c_{si}, c_{sj}, ...

Este mecanismo provoca o mesmo efeito numa rota que a heurística *2-opt*.

```
procedure troca ( n : integer; var  t : tour ; i, j : integer );

    var k, nij : integer;

begin
        nij := ( j - i + n ) mod n + 1;
        for k := 1 to nij div 2 do
              swap (                          t,
                    ( i + k - 2 ) mod n + 1,
                    ( j - k + n ) mod n + 1   );
end; { troca }
```

g) Procedimento *aceita*.

O critério de aceitação pode ser determinístico ou probabilístico. Como se trata de uma abordagem probabilística, logo será usado o critério probabilístico. Uma rota de custo menor ou que não sofreu mudança será sempre aceita. Uma rota de custo maior será aceita com probabilidade $e^{-dE/T}$ que é comparada com um número randômico gerado entre 0 e 1.

```
function aceita ( dE, temp : double ) : boolean;

    var r:double;

begin
        if ( dE > 0.0 ) then begin
              r := random;
              aceita := exp ( - dE / temp ) > r;
  end
        else
              aceita := true;
end; { aceita }
```

4.5.3.3 Implementação das modificações do método proposto

Como já aludido anteriormente, o algoritmo proposto pelo método se mantém praticamente idêntico ao apresentado acima que é uma implementação básica do *Simulated Annealing*, apenas com alterações no procedimento *Annealing* com a inclusão de mais um laço para controle das múltiplas faixas de temperatura, além do acréscimo de alguns comandos que implementam as mudanças de faixas na temperatura. Com estas modificações o procedimento *Annealing* passa a ter a seguinte configuração.

```
procedure annealing ( n : integer; var  t : tour ;
                      Tmax, alfa : double ;
                      tentativas, mudanças : integer;
                      ls,li,lls,lli : double; nfaixa : integer );

     var
               k, faixa  :  integer;
               beta, gama, temp : double;

begin
         beta := exp ( ( ln ( lls ) - ln ( ls ) ) / nfaixa );
         gama := exp ( ( ln ( lli ) - ln ( li ) ) / nfaixa );
         for  faixa :=  1  to  nfaixa  do begin
               temp := ls;
               while ( temp >= li) do begin
                     pesquisa ( n, t, temp, tentativas,mudanças
          );
                     temp  := temp * alfa;
          end;
          ls :=  ls * beta;
               li :=  li * gama;
     end;
end; { anneal }
```

CAPÍTULO 5

Análise dos Resultados

Inicialmente é apresentada a metodologia de como será avaliado o método proposto nesse trabalho, assim como as formas de se obter os resultados. Também é falado sobre as origens das instâncias e procedimentos de como construir a rota inicial. Por último, os resultados computacionais obtidos com a aplicação do algoritmo com uma faixa de temperatura e com múltiplas faixas, que o método proposto, são apresentados.

5.1 Metodologia

Dentre as três maneiras (*analise do pior caso, análise probabilística e análise empírica*) de avaliar a performance de algoritmos aproximados que propõem soluções para o PCV, a *análise empírica* é a mais simples e, portanto, a mais utilizada. Todas têm seus pontos fortes e fracos. Nos casos das abordagens do pior caso e probabilística, o envolvimento matemático pode ser bastante complexo e os resultados alcançados por estes métodos podem muitas vezes ser de pouca utilidade quando na solução de instâncias práticas para o PCV. Por ser de menor complexidade a abordagem do pior caso é também bastante utilizada. Vários dos algoritmos aproximados conhecidos para o PCV têm sua análise no pior caso um tanto pobre (HELSGAUN, 2000). Portanto, a análise empírica parece ser a mais apropriada e é a que será utilizada.

Neste trabalho, o método proposto é avaliado através de um algoritmo derivado do *simulated anneling*, onde ao invés das temperaturas serem tiradas de uma única faixa, como na abordagem original, são retiradas de múltiplas faixas. Esta avaliação se dará em relação

ao desempenho do algoritmo em relação às várias faixas de temperatura quando aplicado aos mesmos problemas testes. O conjunto de problemas testes será dividido em duas categorias, sendo uma delas a mais utilizada para este tipo de análise por tratar de instâncias com dados de situações reais e soluções conhecidas. A outra nem tanto utilizada, mas com a característica de também se conhecer a solução. O que determina uma categoria de problemas testes, neste caso, é a origem da instância, conforme será explanado mais adiante.

Os algoritmos (implementação básica e com múltiplas faixas) serão executados várias vezes para os vários tamanhos de instâncias das duas categorias. Em cada execução será feito um número de tentativas igual ao do tamanho da instância, isto é, o número de cidades. Toda execução, independente da categoria, terá sua rota inicial gerada pela heurística que mostre melhor desempenho para aquele determinado tamanho de instância, portanto, as várias tentativas terão a mesma rota inicial naquela rodada.

5.1.1 Avaliação

Será avaliado o melhor resultado obtido de todas as execuções para um determinado tamanho de instância dentro da categoria. Este dado fornece a *qualidade da solução* que é medida em termos de percentagem a diferença entre a solução encontrada e a melhor solução ótima conhecida, e que é dada pela fórmula:

$$Q^{10} = \frac{f(.) - f_{opt}}{f_{opt}} \times 100, \text{ onde} \qquad (5.1)$$

$f(.)$ é o custo da solução encontrada pelo algoritmo e,

f_{opt} o custo da melhor solução ótima conhecida para aquela instância.

[10] Esta fórmula é a mais encontrada na literatura para avaliar performance de algoritmos aproximados.

Outro resultado que será também avaliada é a *performance*, que mede o número de vezes que o algoritmo numa determinada faixa alcançou a solução ótima, se acontecer, através da seguinte fórmula:

$$P = \frac{Num_{opt}}{Num} \; x \; 100, \text{ onde} \qquad (5.2)$$

Num_{opt} é o número de avaliações que alcançaram o ótimo global e *Num* o número total de avaliações.

