

BRUNO AVILA LEAL DE MEIRELLES HERRERA

**COMBINAÇÃO DE ENXAME DE PARTÍCULAS
COM INSPIRAÇÃO QUÂNTICA E MÉTODO LIN-
KERNIGHAN-HELSGAUN APLICADA AO
PROBLEMA DO CAIXEIRO VIAJANTE**

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Engenharia de Produção e Sistemas.

CURITIBA

2007

BRUNO AVILA LEAL DE MEIRELLES HERRERA

**COMBINAÇÃO DE ENXAME DE PARTÍCULAS
COM INSPIRAÇÃO QUÂNTICA E MÉTODO LIN-
KERNIGHAN-HELGAUN APLICADA AO
PROBLEMA DO CAIXEIRO VIAJANTE**

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Engenharia de Produção e Sistemas.

Área de Concentração: Automação e Controle de Processos.

Orientador: Prof. Dr. Leandro dos Santos Coelho

CURITIBA

2007

HERRERA, Bruno Avila Leal de Meirelles

COMBINAÇÃO DE ENXAME DE PARTÍCULAS COM INSPIRAÇÃO QUÂNTICA E MÉTODO LIN-KERNIGHAN-HELGAUN APLICADA AO PROBLEMA DO CAIXEIRO VIAJANTE

. Curitiba, 2007. 97p.

Dissertação – Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Engenharia de Produção e Sistemas.

1. Problema do Caixeiro Viajante 2. Otimização Combinatória 3. FPGA 4. Computação Quântica 5. Enxame de partículas 6. Mecânica Quântica. 7. Heurísticas. Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e de Tecnologia. Programa de Pós-Graduação em Engenharia de Produção e Sistemas.

“Algo que aprendi em uma longa vida: toda nossa ciência, medida contra a realidade, é primitiva e infantil - e ainda assim, é a coisa mais preciosa que temos.”

Albert Einstein (1879-1955)

Em memória da minha querida avó Neide Ferreira Herrera

À minha família

Agradecimentos

Gostaria de agradecer as seguintes pessoas por terem de, alguma forma, contribuído para esta dissertação de mestrado:

Ao meu orientador Prof. Dr. Leandro dos Santos Coelho, por ser uma fonte de inspiração, crítica e por ter me guiado durante a pesquisa.

À minha família, principalmente minha amada mãe Sylvia, meu querido irmão Gabriel por terem sido um ponto de apoio e uma fonte estímulo, meus avós Roberto e Lydia pelos princípios e ensinamentos que levarei para vida toda. Ao meu avô Ayrton, meu pai Henry, minhas tias e primos. Especialmente grato a minha madrinha Giselle e meu primo Rafael Ribeiro.

Ao colega de trabalho Célio Costa, pelo apoio, incentivo e compreensão durante os momentos de necessidade. A todos os outros que, de certa forma, direta ou indiretamente, contribuíram para este trabalho: Hércio Iwamoto, Osmar Brusamolin, Luciano Ribas e Márcio Machado.

Não menos grato a todos meus colegas de curso e amigos, dentro e fora da Universidade, pelo carinho, disposição e momentos de alegria. Especialmente grato aos amigos: Marcelo Yanaga, Leonardo Ribas, Carlos Juliano, Cássio Milani, Felipe Sória, Andrei Biscaro e Fábio Guerra.

À Pontifícia Universidade Católica do Paraná pela sua estrutura e seu apoio financeiro.

A todos, muito obrigado!

Resumo

O Problema do Caixeiro Viajante (PCV) é um dos mais bem conhecidos e estudados problemas da Teoria dos Grafos e da Complexidade. Neste contexto, pode-se interpretá-lo como o problema de determinar um ciclo ou circuito Hamiltoniano de menor valor de função objetivo, ou seja, menor distância Euclidiana para todo o circuito em um grafo com custos associados às arestas. Uma das principais questões envolvidas neste problema, e certamente uma das questões das áreas de Otimização e Computação, é saber se existe um algoritmo eficiente de tempo polinomial para computar tal ciclo, ou se um algoritmo como este não pode existir, caracterizando-o então como um problema intratável. Devido à dificuldade de solução exata dos problemas de Otimização Combinatória e em especial o PCV, vários métodos heurísticos têm sido desenvolvidos nas últimas décadas. A proposta deste trabalho é realizar uma análise estatística de algumas abordagens de otimização aplicadas para instâncias simétricas e assimétricas do PCV. A contribuição principal desta dissertação é avaliar o desempenho de algumas abordagens em relação à obtenção da solução ótima para determinadas instâncias do PCV. As abordagens avaliadas para a resolução do PCV foram divididas em quatro métodos de Otimização Combinatória: (i) algoritmo Lin-Kernighan-Helsgaun (LKH), (ii) LKH com iniciação baseada na geração de soluções iniciais com distribuição uniforme, (iii) meta-heurística de enxame (ou nuvem) de partículas com inspiração quântica, e (iv) uma proposta de combinar enxame de partículas com inspiração quântica e método LKH como procedimento de busca local. As abordagens para resolução do PCV foram testadas em um computador pessoal com ambiente Windows para instâncias de PCV simétrico e assimétrico, estas obtidas da biblioteca de problemas testes da TSPLIB (*Traveling Salesman Problem Library*). Outra contribuição desta dissertação é a validação dos algoritmos de Otimização Combinatória mencionados em uma placa RC200 da Celoxica que possui uma *Field Programmable Gate Array* Virtex II XC2V1000 da Xilinx. Neste contexto, os algoritmos testados apresentaram resultados promissores, pois foram capazes de produzir um resultado satisfatório, em termos de custo computacional e qualidade da solução para o PCV.

Palavras-Chave: Otimização Combinatória, Problema do Caixeiro Viajante, FPGA, Enxame de Partículas, Enxame de Partículas com Inspiração Quântica, Computação Quântica, Mecânica Quântica, Algoritmo Lin-Kernighan-Helsgaun, Heurísticas.

Abstract

The Traveling Salesman Problem (TSP) is one of the most studied and well known problems of Graph's and Complexity's theory. In this context, it can be defined as finding the a Hamiltonian cycle which cost function is minimum, in other words it can be defined as finding the lower Euclidian distance for the graph, where the costs are associated with graph's edges. One of the main questions involved in this Optimization and Computing problem is to find an efficient algorithm that can solve the problem in polynomial time or if there is not such algorithm, the problem can be defined as untreatable. Due the difficult to find an exact solution for Combinatorial Optimization problems, in special the TSP, many heuristic methods have been developed during past years. The proposal of this work is realize a statistic analysis based on optimization approaches applied to some symmetrical and asymmetrical TSP instances. The main contribution of this work is to test optimization approaches using optimum value of TSP as reference. The used approaches for solving the TSP were dived in four Combinatorial Optimization methods: (i) Lin-Kernighan-Helsgaun (LKH) algorithm, (ii) LKH with initial tour based on uniform distribution, (iii) meta-heuristic based on particle swarm optimization with quantum inspired behavior, and (iv) an hybrid proposal combining particle swarm optimization with quantum inspired behavior and LKH for local search procedure. These approaches for TSP resolution, symmetrical and asymmetrical instances, were tested in a personal computer running Windows XP. The instances were taken from TSPLIB (Traveling Salesman Problem Library). As another contribution of this work the Combination Optimization algorithms mentioned were tested against an embedded system, a RC200 Celoxica's board composed by a FPGA (Field Programmable Gate Array) Xilinx's Virtex II XC2V1000. The tested algorithms presented promising results in terms of computational cost and solution quality for the TSP.

Keywords: Combinatorial Optimization, Traveling Salesman Problem, FPGA, Particle Swarm, Quantum Inspired Particle Swarm, Quantum Computing, Quantum Mechanics, Lin-Kernighan-Helsgaun, Heuristics

Sumário

AGRADECIMENTOS	V
RESUMO	VI
ABSTRACT	VII
LISTA DE FIGURAS	X
LISTA DE TABELAS	XI
LISTA DE ALGORITMOS	XII
LISTA DE ABREVIATURAS	XIII
1. INTRODUÇÃO	1
1.1. MOTIVAÇÃO E JUSTIFICATIVA	3
1.2. OBJETIVO E METODOLOGIA.....	5
1.3. CONTRIBUIÇÃO.....	7
1.4. ESTRUTURA DO TRABALHO	7
2. CONSIDERAÇÕES SOBRE O PROBLEMA DO CAIXEIRO VIAJANTE E UMA REVISÃO DA LITERATURA	9
2.1. PROBLEMA DO CAIXEIRO VIAJANTE: FUNDAMENTOS E BREVE HISTÓRICO.....	10
2.1.1. <i>Modelagem Matemática do PCV</i>	12
2.1.2. <i>Teoria da Complexidade</i>	13
2.1.3. <i>Métodos de Resolução do PCV</i>	19
2.1.3.1. <i>Métodos Exatos</i>	20
2.1.3.2. <i>Métodos Heurísticos</i>	21
2.2. <i>FPGA – FIELD PROGRAMMABLE GATE ARRAY</i>	23
2.3. MECÂNICA QUÂNTICA	25
2.3.1. <i>Efeito Fotoelétrico</i>	25

2.3.2.	<i>Princípio da Incerteza de Heisenberg</i>	26
2.3.3.	<i>Função de Onda</i>	28
2.3.4.	<i>Equação de Schrödinger</i>	29
2.4.	COMPUTAÇÃO QUÂNTICA	29
3.	METODOLOGIA USANDO HEURÍSTICAS E METAHEURÍSTICAS	31
3.1.	HEURÍSTICAS OU ALGORITMOS APROXIMADOS	32
3.1.1.	<i>Heurística de Melhoria k-opt</i>	32
3.1.2.	<i>Heurística de Melhoria Lin-Kernighan</i>	34
3.1.3.	<i>Lin-Kernighan Helsgaun</i>	37
3.1.4.	<i>Procedimentos de Busca Local</i>	39
3.2.	METAHEURÍSTICAS	39
3.3.	ALGORITMO DE ENXAME DE PARTÍCULAS (ALGORITMO PSO)	44
3.3.1.	<i>QPSO (Quantum Particle Swarm Optimization)</i>	46
3.3.1.1.	<i>Vantagens do QPSO</i>	49
3.3.1.2.	<i>Discretização</i>	50
4.	PLATAFORMA DE DESENVOLVIMENTO	53
4.1.	<i>HARDWARE</i>	54
4.1.1.	<i>Placa Celoxica RC200</i>	54
4.1.2.	<i>MicroBlaze</i>	55
4.2.	<i>SOFTWARE</i>	58
5.	IMPLEMENTAÇÃO COMPUTACIONAL E ANÁLISE DE RESULTADOS	60
5.1.	FORMA DE EXECUÇÃO	60
5.2.	RESULTADOS PARA O PCV SIMÉTRICO	61
5.3.	PROBLEMAS ASSIMÉTRICOS	69
5.4.	EXECUÇÕES EMBARCADAS NA FPGA	72
	CONCLUSÕES E TRABALHOS FUTUROS	74
	REFERÊNCIAS	76