Outro método freqüentemente usado no estudo para avaliar performance de heurísticas para PCV é resolver instâncias de problemas que consiste de pontos distribuídos randômicos dentro de um quadrado, também sobre a métrica euclidiana. Por esse método, as instâncias são resolvidas para valores crescentes de n e comparado a um valor teórico para o custo esperado da rota ótima. Uma fórmula bem conhecida, proposta por HELP & KARP (1970), é $L_{opt}(n,A) = K \sqrt{nA}$ quando n cidades estão distribuídas uniformemente e randomicamente sobre um área quadrada de A unidades. Experimentos de JOHNSON et al. (1996) sugerem que K tem um valor aproximado de 0,7124.

5.1.2 Origem das instâncias

As diversas instâncias do PCV que são encontradas têm suas origens baseadas nas mais diversas situações, que tanto pode representar situações do mundo real como imaginária, simplesmente para estudo que podem representar problemas com ou sem restrições. Dentre os problemas, os que mais têm causado interesse, principalmente em função de sua aplicabilidade, em situações práticas, é o que trata de instâncias euclidianas (o problema é chamado euclidiano), ou seja, as cidades correspondem a pontos no plano e as distâncias são calculadas na métrica euclidiana, que tem logicamente como restrição à desigualdade triangular e a simetria.

As instâncias utilizadas neste trabalho têm duas fontes, a TSPLIB que contém instâncias de diversas origens e que também é as mais utilizadas, pois apresentam soluções ótimas conhecidas para a grande maioria delas; a outra não tão comum, mas citada em TIAN (1999), pode ser considerada uma variante das instâncias onde as cidades são pontos randômicos distribuídos num quadrado, onde têm também suas soluções ótimas conhecidas.

a) **Instâncias da TSPLIB**

A grande maioria das instâncias do PCV é de natureza geométrica, portanto obedecendo a métrica euclidiana. A principal fonte de tais instâncias é a TSBLIB que é uma biblioteca de instâncias para o PCV (TSP em inglês) colecionado por REINELT (1991) *apud* HELSGAUN (2000) e disponível via internet através do endereço eletrônico ftp://softlib.rice.edu/pub/tsplib. Lá também podem ser encontrados os resultados das mais recentes pesquisas sobre o assunto. A biblioteca contém instâncias do PCV de várias fontes e com várias propriedades com o número de cidades variando de 14 a 85.900, incluindo muitas aplicações sobre a fabricação de *chip* VLSI, bem como instâncias geográficas baseadas nas distâncias de cidades reais.

b) Instâncias em grade

Embora as instâncias em grade não sejam utilizadas comumente para testes de algoritmos, tem uma característica bastante interessante que é a possibilidade de se poder determinar a rota de custo ótimo. Nas instâncias em grades, as cidades são colocadas nos vértices de uma grade regular quadrada no espaço euclidiano. A figura 5.1 mostra uma grade com 16 cidades. À distância entre duas cidades é determinada pelo comprimento do lado do quadrado ou pela distância da diagonal quando for o caso.

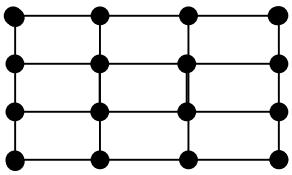

Figura 5.1 – Uma grade 4 x 4 que corresponde a 16 cidades

Para o PCV com as cidades distribuídas dessa forma, a solução ótima (a rota de menor tamanho) para uma determinada instância pode ser facilmente determinada já que o

custo f_{opt} corresponde às somas das distâncias que pode simplesmente ser calculado em função do tamanho *g* do lado do quadrado, nos dois casos seguintes:

$$f_{opt} = \begin{cases} gn, & n = \text{número par de cidades,} \\ g(n-1+v2), & n = \text{número impar de cidades.} \end{cases} \quad (5.3)$$

5.1.3 Rota inicial

Como se trata de uma abordagem (tendo o algoritmo *simulated anneling* como base) que exige uma rota completa válida para iniciar a execução do algoritmo, decidiu-se iniciar fazendo uma avaliação de qual procedimento de construção de rota inicial seria mais adequado levando-se em consideração aos respectivos tamanhos de instâncias dentro das categorias, ou seja, qual o procedimento de construção de rota seria melhor para gerar a rota inicial.

Os procedimentos descritos a seguir para gerar a rota inicial, também são os mais utilizados nas principais abordagens de sucesso presentes na literatura, que são: o método de geração randômico e método heurístico de construção de rotas, o vizinho mais próximo.

a) Método de geração randômico. Neste método, como o próprio nome diz, a rota inicial é gerada através de uma escolha aleatória da seqüência de cidades que deverá ser visitada pelo caixeiro. Inicialmente a rota está vazia, então são escolhidas aleatoriamente uma a uma as cidades que comporão a seqüência da rota, dentro de um limite, obviamente (o número de cidades), até que todas as cidades estejam presentes na rota.

b) Vizinho mais Próximo. Este é uma dos métodos heurísticos para construção da rota inicial mais utilizados na maioria das abordagens com melhoramento de rota. Vê-se mais detalhe sobre este método na secção 2.3.4.1.

5.2 Experimentação

Para um melhor entendimento sobre a implementação do algoritmo onde algumas de suas rotinas estão descritas acima (5.2.2), é que o mesmo será chamado de algoritmo básico, uma vez que tenta transcrever a abordagem *Simulated Annealing* em sua forma genérica. Para realização dos experimentos, apenas uma rotina deste algoritmo será modificada para que seja possível trabalhar como mais de uma faixa de temperatura.

5.2.1 Construção da rota inicial

Conforme mencionado antes, era intenção realizar um conjunto de testes para se determinar qual procedimento de construção de rota inicial (vizinho mais próximo ou geração randômica) seria utilizado para cada uma das instâncias dos problemas teste, o que findou não acontecendo.

Para efeito de ajuste inicial dos parâmetros do programa de resfriamento, utilizou-se como rota inicial aquela gerada a partir da leitura seqüencial das coordenadas das cidades, isto é, o primeiro par de coordenada lida corresponderia a primeira cidade da rota, o segundo par de coordenada lida corresponderia a segunda cidade da rota, e seguindo desta forma se completaria com as demais cidades a rota inicial.