Lista de Figuras

Figura 2.1. Exemplo de uma rota para o PCV.....	13
Figura 2.2. Estrutura básica de um <i>FPGA</i>	23
Figura 3.1. Troca <i>2-opt</i>	33
Figura 3.2. Troca <i>3-opt</i>	34
Figura 3.3. Espaço de Busca PSO e QPSO.	47
Figura 4.1. Placa Celoxica RC200.....	54
Figura 4.2. Placa Celoxica RC200 (Blocos Lógicos).	55
Figura 4.3. Visão geral processador <i>soft core</i> MicroBlaze.	56
Figura 4.4. Barramento de periféricos <i>Microblaze</i>	57
Figura 4.5. Endereçamento de periféricos <i>Microblaze</i>	58
Figura 4.6. <i>Boot</i> ucLinux compilado para placa RC200.	59
Figura 5.1. Instância swiss42 (Mínimo).....	64
Figura 5.2. Instância swiss42 (Média).	64
Figura 5.3. Instância swiss42 (Mínimo e Média).	65
Figura 5.4. Instância pcb1173 (Mínimo).....	65
Figura 5.5. Instância pcb1173 (Média).	66
Figura 5.6. Instância pcb1173 (Mínimo e Média).	66
Figura 5.7. Instância fl3795 (Mínimo).....	67
Figura 5.8. Instância fl3795 (Média).....	67
Figura 5.9. Instância fl3795 (Mínimo e Média).....	68
Figura 5.10. Exemplo da melhor rota encontrada para o problema pcb1173.....	68
Figura 5.11. Instância rbg443 (Mínimo).	71
Figura 5.12. Instância rbg443 (Média).....	71
Figura 5.13. Instância rbg443 (Mínimo e Média).....	72
Figura 5.14. Mínimo obtido para instância swiss42 em <i>FPGA</i>	73
Figura 5.15. Mínimo obtido para instância ftv130 em <i>FPGA</i>	73

Lista de Tabelas

Tabela 2-1. Explosão combinatória e o tempo estimado.	14
Tabela 5-1. Resultados de Otimização das Instâncias para o PCV Simétrico.....	63
Tabela 5-2. Resultados de Otimização das Instâncias para o PCV Assimétrico.	70
Tabela 5-3. Resultados das Instâncias em Execuções Embarcadas.	72

Lista de Algoritmos

Algoritmo 3.1 Pseudocódigo QPSO.	49
Algoritmo 3.2 Discretização da Matriz de População.	52

Lista de Abreviaturas

Abreviatura	Descrição
ATA	<i>Advanced Technology Attachment</i>
BRAM	<i>Block Random Access Memory</i>
CPLD	<i>Complex Programmable Logic Device</i>
CPU	<i>Central Processing Unit</i>
DNA	<i>DeoxyriboNucleic Acid</i>
FPGA	<i>Field Programmable Gate Array</i>
FSL	<i>Fast Simplex Link</i>
GNU	Acrônimo recursivo para <i>GNU's Not Unix</i>
GPIO	<i>General Purpose Input/Output</i>
GRASP	<i>Greedy Randomized Adaptive Search Procedure</i>
IO	<i>Input/Output</i>
IP	<i>Intellectual Property</i>
JTAG	<i>Joint Test Action Group</i>
LED	<i>Light Emitting Diode</i>
LIP	<i>Learning Inclination Point</i>
LK	Lin-Kernighan
LKH	Lin-Kernighan-Helsegaun
LMB	<i>Memory Bus</i>
LUT	<i>Look-Up Table</i>
MMC	Método de Monte Carlo
NP	<i>Non-Polynomial</i>
OPB	<i>On-Board Processing</i>
PCV	Problema do Caixeiro Viajante
PROM	<i>Programmable Read-Only Memory</i>

PSO	<i>Particle Swarm Optimization</i>
QPSO	<i>Quantum Particle Swarm Optimization</i>
RISC	<i>Reduced Instruction Set Computer</i>
RTOS	<i>Real Time Operating System</i>
SRAM	<i>Static Random Access Memory</i>
TC	Teoria da Complexidade
TSPLIB	<i>Traveling Salesman Problem Library</i>

Capítulo 1.

1 Introdução

A Otimização Combinatória é um ramo da computação que estuda os problemas de otimização em conjuntos [GOL00]. Os problemas de Otimização Combinatória, devido à grande dificuldade de solução e sua presença em várias situações do cotidiano, têm atraído cada vez mais a atenção de pesquisadores de diversas áreas que têm realizado esforços para desenvolver algoritmos cada vez mais eficientes para serem aplicados a tais problemas [PRE06].

Em um problema de otimização tem-se uma ou mais funções objetivo e um conjunto de restrições (opcional), ambos relacionados às variáveis de decisão. Em problemas de Otimização Combinatória, o objetivo é atribuir valores a um conjunto de variáveis de decisão, de tal modo que uma determinada função objetivo seja otimizada (minimizada ou maximizada, dependendo do objetivo) atendendo um determinado conjunto de restrições.

Matematicamente, pode-se exprimir um problema de Otimização Combinatória através de um conjunto finito representado por $N = \{1, \dots, n_i\}$, com ponderações c_j associadas a cada $j \in N$ e um conjunto F formado por subconjuntos viáveis de N , onde n_i é o número de instâncias do problema em questão. Desta forma, deseja-se determinar os elementos de F , tais que o somatório das ponderações associadas seja mínimo, ou seja, determinar:

$$\min_{S \subseteq N} \left\{ \sum_{j \in S} c_j : S \in F \right\} \quad (1.1)$$

Em problemas deste tipo, uma solução ótima trivial para se resolver este tipo de problema seria verificar todas as possíveis soluções e, a partir dos resultados, verificar qual

seria a melhor solução. Deve-se observar que a determinação do melhor valor de uma função pode ser inviável para os meios computacionais atuais, dependendo da complexidade do modelo matemático adotado para representar o problema e número de variáveis a serem otimizadas no problema abordado.