Após os ajustes iniciais dos parâmetros, iniciou-se a execução de algumas instâncias do problema tendo a rota inicial obtida através de procedimentos heurísticos. Os testes mostraram que apesar de existir uma melhoria significativa no custo da rota inicial para a maioria das instâncias quando se utilizou a heurística “vizinho mais próximo” ou “geração randômica” comparativamente com a gerada seqüencialmente, não houve nenhum benefício acentuado em relação ao custo da rota final. Justificativa suficiente para não usar nenhuma heurística na construção da rota inicial na realização dos testes, ou seja, a rota inicial foi gerada automaticamente de forma seqüencial quando da leitura das coordenadas das cidades, o que comunga com os argumentos de LIN & KERNIGHAN (1973) que diz

que utilizar heurística para construir a rota inicial, em alguns casos é desperdício de tempo já que rotas (soluções) construídas são comumente determinísticas e que pode não ser possível obter mais que uma rota inicial.

5.2.2 Ajustando os parâmetros

Resolvida questão da rota inicial, partiu-se para o ajuste dos parâmetros do programa de resfriamente (*cooling schedule*) que é considerado o ponto mais sensível quando se utiliza a abordagem *Simulated Annealing*. Para se conseguir chegar aos primeiros valores dos parâmetros utilizou-se o algoritmo com a implementação básica. Nesta etapa os testes foram realizados com o intuito de adequar os valores dos parâmetros do programa de resfriamento conhecidos na literatura às classes de problemas do caixeiro viajante experimentados, com suas respectivas instâncias.

Neste sentido, foram realizados vários testes com diversas instâncias do problema em ambas as classes. Alguns valores dos parâmetros[11], como número de passos, números de tentativas e número de mudanças foram afinados, através do processo de tentativa e erro de tal modo que se comportassem bem nas duas classes. Entretanto, os valores da temperatura inicial e do fator redução da temperatura em cada iteração sofrem pequenas mudanças quando executados em instâncias de classes diferentes.

5.2.3 Resultados Numéricos

Para avaliar o desempenho do método e encontrar as melhores combinações dos parâmetros do programa de resfriamento várias instâncias do problema foram selecionadas de duas origens diferentes. O primeiro grupo pertence a instâncias do problema disponíveis na internet, conforme endereço já citado e que têm suas soluções ótimas conhecidas. No

[11] Estes parâmetros dizem respeito a essa implementação e não aos da definição do SA.

outro grupo, então instâncias do problema geradas sobre determinadas condições que é a de que as cidades se situam no ponto de interseção das retas que formam uma grade quadrada, com lado conhecido, conforme figura 5.1. Para este grupo, também é possível conhecer a rota ótima pois basta aplicar a fórmula 5.3 já que o comprimento lado e o número de cidades são conhecidos.

5.2.3.1 Desempenho em relação às faixas

Após os valores dos parâmetros estarem definidos no algoritmo com implementação básica, começaram os testes com as mudanças propostas pelo método, uma vez que, as mudanças introduzidas por este não alterariam a performance do mesmo. Com essas mudanças era possível testar o desempenho do método com mais de uma faixa. A execução do algoritmo somente com uma faixa de temperatura tem o mesmo comportamento do algoritmo básico, onde o comprimento da temperatura (*temperature length* - nomenclatura usada por JOHNSON & MCGEOCH – 1997) é representado pelo número de passos.

Para comprovar a eficiência do método proposto, utilizou-se instância do PCV de tamanhos razoável, com 100 e 150 cidades e portanto, de solução não trivial para algoritmo que utiliza SA como abordagem. Sem antecipar qualquer conclusão sobre qual o número de faixa que produz os melhores resultados, decidiu-se centrar os testes em 1, 3, 5, 8 e 10 faixas (escolha aleatória).

Os resultados tabelados em 5.1 e 5.2 representam os valores máximos, médios e mínimos obtidos com cada uma das faixas para as instâncias com 100 e 150 cidades disponíveis na internet. Já os tabelados em 5.3 e 5.4 com instância com 100 e 144 cidades também com as mesmas representatividades, só que geradas (posição das cidades) em grade. Os valores mostrados em cada uma das linhas das tabelas, para as duas classes de instâncias, foram conseguidos após 50 execuções.

	Valor			
Núm. de Fixas	Máximo	Médio	Mínimo	Desvio Padrão
1	21864	21447	21285*	125,84
3	21775	21380	21285	83,31
5	21599	21352	21285	58,81
8	21390	21319	21285	32,85
10	21393	21311	21285	31,08

Tabela 5.1 - Instância com 100 cidades (internet)
* Valor ótimo conhecido.

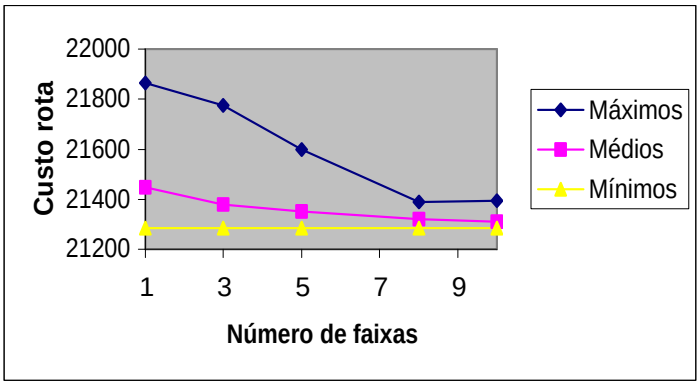

Figura 5.2 – Instância com 100 cidades (internet)

Cabe salientar que todos os testes para cada número de faixa foram realizados de forma independente, ou seja, para se obter o valor máximo para a linha que corresponde a 3 faixas, por exemplo, foram realizadas 50 execuções, todas com 3 faixas de temperatura, que produziram 50 valores diferentes e destes escolhido o maior. Mesmo procedimento para o valor mínimo e, para o médio, a média dos 50 valores. Tais resultados já evidenciam uma tendência de melhores valores à medida que o número de faixas aumenta.