O Problema do Caixeiro Viajante (PCV, *Traveling Salesman Problem*), o Problema da Mochila (*Knapsack Problem*), da Cobertura Mínima por Conjuntos, da Árvore Geradora Mínima, da Árvore de Steiner e do roteamento de veículos são exemplos de problemas de Otimização Combinatória [PRE06].

Como exemplo clássico de problema de Otimização Combinatória, o PCV surge naturalmente em aplicações práticas, tais como problemas de otimização de redes de telecomunicação, movimento de ferramentas de corte, perfurações de furos em placas de circuitos impressos, alocação de trabalhadores ou máquinas a tarefas, cristalografia, controle de robôs móveis, seqüenciamento cronológico de eventos, biologia molecular com o alinhamento de cadeias de DNA (*DeoxyriboNucleic Acid*), entre outros.

O PCV é definido por um conjunto de cidades e uma função objetivo envolvendo os custos de viagem entre cada par de cidades. O objetivo do PCV é encontrar uma rota (*tour* ou caminho) pelo qual o caixeiro passe por todos os nós do grafo conectado (instâncias) com menor valor de função objetivo (problema de minimização), isto é, o menor somatório da distância Euclidiana entre os pontos (coordenadas no plano cartesiano) de um grafo. Outra interpretação para o objetivo do PCV é em obter rotas que formem um ciclo (ou circuito) Hamiltoniano de custo total mínimo, ou seja, um ciclo que passe por cada um dos nós do grafo somente uma vez.

Uma forma de resolver tais problemas seria simplesmente enumerar todas as soluções possíveis e guardar aquela de menor custo. Entretanto, essa é uma idéia ingênua, pois para a maioria dos problemas, esse método torna-se impraticável, já que existe um elevado número de soluções possíveis. Mesmo que se utilize um supercomputador para resolver o problema, o tempo de processamento pode ser de várias horas, vários dias ou até anos. Portanto, técnicas computacionais mais apuradas são necessárias para resolver esses problemas [PRES06].

Os Problemas de Otimização Combinatória têm sido utilizados como modelos para diversas situações reais [LOU01] [JOH97], tanto em suas versões com um único objetivo a ser otimizado como em suas versões com múltiplos critérios [ISH98], [JAS02]. Grande parte desses problemas pertence à classe dos problemas NP-árduos (ou NP-difícil), o que significa que dificilmente, algum dia, os algoritmos polinomiais que os solucionem serão apresentados,

ou seja, há fortes evidências de que não existe algoritmo polinomial para solucionar problemas dessa classe. Segundo Ramos [RAM05], essa constatação foi reforçada por Karp [KAR72], que apresentou, em seu trabalho, 24 problemas NP-completos, tendo sido amplamente difundida por Garey e Johnson, em 1979 [GAR79].

Neste contexto, os problemas de Otimização Combinatória têm envolvido muitos pesquisadores na busca por soluções aproximativas de boa qualidade para aqueles, desde a aceitação de que eles são considerados insolúveis em tempo polinomial.

1.1. Motivação e Justificativa

O trabalho desenvolvido nesta dissertação foi motivado por dois fatores. Em primeiro lugar pretendeu-se tratar problemas combinatórios do tipo PCV, por serem problemas de difícil solução. Por outro lado, e em segundo lugar, vislumbra-se adquirir competências com a tecnologia de FPGA, projeto de metaheurísticas bio-inspiradas (por exemplo, o algoritmo PSO) e interesse em aprender conceitos de Mecânica e Computação Quântica.

Em termos da justificativa, este trabalho pretende apresentar uma nova abordagem híbrida de LKH e algoritmo PSO com inspiração quântica na resolução de algumas classes de problemas do PCV. No entanto, deve-se ressaltar que a metodologia aqui desenvolvida não pretende ser, de forma alguma, uma solução definitiva, atendendo à complexidade do problema tratado.

Lembrando que o foco deste trabalho é o uso de heurística combinada a uma metaheurística aplicadas em PCVs, a dificuldade de solução do PCV está no número elevado de soluções existentes. Assumindo que a distância de uma cidade i a outra j seja simétrica, isto é, que $d_{ij} = d_{ji}$, o número total de soluções possíveis é $(n - 1)! / 2$, onde n é o número de cidades, sendo classificado na literatura como NP-árduos, como apresentado em [MEL01], isto é, não existem algoritmos que os resolvam em tempo polinomial.

Desde então, a comunidade científica tem realizado esforços na busca de soluções aproximativas para problemas dessa classe. Campello e Maculan [CAM94] apresentam uma revisão a respeito da teoria da NP-completude e se referem aos problemas NP-árduos como pelo menos tão difíceis quanto qualquer problema classificado como NP-completo. Apresentam, ainda, definições para problemas de Otimização Combinatória, especialmente

para o PCV, enfatizando que se justifica o uso de heurísticas – algoritmos especializados na busca de soluções aproximadas de boa qualidade – inclusive “para a obtenção de boas soluções viáveis iniciais para diversos algoritmos exatos” [RAM05].