Para melhor visualizar esta tendência observe o gráfico da figura 5.2 com os mesmos valores apresentados na tabela 5.1, onde pode ser observado um padrão de comportamento em relação as três linhas que representam os valores máximos, médios e mínimos, isto é, as linhas decrescem à medida que o número de faixas aumenta, com exceção da dos valores mínimos que se mantém constante.

Apesar de não apresentar o mesmo desempenho da instância com 100 cidades, o método mostrou o mesmo padrão de comportamento para a instância com 150 cidades, isto é, uma tendência de se obter soluções com custo menor à medida que aumenta o número de faixas. Isto pode se visto com os resultados presentes na tabela 5.2 com a instância de 150 cidades proveniente da internet. Uma diferença que pode ser no gráfico na figura 5.3 que representa os valores contidos na tabela 5.2 é que os valores mínimos não se comportam como uma constante, como no gráfico da figura 5.2. Existem diferenças nos valores mínimos e que em apenas uma das faixas, no caso a 8, é apresentado um valor ótimo.

	Valor			
Núm. de Fixas	Máximo	Médio	Mínimo	Desvio Padrão
1	27136	26829	26582	148,17
3	27047	26785	26614	124,36
5	26953	26705	26566	109,55
8	26763	26636	26524*	71,91
10	26824	26707	26563	84,14

Tabela 5.2 - Instância com 150 cidades (internet)
* Valor ótimo conhecido.

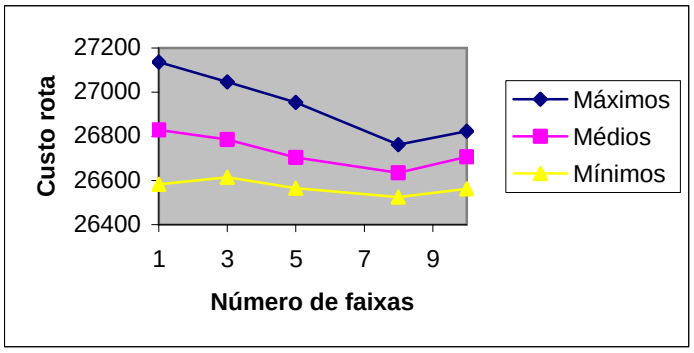

Figura 5.3 – Instância com 150 cidades (internet)

Seguindo a mesma seqüência de resultados apresentados nas tabelas e gráficos acima em relação às instâncias da internet, serão mostrados a seguir resultados com a

mesma representatividade, só que com instâncias geradas em grades. Os valores obtidos estão representados na tabela 5.3 para instância com 100 cidades e na tabela 5.4 para instância com 144 cidades, assim como seus respectivos gráficos das figuras 5.4 e 5.5.

	Valor			
Núm. de Fixas	Máximo	Médio	Mínimo	Desvio Padrão
1	101,6569	100,9941	100,8284	0,3314
3	100,8284	100,6627	100,0000	0,3314
5	100,8284	100,4142	100,0000	0,4142
8	100,8284	100,3314	100,0000*	0,4058
10	100,8284	100,2485	100,0000	0,3796

Tabela 5.3 - Instância com 100 cidades (grade)
* Valor ótimo calculado - fórmula 5.3

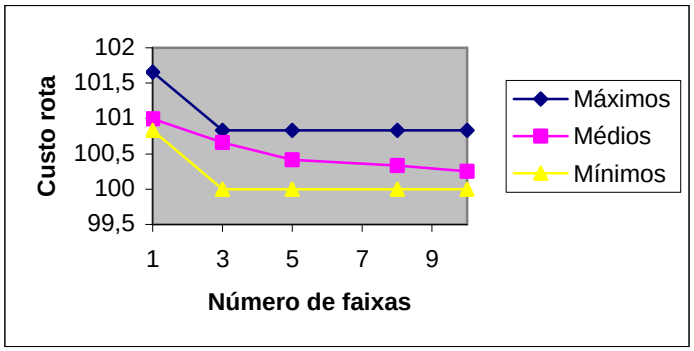

Figura 5.4 – Instância com 100 cidades (grade)

	Valor			
Num. de Fixas	Máximo	Médio	Mínimo	Desvio Padrão
1	148,1421	146,4853	144,8284	0,9075
3	146,4853	145,2426	144,8284	0,5557
5	145,6569	145,0770	144,0000	0,5305
8	145,6569	144,8284	144,0000	0,3314
10	145,6569	144,6627	144,0000*	0,4971

Tabela 5.4 - Instância com 144 cidades (grade)
* Valor ótimo calculado - fórmula 5.3

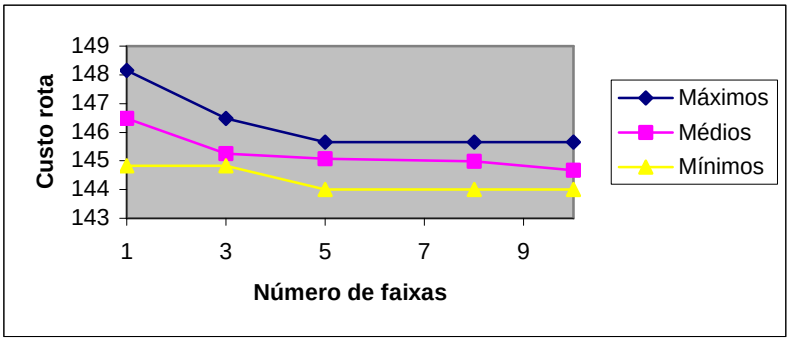

Figura 5.5 - Instância com 144 cidades (grade)

5.2.3.2 Acertos por faixas

Mesmo para uma abordagem típica de SA, obter resultados ótimos para instâncias moderadas não é tão freqüente. Este método mostrou-se bastante otimista ao alcançar muitas soluções ótimas, principalmente à medida que o número de faixa aumenta. Esta taxa de acerto foi obtida executando-se o algoritmo 50 vezes com as mesmas instâncias do item anterior, ou seja, 100 e 150 cidades. O gráfico da figura 5.5 se refere à performance do método com os dados oriundos da internet. O eixo y demonstra, para as 50 rodadas, a quantidade de vezes que o algoritmo alcançou a solução ótima medido em percentual. Valores obtidos por meio da fórmula (5.2).

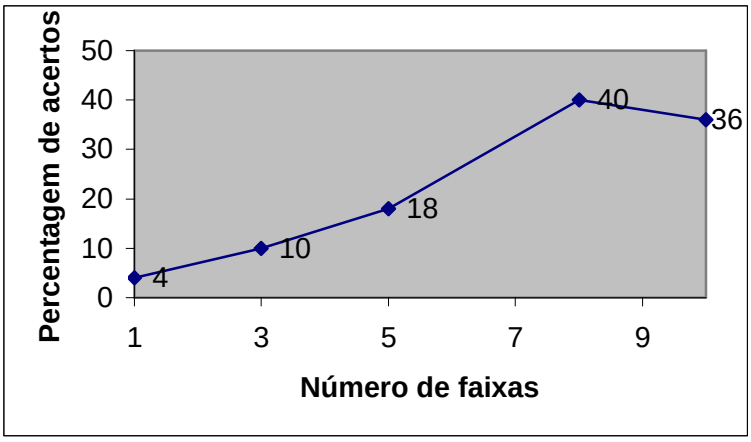

Figura 5.6 - Performance do método com 100 cidades (internet)

O gráfico da figura 5.6 reflete o comportamento do método de que à medida que o número de faixa aumenta, aumenta também o percentual de acerto na solução ótima. Não é apresentado o gráfico sobre a performance do método com a instância de 150 cidades (internet), já que, conforme se observa na tabela 5.2 e no gráfico da figura 5.3, o percentual de acerto para a maioria das faixas é zero, com exceção apenas do número de faixa igual a 8 que apresentou apenas uma ocorrência ótima de solução.

Comportamentos idênticos ao do gráfico da tabela 5.6 pode também ser observado nos gráficos das figuras 5.7 e 5.8 com instâncias em grade para 100 e 144 cidades, respectivamente.

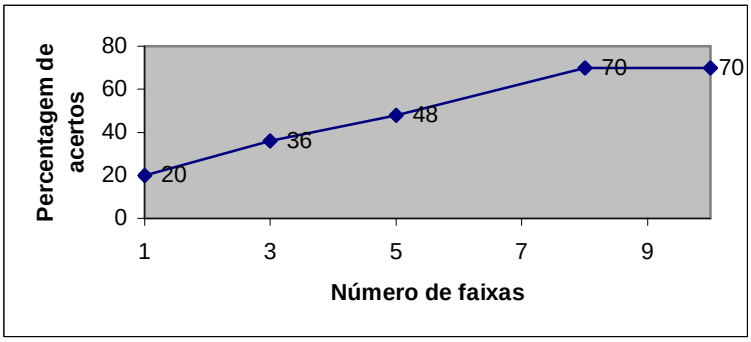

Figura 5.7 - Performance do método com 100 cidades (grade)

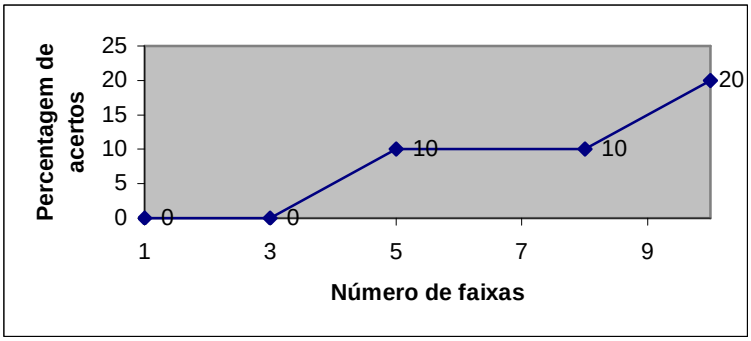

Figura 5.8 - Performance do método com 144 cidades (grade)

5.2.3.3 Qualidade da solução e performance do método

Para comprovar o desempenho do método proposto efetuo-se testes com várias instâncias dos dois grupos, onde é possível fazer uma comparação entre os resultados obtidos pelo método proposto que é o algoritmo executado com 8 faixas (número de faixa sugestivo) e o SA básico que é o algoritmo executado com 1 faixa. Para cada instância foram realizadas 20 execuções com 8 faixas e 20 execuções também com uma faixa de modos independentes. Estes resultados estão mostrados na tabela 5.5 e 5.6 para instâncias da internet e para instâncias em grade, respectivamente.

		SA Básico		Método Proposto	
Núm. de Cidades	Ótimo Conhecido	% de Acerto	% Médio acima Ótima	% de Acerto	% Médio acima Ótima
30	48872	0	6,45	20	5,06
50	425	0	2,11	30	0,23
75	435	0	30,11	5	24,13
100	21282	5	0,58	40	0,11
150	26524	0	1,67	0	0,54
200	29368	0	1,80	0	0,81

Tabela 5.5 – Desempenho geral com instâncias da internet

		SA Básico		Método Proposto	
Núm. de Cidades	Ótimo Conhecido	% de Acerto	% Médio acima Ótima	% de Acerto	% Médio acima Ótima
64	64,0000	55	0,70	100	0,00
81	81,4142	60	0,40	100	0,00
100	100,0000	15	1,15	60	0,33
144	144,0000	0	1,70	15	0,66
196	196,0000	0	2,00	0	0,90
225	225,4142	0	1,87	0	0,73

Tabela 5.6 – Desempenho geral com instâncias em grade

Cabe ressaltar que todos os testes, inclusive para as instâncias maiores, foram executados dentro do mesmo conjunto de parâmetros, ou seja, para as instâncias maiores não foi feita nenhuma alteração diferenciada, como por exemplo, em ao seu tempo de execução (maior tempo), senão os definidos pelas suas próprias configurações. Os dados tabelados em 5.5 e 5.6 foram obtidos pela aplicação das fórmulas 5.1 e 5.2 correspondentes a percentual de acertos e percentual médio acima do ótimo (qualidade da solução), respectivamente.