Segundo Prestes [PRE06], ao longo dos anos, diversas técnicas foram desenvolvidas para a construção de algoritmos exatos que resolvessem tais problemas, tais como: programação dinâmica, *branch-and-bound* e métodos de planos de corte [PAP82], [NEM88], [HOF00]. Entretanto, como os tempos para solucionar instâncias de grande porte por algoritmos exatos são inviáveis, a opção, nesses casos, seria o uso de algoritmos heurísticos.

Define-se heurística como sendo uma técnica que procura boas soluções (próximas da otimalidade) a um custo computacional razoável, sem, no entanto, estar capacitada a garantir a otimalidade, bem como garantir quão próxima uma determinada solução está da solução ótima [CHA03].

Neste contexto, quando não se necessita determinar uma solução ótima para um estudo de caso de PCV, muitas vezes, por causa do tempo necessário para a mesma, utilizam-se então os métodos heurísticos, que conseguem dar uma resposta satisfatória em muitos PCVs. Os métodos aproximativos podem se enquadrar nesta categoria, acrescentando-se que, para estes casos, são conhecidas propriedades com garantia do pior caso. Além disso, segundo Osman e Laporte [OSM96] é comum, na literatura de Otimização Combinatória, os algoritmos aproximativos serem tratados como algoritmos heurísticos. As heurísticas para o PCV abrangem os seguintes tipos: métodos construtivos, métodos de enumeração limitada e métodos de melhoria. Deve-se destacar, no entanto, que os algoritmos aproximativos possuem propriedades matemáticas para garantir o pior caso, mas que dado à dificuldade do PCV são aplicados apenas a instâncias muito particulares.

Em síntese, as heurísticas são algoritmos que não têm garantia de encontrar a solução ótima, ou seja, a melhor solução existente para o problema. No entanto, existem diversas abordagens para a construção de algoritmos heurísticos. Inicialmente, os pesquisadores utilizavam técnicas ditas gulosas ou míopes no desenvolvimento de tais algoritmos [PRE06], tais como método do vizinho mais próximo, métodos de inserção, métodos de melhoria de roteiros do tipo *k-opt* [LAP92] e algoritmo Lin-Kernighan [LIN73]. Neste contexto, deve-se enfatizar que o método de Lin-Kernighan é um método relativamente antigo [LIN73], mas que ainda é muito difundido entre os estudos relacionados à obtenção de promissoras soluções do PCV, uma vez que é um dos métodos mais poderosos e robustos que existem para tal aplicação. A idéia geral de LK, por sua vez, é concretizar substituições contínuas entre as

rotas do problema para que se possam melhorar os roteiros já existentes. Este tipo de metodologia é um dos métodos mais rápidos e se consegue chegar a bons resultados com sua utilização. A literatura recente de PCV tem apresentado algumas modificações importantes para o método de Lin-Kernighan, tais como as variantes propostas em [AAR97], [NET99], [HEL00], [WAL01].

Uma desvantagem das heurísticas reside na dificuldade de fugir de ótimos locais, o que deu origem à outra classe de metodologias, denominada metaheurística, que possui ferramentas que possibilitam sair destes ótimos locais, permitindo a busca em regiões mais promissoras. O desafio é produzir, em tempo mínimo, soluções tão próximas quanto possíveis da solução ótima (quando conhecida a priori, principalmente para problemas teste da literatura).

1.2. Objetivo e Metodologia

Quando se quer responder um problema através de uma pesquisa científica deve-se adotar uma seqüência de etapas para fornecer respostas ao problema. Esta seqüência de etapas, metodologia adotada, guia o trabalho para se obter os resultados esperados cientificamente. O projeto de pesquisa é a seqüência lógica que conecta os dados empíricos às questões de pesquisa iniciais e, em última análise, às suas conclusões [YIN02].

A pesquisa experimental é considerada o melhor exemplo de pesquisa científica, pois há um alto nível de controle da situação, podem-se isolar todas as estruturas de qualquer interferência do meio exterior, gerando maior confiabilidade em seus resultados. Mesmo assim ela é flexível, podendo dar inúmeras respostas diferentes a problemas diferentes com um único experimento. A pesquisa experimental constitui um delineamento prestigiado nos meios científicos, e consiste, essencialmente, em determinar o objeto de estudo, selecionar as variáveis que seriam capazes de influenciá-lo, definir as formas de controle e de observação dos efeitos que a variável produz no objeto. Resumindo, a pesquisa experimental trata-se de uma pesquisa onde o pesquisador é um agente ativo e não apenas um observador passivo [GIL02].

No contexto deste trabalho, adotou-se como objetivo principal a utilização de uma heurística de Lin-Kernighan-Helsgaun [HEL00] combinada a uma metaheurística de enxame (ou nuvem) de partículas para validação experimental em PVC, tanto simétrico quanto

assimétrico, presentes na TSPLIB [REI91]. Lembrando que no PVC simétrico, o custo não depende da direção da viagem entre duas cidades.

Deve enfatizar que em termos de metaheurísticas, uma abordagem promissora é a de enxame de partículas (PSO, *Particle Swarm Optimization*). O algoritmo PSO foi desenvolvido inicialmente por Kennedy e Eberhart [KEN95] baseada nos estudos do sócio-biologista Edward Osborne Wilson [WIL71]. O fundamento de desenvolvimento do algoritmo PSO é uma hipótese na qual a troca de informações entre seres de uma mesma espécie oferece uma vantagem evolucionária. O algoritmo PSO constitui uma técnica evolutiva (alguns autores classificam como uma abordagem de inteligência coletiva ou de enxames) baseada em uma população de soluções e transições aleatórias. O algoritmo PSO apresenta características similares a técnicas da computação evolutiva, que são baseadas em uma população de soluções. Entretanto, no algoritmo PSO é motivada pela simulação de comportamento social e cooperação entre agentes em vez da sobrevivência do indivíduo mais apto. No algoritmo PSO, cada solução candidata (denominada partícula) possui associada uma velocidade. A velocidade é ajustada através de uma equação de atualização que considera a experiência da partícula correspondente e a experiência das outras partículas da população.