5.2.4 Comentários finais

Para finalizar, alguns comentários adicionais merecem destaque:

1) Todo método que tem como paradigma o *Simulated Annealing* deve preocupar-se com o excessivo tempo de computação que o mesmo tomará. Portanto, parâmetros adequados no programa de resfriamento contribuem bastante para diminuir esse tempo. Várias propostas de parâmetros são encontradas na literatura pertinente que embora seja um bom indicativo, não é imperativo segui-los.

Neste trabalho, para os teste iniciais alguns desses parâmetros foram seguidos como orientação inicial, para depois se adequarem melhor à situação, outros foram se ajustando por meio de tentativa, conforme iam produzindo soluções melhores. Dentre estes, os valores dos parâmetros que correspondem ás temperaturas inicial e final são bastante crítico dentro da abordagem, pois tem relação direta com a convergência do método e portanto influenciando no tempo computacional gasto.

O método proposto teve sua temperatura inicial atribuída a um valor idêntico ao número de cidades para as instâncias da internet. Não combinando portanto, com uma das sugestões da literatura de que o valor da temperatura inicial deve ser tal que a taxa de aceitação (razão entre o número de transições aceitas pelo número de transições propostas) para aquela temperatura deva ser superior a 80% (KIRKPATRICK et al., 1983). O que neste caso, elevaria a temperatura inicial para o problema com 100 cidades (instância da internet), por exemplo, para 10.000. Isto levaria mais tempo do que se a mesma iniciasse com 100, como foi testado, já que esse valor tende para valores próximo de zero e que o fator de redução seria o mesmo.

Tal fato demonstra o ganho significativo que se obtém, mesmo considerando que a aplicação do método com 8 faixas de temperaturas (este é o número de faixas que apresentou melhor performance) consomem mais tempo de computação só que com um percentual de solução ótima bastante superior ao com uma faixa, neste caso, considerando a taxa de aceitação sugerida pela literatura. Embora não se tenha documentado o tempo, vale a pena observar as informações abaixo.

Com 1 faixa – Tempo de computação 250 segundos $\rightarrow$ 4% soluções ótimas.

Com 8 faixas –Tempo de computação 650 segundos $\rightarrow$ 40% soluções ótimas.

2) Apesar de não ser conclusivo, um fato que durante os testes chamou bastante a atenção e que pode se comprovado com os resultados já apresentados, é que à medida que o número de faixas aumenta, aumenta também o percentual de soluções ótimas. Entretanto, o que se pode observar é que essa proporcionalidade não é indefinida, ou seja, quando maior o número de faixas, maior os acertos em soluções ótimas, o que certamente comprometeria

a eficácia do método. Apesar de não estar tabelado, pelos gráficos acima, pode-se verificar uma tendência de que 8 é um número de faixas bastante recomendado.

Em experimentos, com a mesma instância e o mesmo número de execuções, se constatou que o percentual de soluções ótimas alcançadas pelo método com 10 e 12 faixas apresentou diferença desprezível (maior para 12) que pode ser entendido como dentro da margem probabilística, como o método.

3) Embora os dois grupos de problemas apresentem características idênticas com relação às restrições, o mesmo não se pode dizer sobre o intervalo de temperatura onde ocorre a maioria das melhorias na rota, que apresenta variação bastante acentuada. Apesar dos testes terem sido feitos pelo mesmo algoritmo (com múltiplas faixas), apenas com pequenas mudanças nos parâmetros do programa de resfriamento um comportamento muito diferente foi observado nos testes.

Para os problemas do primeiro grupo, os provenientes da internet, observou-se que o intervalo da temperatura onde ocorrem mais melhoramentos, em qualquer faixa, é na parte onde a temperatura está ainda alta, ou seja, nos 10% finais da temperatura as ocorrências de melhorias são quase nulas.

Nas instâncias dos problemas oriundas do segundo grupo a maioria das mudanças com melhoramento na rota, também em qualquer faixa, ocorrem mais acentuadamente na parte final do intervalo, ou seja, onde a temperatura é mais baixa.

CAPÍTULO 6

Conclusões e sugestões para futuras pesquisas

Neste capítulo, destacam-se, a importância do método aqui proposto e a contribuição que o mesmo oferece de como conseguir melhorar a performance de um algoritmo que tem o SA como base. Também são recomendadas algumas sugestões que podem ser exploradas em futuras pesquisas.

6.1 Conclusões

Este trabalho propõe e investigada a potencialidade de um método baseado na abordagem *Simulated Annealing* (SA) na solução do clássico problema do Caixeiro Viajante em instâncias de tamanho moderado, tendo como principal mudança em relação ao SA básico, múltiplas faixas de temperatura.

Devido sua vasta área de aplicabilidade, o PCV tem despertado grande interesse dos pesquisadores em apresentar novos métodos que resulte em boas soluções, principalmente para problemas reais. Este fato, especialmente, em se tratando de problema de otimização combinatorial, se traduz em uma tarefa árdua devido a sua grande complexidade, já que trata de um dos problemas mais difíceis de otimização combinatorial conhecidos.

Atualmente observa-se uma forte tendência em se utilizar métodos aproximados na resolução desta classe de problemas. Uma abordagem dessa natureza, apesar de não garantir encontrar uma ótima solução para o problema, é capaz de oferecer uma solução de

boa qualidade, em tempo de processamento aceitável, o que é perfeitamente considerado em muitas situações reais.

Para demonstrar a competitividade do método proposta, estudos de comparação foram realizados com diferentes números de faixas. Estes estudos foram baseados em dois parâmetros: qualidade da solução e performance do método. Os testes foram realizados com várias instâncias de origens diferentes (internet e grade), todas com soluções ótimas conhecidas.

Os experimentos mostraram que o método proposto representa uma boa alternativa relativa a outras técnicas heurísticas empregadas na solução do problema em questão que utilizam SA básico. Os resultados obtidos demonstram que o método apresenta performance variada e que vai melhorando à medida que o número de faixa aumenta.

Os resultados apontaram que o método proposto supera bastante o método com implementação básica do SA, principalmente em relação à qualidade da solução que na maioria das instâncias testadas alcança a solução ótima com bastante freqüência.