Outra abordagem recente é da utilização de conceitos da mecânica quântica [PAN05] e também da computação quântica [CHU00] no projeto de algoritmos de otimização [HOG00], [PRO02], [COE07]. A Computação Quântica foi proposta por Benioff [BEN80] e Feynman [FEY82] e é uma área de pesquisa com crescimento acentuado nos últimos anos, de investigação e aplicação da Mecânica Quântica. A Mecânica Quântica é uma estrutura matemática, ou conjunto de regras para a construção de teorias físicas. Mas a computação baseada em leis da física clássica leva a abordagens completamente distintas sobre o processamento da informação que aquela baseada na Mecânica Quântica. Neste sentido, os conceitos oriundos da Computação Quântica (por exemplo, bits quânticos ou *qu-bits*, portas quânticas e paralelismo de processamento) e Mecânica Quântica (superposição de estados, interferência, energia potencial delta e equação de Schrödinger) podem ser explorados para concepção de novos métodos ou mesmo melhorar a eficiência de métodos de otimização já existentes.

Uma abordagem de algoritmo PSO inspirada em conceitos da mecânica proposta por [SUN04a], [SUN04b], [SUN05], [LIU05] para problemas de otimização contínua é modificada neste trabalho com uma representação discreta para a resolução do PCV.

1.3. Contribuição

A contribuição principal desta dissertação é avaliar o desempenho de algumas abordagens de otimização em relação à obtenção da solução para determinadas instâncias do PCV. As abordagens avaliadas para a resolução do PCV foram divididas em quatro métodos de Otimização Combinatória: (i) algoritmo Lin-Kernighan-Helsgaun (LKH), (ii) LKH com iniciação baseada na geração de soluções iniciais (rotas iniciais) com distribuição uniforme, (iii) algoritmo PSO com inspiração quântica, e (iv) uma proposta de combinar o algoritmo PSO com inspiração quântica e método LKH como procedimento de busca local. As abordagens mencionadas para resolução do PCV foram testadas em um computador pessoal com ambiente Linux para instâncias de PCV simétrico e assimétrico, estas obtidas da biblioteca de problemas testes TSPLIB [REI91].

Outra contribuição desta dissertação é a validação dos algoritmos de Otimização Combinatória para alguns testes sugeridos pela TSPLIB em um *softcore processor*. Para que fosse possível as suas implementações utiliza-se o *ucLinux Kernel 2.4* para uma placa RC200 da Celoxica que possui uma *Field Programmable Gate Array (FPGA)* Virtex II XC2V1000 da Xilinx. Deve-se enfatizar que a utilização de sistemas embarcados de *hardware* reconfigurável do tipo FPGA está se tornando uma realidade em diversas áreas da Engenharia, incluindo-se para validação de metaheurísticas [ZEB02], [MER02], [SCH04]. Diferentemente de um computador de uso geral como, por exemplo, um computador pessoal que podem executar várias tarefas simultaneamente, os sistemas embarcados são desenvolvidos para realizar um número limitado de tarefas previamente definidas com objetivo de resolver um problema específico.

1.4. Estrutura do Trabalho

Este trabalho está dividido em seis capítulos: esta introdução, a revisão bibliográfica, a metodologia proposta, a plataforma de desenvolvimento, os resultados e discussões e as conclusões e perspectivas de estudos futuros.

O presente capítulo introduz o PCV e resalta a importância de resolvê-lo eficientemente por meio de heurísticas e metaheurísticas. O capítulo 2 apresenta uma breve revisão da literatura nas áreas de Otimização Combinatória (Problema do Caixeiro Viajante e Teoria de Grafos), Teoria da Complexidade, FPGA e Computação Quântica.

No capítulo seguinte, o capítulo 3, uma breve fundamentação relativa a heurísticas e metaheurísticas é apresentada. Além disso, a metodologia proposta para o PCV utilizando heurística LKH isolada e/ou combinada com algoritmo PSO com inspiração quântica é detalhada.

Em seguida, o capítulo 4 apresenta a plataforma de desenvolvimento abordando os principais aspectos de *hardware* e *software* utilizados, bem como a criação de *kernel* do ucLinux compatível com a placa RC200 da Celoxica.

No capítulo 5 são apresentados os resultados obtidos da aplicação dos métodos de otimização propostos para alguns problemas teste para o PCV (simétrico e assimétrico) da TSPLIB e com as discussões pertinentes.

As conclusões, baseadas nos resultados obtidos com a realização de simulações computacionais sintetiza alguns pontos de relevância levantados neste trabalho, suas principais contribuições. Finalizando, algumas sugestões para pesquisas futuras são sugeridas.

Capítulo 2.

2 Considerações sobre o Problema do Caixeiro Viajante e uma Revisão da Literatura

Muitos problemas práticos das mais diversas áreas, podem ser enquadrados como problemas de Otimização Combinatória e mais especificamente como PCVs. Estes problemas são classificados na literatura como NP-difíceis, isto é, não são conhecidos algoritmos que os resolvam em tempo polinomial (os detalhes serão abordadas no Capítulo 3). As soluções ótimas para esses tipos de problemas poderiam ser encontradas através de uma enumeração completa de todas as soluções possíveis, porém, mesmo com o avanço tecnológico dos computadores nas últimas décadas, isto torna-se impraticável à medida que o tamanho do problema aumenta [CHA05].