O trabalho mostra que a taxa média de acerto do método proposto é bastante superior a taxa média de acerto do SA básico. Sem considerar as instâncias com número de cidades acima de 150, onde a taxa de acerto é zero, a taxa média para o caso das instâncias da internet é 1,2 % para o SA básico e de 23,75 % para o método proposto. Para o caso das instâncias em grade, na mesma situação, o SA básico atinge 32,5 % enquanto que o método proposto alcança 68,75 %.

Estes resultados, embora ainda não conclusivos, produzem uma projeção muito otimista em relação ao potencial do método proposto, já que estes são os primeiros testes realizados e ainda sem nenhum estudo de controle de parâmetros, estudos estes, que podem produzir uma melhora ainda maior.

Finalmente, os resultados alcançados pelo método com múltiplas faixas aplicado em diversas instâncias do PCV, mostram a viabilidade da sua utilização como um meio de se obter soluções melhores ou com maior taxa de acerto do que se for considerado o caso de faixa único de temperatura, que é o SA básico. Vale salientar que a única referencia sobre múltiplas faixas é encontrada em MAZZUCCO JUNIOR (1999), só que para um método híbrido e aplicado aos problemas de programação da produção. Não existindo, portanto, ao

menos na literatura consultada, qualquer referência teórica ou prática sobre o método aqui proposto, sendo portanto considerado um novo método.

6.2 Sugestões para novas pesquisas

Trabalhos futuros poderiam ser focalizados na redução do tempo que o algoritmo leva para encontrar uma solução, investigando novas estruturas de vizinhanças de modo a tornar o método aqui proposto mais prático e eficiente. Um estudo mais aprofundado poderá resultar em novos métodos mais apropriados para essas abordagens (otimização combinatorial), desde que se consiga combinar novos procedimentos de pesquisa local aos algoritmos evolutivos. Uma melhor combinação dos parâmetros para o SA pode produzir um programa de resfriamento mais adequado a cada tipo de instância. Utilizando o SA como base e combinar com outra heurística tirando-se proveito dos bons de ambos os lados. Considerando essas argumentações pode-se propor investigações nas seguintes direções:

a) Novas estruturas de vizinhanças

Observou-se que durante os testes, alguns valores da solução eram alcançados já na primeira faixa, e que assim permaneciam até a última faixa sem sobre alteração, mesmo com as perturbações introduzidas nos aumentos bruscos de temperatura. Acredita-se que neste caso a introdução de uma nova estrutura de vizinhança de modo conveniente poderia resolver o problema.

b) Parâmetros adaptativos

O método estudado produz soluções superiores ao método alternativo apresentado, como mostrado anteriormente, porém, com a qualidade de solução e o tempo de computação relativamente diferente para instâncias com o mesmo número de cidades. Isto se deu com instâncias de origens diferentes ou quando as coordenadas das cidades apresentaram diferenças acentuadas em seus valores. Essa instabilidade mostra que os parâmetros aqui testados podem não representar uma combinação ótima ou que tal

sensibilidade possa ser minimizada com parâmetros adaptativos às origens das instâncias ou as coordenadas das cidades.

c) Hibridização

O desenvolvimento de novas heurísticas, fundamentadas, por exemplo, no *Simulated Annealing*, *tabu search* e Genético tem levado às pesquisas na direção das combinações (hibridização), ou seja, combinar de forma conveniente duas heurísticas, como por exemplo, o SA com operadores genéticos, como propõe MAZZUCCO JUNIOR (1999). A hibridização do SA com estratégias de pesquisa local mais complexas podem ser também uma promissora área de investigação e que pode ser facilmente implementado a partir do método proposto neste trabalho.

REFERENCIAS BIBLIOGRÁFICAS

AARTS, E. AND KORST, J., *Simulated annealing and Boltzmann Machines*, John Wiley & Sons, 1990.

AARTS, E.H.L. AND van LAARHOVEN, P. J. M., *A new polynomial-time cooling schedule*, In : Proccedings IEEE International Conference on CAD (ICCAD 85), 1985, p. 206-208.

APPLEGATE, D., BIXBY, R., CHVÀTAL, V. AND COOK, W., Microelectronics Technology Alert, July 1998, página HTML (http://www.crpc.rice.edu/CRPC/news Archive/mta_7_24_98.html.

BALL, M. AND MAGAZINE, M., *The design and analysis of heuristics*, Networks, vol. 11,1981, p. 516-525.

BARROS NETO, J. F., *Análise de desempenho dos operadores genéticos aplicados ao problema do caixeiro viajante*, página HTML (http://po1.pep.ufrj.br/~jfbarros/ agcap#.html).

BLAND, R. G. AND SHALLCROSS, D. F., *Large traveling salesman problems arising from experiments in X-ray crystallography: A preliminary report on computation*, Operations Research Letters, vol. 8, 1989, p. 125-128.

CARPANETO, G. AND TOTH, P., *Some new branching and bounding criteria for the asymmetric traveling salesman problems*, Management Science, vol. 26, 1980, p. 736-743.

CERNY, V., *Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm*, Journal of Optimisation Theory and Applications, n. 45, 1985, p. 41-51.

CHRISTOFIDES, N. AND EILON, S., *Algorithms for large-scale traveling salesman problems*, Operational Research Quarterly, vol. 23, 1972, p. 511-518.

COOK, S., *The complexity of Theorem Proving Procedures,* Proc. 3rd ACM Symp. On the Theory of Computing, 1971, p. 151-158.

CROES, G. A., *A method for solving traveling salesman problems,* Operations Research, vol. 6, 1958, p. 791-812.

CROWDER, H. AND PADBERG, M., *Solving large-scale symmetric travelling salesman problems to optimality,* Mgmt Sci, vol. 26, 1980, p. 495-509.

DANTZIG, G., FULKERSON, R. AND JOHNSON, S., *Solution of a large-scale traveling-salesman problem,* Operations Research, vol. 36, 1954, p. 393-410.

FLOOD, M. M., *The traveling-salesman problem,* Operations Research, vol. 4, 1956, p. 61-75.

FREDMAN, M. L., JOHNSON, D. S., MCGEOCH, L. A., OSTHEIMER, G., *Data structures for traveling salesman,* in proceedings of the 4th annual ACM-SIAM symposium on discrete algorithms, 1993, p. 145-154.