As heurísticas (da antiga palavra grega “heuriskein”, que significa descobrir, inventar, ter uma idéia) para otimização, são métodos aproximados de busca. As heurísticas ou também denominadas algoritmos heurísticos foram desenvolvidos com o propósito de resolver problemas de elevado nível de complexidade em tempo computacional razoável. As heurísticas procuram encontrar soluções próximas da otimalidade em um tempo computacional razoável, sem, no entanto, conseguir definir se esta é a solução ótima, nem quão próxima ela está da solução ótima [CHA05]. Mas especificamente, as heurísticas de construção para o PCV são algoritmos que geram um circuito viável partindo de um conjunto inicial (que pode ser vazio) de vértices e/ou arestas, e modificando esse conjunto a cada iteração utilizando algum critério de escolha [RIB02]. Exemplos de heurísticas são a *2-opt* [CRO58], *3-opt* [BOC58], Lin-Kernighan [LIN73] e suas variantes.

As metaheurísticas são procedimentos destinados a encontrar uma boa solução, eventualmente a ótima, consistindo na aplicação, em cada iteração ou em determinada etapa,

de uma heurística subordinada, a qual tem que ser modelada para cada problema específico. A principal característica das metaheurísticas é a capacidade que estas possuem de escapar de ótimos locais. A metaheurística pode ser considerada como uma evolução dos algoritmos heurísticos [ROM04].

Recentemente, muito esforço tem sido concentrado no estudo de métodos novos, modificados ou híbridos para aplicações em PCV. A seguir nas próximas seções são apresentados os fundamentos do PCV, teoria da complexidade, mecânica quântica e computação quântica. Além disso, é apresentado um apanhado da literatura recente e métodos de resolução para o PCV.

2.1. Problema do Caixeiro Viajante: Fundamentos e Breve Histórico

O Problema do Caixeiro Viajante – PCV – mais conhecido na literatura como *Traveling Salesman Problem*, é um problema clássico mais proeminente dentre um amplo conjunto de problemas de Otimização Combinatória, consistindo em determinar, num grafo ponderado, um ciclo Hamiltoniano de custo mínimo. O PCV é classificado como NP-difícil intratável, segundo [GAR79] e [REI94]. Devido à explosão combinatória inerente ao PCV, ele faz parte da classe de problemas que se acredita ser de um problema insolúvel em tempo polinomial.

O estudo do PCV tem atraído pesquisadores de diferentes áreas de conhecimento, entre as quais Pesquisa Operacional, Matemática, Física, Biologia, Inteligência Artificial, Computação e Engenharia. Isso se deve ao fato de que, apesar da simplicidade da sua formulação, no PCV é possível encontrar a maioria das questões que envolvem Otimização Combinatória. Conseqüentemente, o PCV tem sido usado como problema teste para avaliação de novos algoritmos e estratégias de busca e otimização.

O PCV em sua forma mais geral consiste de um vendedor (caixeiro viajante) que pretenda visitar uma única vez cada uma das cidades (ciclo Hamiltoniano) que constam de uma lista e regressar à cidade donde partiu. Neste caso, admite-se também que ele conhece a distância da viagem entre qualquer par de cidades, ele obtenha uma seqüência de cidades (rota) que constitui o percurso associado a um custo total que seja o menor possível.

Um breve histórico sobre o problema matemático pode ser encontrado no *website* desenvolvido por [APP95a], que foi um dos desenvolvedores do método Concorde, este um dos métodos mais eficientes já criados para resolução do PCV.

Acredita-se que um dos trabalhos pioneiros quanto ao problema matemático do caixeiro viajante foi do matemático irlandês Sir Willian Rowan Hamilton e o matemático britânico Thomas Penyngton Kirkman no século 19, através de um jogo, aonde você tinha que conseguir fazer uma “viagem” por 20 pontos, e para isto podendo utilizar apenas algumas conexões específicas entre os pontos. No entanto, Goldbarg e Luna [GOL00] relatam que Hamilton não foi o primeiro a apresentar o problema, mas o seu jogo ajudou a divulgá-lo. Os autores mencionam ainda que modernamente, a primeira menção conhecida do PCV é devida a Hassler Whitney, em 1934, em um trabalho na Princeton University. Com o passar do tempo, a quantidade de versões do problema aumentou, sendo que algumas delas apresentam particularidades que tornam mais fáceis às tentativas de solução.

Em meados de 1920, o matemático e economista Karl Menger, mais conhecido na época por seu interesse em Geometria Probabilística e Álgebra tenha o publicado em Viena. Em 1932, Menger publicou "*Das botenproblem*" nos *Ergebnisse eines Mathematischen Kolloquiums* (resultados de um colóquio matemático). Este volume contém o problema proposto por Menger em 5 de fevereiro de 1930, no colóquio de matemática de Viena. Na época, Menger o chamou de “Problema do Mensageiro”, um problema comum encontrado por mensageiros do serviço postal assim como por muitos viajantes. O problema foi descrito da seguinte forma: “*A tarefa de encontrar, em um determinado número finito de pontos com distâncias conhecidas par a par, o menor caminho entre os pontos. A regra que ir do ponto inicial ao ponto mais próximo e assim sucessivamente pode nem sempre resultar no caminho mais curto*”. De fato atualmente pode-se definir o PCV como sendo: “*Dado um número N de cidades e os custos (distância) de viajar de uma cidade a outra, qual é a rota mais barata em que se visita cada cidade exatamente uma vez e retorna então à cidade de origem?*”.