GAREY, M. R. AND JOHSON, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness,* Ed. W. H. Freeman, San Francisco, 1979.

GOLDEN, B. L. AND SKISCIM, C.C., *Using Simulated Annealing to Solve Routing and Location Problems,* Naval Research Logistics Quarterly, vol. 33, 1986, p. 266-279.

GU, J. AND HUANG, X., *Efficient local search with search space smoothing: A case study of the traveling salesman problem (TSP),* IEEE Trans. Systems, Man, and Cybernetics, vol. 24, 1994, p. 728-735.

GUTIN, G. AND YEO, A., *Small diameter neighbourhood graphs for the traveling salesman problem.* Technical Report DMS of Brunel University, 1998.

HAJEK, B., *Cooling Schedules for Optimal Annealing,* Mathematics of Operations Research, vol. 13, 1988, p. 311-329.

HELP, M. AND KARP, R. M., *The traveling-salesman problem and minimum spanning trees,* Operations Research, vol. 18,1970, p. 1138-1162.

HELSGAUN, K., *An effective implementation of the Lin-Kernighan traveling salesman heuristic,* European Journal of Operational Research, vol. 126, 2000, p. 106-130.

JOHNSON, D. S. AND MCGEOCH, L. A., *The traveling salesman problem: A case study in local optimization,* in (E.H.L. Aarts and J. K. Lenstra,eds), *Local Search in Combinatorial Optimization,* Wiley & Sons, New York, 1997.

JOHNSON, D. S., ARAGON, C. R., MCGEOCH, L. A. AND SCHEVON, C., *Optimization by Simulated Annealing: An Experimental Evaluation, Part I (Graph Partitioning),* Operations Research, vol. 37, 1989, p. 865-892.

JOHNSON, D. S., MCGEOCH, L. A. AND ROTHENBERG, E. E., *Asymptotic experimental analysis for the Held-Karp traveling salesman bound,* in proceedings of the 7th annual ACM-SIAM symposium on discrete algorithms, 1996, p. 341-350.

KIRKPATRICK, S., GELATT, Jr. C. D. AND VECCHI, M. P., *Optimization by Simulated Annealing,* Science, n. 220, 1983, p. 671-680.

KORTE, B., *Applications of combinatorial optimization*, talk at the 13th International Mathematical Programming Symposium, Tokyo, 1988.

LAPORTE, G., *The traveling salesman problem: An overview of exact and approximate algorithms,* European Journal of Operational Research, vol. 59, 1992, p. 231-247.

LAWLER, E. L., LENSTRA, J. K., RINNOOY KAN, A. H. G. AND SHMOYS, D. B., *The Traveling Salesman Problem: a guided tour of combinatorial optimization*, Ed. John Wiley & Sons, Chichester, 1985.

LEWIS, H. R. AND PAPADIMITRIOU, C. H., *Elementos de Teoria da Computação,* : trad. Edson Furmankiewicz, 2ed, Bookman, Porto Alegre-RS, 2000.

LIN, S. AND KERNIGHAN, B. W., *An Effective Heuristic Algorithm for the Traveling Salesman Problem*, Operations Research, vol. 21, 1973, p. 498-516.

LIN, S., *Computer solutions of the traveling salesman problem,* Bell System Technical Journal, n. 44, 1965, p. 2245-2269.

LOURENÇO, H. R., *Job Shop Scheduling: Computational Study of Local Search and Large-Step Optimization Methods,* European Journal of Operational Research, vol. 83, 1995, p. 347-364.

LUNDY, M. AND MEES, A., *Convergence of an annealing algorithm,* Mathematical Programming, vol. 34, 1986, p. 111-124.

MARTIN O. C. AND OTTO, S. W., *Combining simulated annealing with local search heuristics*, Annals of Operations Research, Vol. 60, Balzer Scientific Publishers, Amsterdam, 1995.

MAZZUCCO JUNIOR, J., *Uma Abordagem Híbrida do Problema da Programação da Produção através dos Algoritmos Simulated Annealing e Genético*, Tese de Doutorado PPGEP / UFSC, 1999.

METROPOLIS, W., ROSENBLUTH, A., ROSENBLUTH, M., TELLER, A. AND TELLER, E., *Equation of State Calculations by Fast Computing Machines*, Journal of Chemical Physics, vol. 21, 1953, p. 1087-1092.

MILLER, D. L. AND PEKNY, J. F., *Exact solution of large asymmetric traveling salesman problems*, Science, nr. 251, 1991, p. 754-761.

PAPADIMITRIOU, C. H. AND STEIGLITZ, K., *Combinatorial optimization: algorithms and complexity*, Englewood Cliffs, N. J.: Prentice-Hall,1982.

PEARL, Judea, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, New York, The Addison-Wesley, 1984.

PUNNEN, A. P., *The traveling salesman problem: new polynomial approximation algorithms and domination analysis.* Manuscript, December, 1996.

REEVES, C. R., *An Improved Heuristic for the Quadratic Assignment Problem*, Journal of the Operation Research Society, vol. 36, 1985, p. 163-167.

REEVES, C. R., *Modern heuristic techniques for combinatorial problems*, Advanced topics in computer science, McGraw Hill, 1995.

REINELT, G., *A traveling salesman problem library*, ORSA journal of computing, v. 3, 1991, p. 376-384.

ROUTO, Terada, *Desenvolvimento de algoritmos e estruturas de dados*, São Paulo, McGraw-Hill, Makron, 1991.

TANOMARU, J., *Motivação, Fundamentos e Aplicações de Algoritmos Genéticos*, II Congresso Brasileiro de Redes Neurais,Curitiba-Pr, 1995.

TIAN, P., MA, J. AND ZHANG, D., *Application of the simulated annealing algorithm to the combinatorial optimisation problem with permutation property: An investigation of*

generation mechanism, European Journal of Operational Research, vol. 118, 1999, p. 81-94.

VOUDOURIS, C. AND TSANG, E., *Guided local search and its application to the traveling salesman problem*, European Journal of Operational Research, vol. 113, 1999, p. 469-499.