Desde então já apareceram várias estudos e técnicas para se achar o melhor caminho, algumas delas serão mencionadas neste trabalho, mas sem o intuito de apresentar-se um histórico completo quanto às abordagens e progressos em termos de aplicações e resolução do PCV.

Em 1962, por exemplo, houve um concurso para quem conseguisse achar o melhor caminho para os patrulheiros passarem por 33 cidades. Nesse caso, o ganhador foi o professor da Universidade de Carnegie, o senhor Gerald Thompson.

Segundo o mencionado em [APP95a], em 1977, Groetschel encontrou o melhor caminho para as maiores 120 cidades da Alemanha Ocidental. Em 1987, Padberg e Rinaldi [PAD87] obtiveram a solução para 532 cidades dos Estados Unidos da América, em uma pesquisa realizada na AT&T que na época era maior empresa de telefonia do mundo.

Em 1995, os pesquisadores David Applegate (laboratórios da AT&T Bell), Robert Bixty (Rice University), Vasek Chvátal (RutgersUniversity) e William Cook (Bellcore) [APP95b] obtiveram um novo recorde quanto a melhor solução do PCV para um caso abrangendo 7397 cidades. Em 1998, estes mesmos pesquisadores [APP98] quebraram a barreira dos PCVs de 10000 cidades lidando com um problema de 13509 cidades, que consistiam na época, de todas as cidades que continham mais de 500 habitantes nos Estados Unidos da América. Mais recentemente, em 2001, estes pesquisadores quebraram um novo obtendo a melhor rota para se visitar todas as 15112 cidades da Alemanha reunificada.

Em 2007, [NGU07] Nguyen et al encontraram o melhor resultado conhecido para a instancia de 1.904.711 cidades, conhecido como *World TSP Challenge*. O melhor valor havia sido conseguido por Helsgaun com LKH. Os testes foram executados em 32 maquinas clusterizadas com processador Pentium III 933MHz. Entretanto a solução encontrada (7.518.528.361m) ainda pode ser melhorada utilizando o mesmo algoritmo na ordem de 15.000 m/dia.

2.1.1. Modelagem Matemática do PCV

Matematicamente o PCV se encontra baseado em conceitos da Teoria de Grafos, onde pode ser tratado como um ciclo Hamiltoniano. Utilizando a notação de grafos, pode-se definir o problema como sendo $G = (V, E)$ um grafo (direto ou indireto) e conjunto F uma família de todos os ciclos Hamiltonianos de G . Para cada aresta $a \in A$ existe um custo c_a associado. O problema do caixeiro viajante é encontrar uma rota (ciclo Hamiltoniano) em G , onde a soma dos custos das arestas seja o menor possível. Na Figura 2.1 é apresentada uma representação de uma rota para um PCV.



Figura 2.1. Exemplo de uma rota para o PCV.

O PCV pode ser representado como um problema de permutação. Assumindo P_n uma coleção de todas as permutações do conjunto $A = \{1, 2, \dots, n\}$. A meta para a solução do PCV é determinar $\Pi = (\Pi(1), \Pi(2), \dots, \Pi(n))$ em P_n para que $\left(\sum_{i=1}^{m-1} d(c_{\Pi(i)}, c_{\Pi(i+1)}) \right) + d(c_{\Pi(m)}, c_{\Pi(1)})$ seja minimizado. Por exemplo, se $n = 5$ e $\Pi = (3, 4, 1, 5, 2)$, então a rota correspondente será (3-4-1-5-2).

Matematicamente o PCV é definido como um problema de decisão e não de otimização. Sendo assim a resposta fundamental que devemos dar é se existe ou não uma rota com distância total mais curta. Neste contexto, o problema de decisão pode ser modelado da seguinte forma:

Temos um conjunto finito $C = \{c_1, c_2, \dots, c_m\}$ de cidades à distância $d(c_i, c_j) \in \mathbb{Z}^+$ para cada par de cidades $c_i, c_j \in C$ e o custo total inicial $T \in \mathbb{Z}^+$, onde \mathbb{Z}^+ é o conjunto de inteiros positivos, tal que:

$$\left(\sum_{i=1}^{m-1} d(c_{\Pi(i)}, c_{\Pi(i+1)}) \right) + d(c_{\Pi(m)}, c_{\Pi(1)}) \leq T \quad (2.1)$$

2.1.2. Teoria da Complexidade

O PCV está incluso no contexto dos problemas de Otimização Combinatória e para este tipo de problema os fundamentos da teoria da complexidade são importantes para a

realização de análises de complexidade computacional dos algoritmos propostas para sua solução.

Na classe de problemas combinatórios, o objetivo é assinalar valores a um conjunto de variáveis de decisão, de tal modo que uma função dessas variáveis (função objetivo) seja minimizada (ou maximizada) na presença de um conjunto de restrições. Formalmente, um problema de Otimização Combinatória é definido através de um conjunto finito $N = \{1, \dots, n\}$, com ponderações (ou pesos) c_j associadas a cada $j \in N$, e também um conjunto F formado por subconjuntos viáveis de N . Deseja-se determinar os elementos de F , tais que o somatório das ponderações associadas seja mínimo, isto é, determinar:

$$\min_{S \subseteq N} \left\{ \sum_{j \in S} c_j : S \in F \right\}. \quad (2.2)$$

Em problemas deste tipo, uma estratégia trivial para obtenção de soluções ótimas consiste na avaliação de todas as soluções viáveis e na escolha daquela que minimize o somatório das ponderações. O único inconveniente dessa estratégia está na explosão combinatória. Tomando como exemplo, um PCV com n cidades conectadas par a par, o número de soluções viáveis é da ordem $\frac{(n-1)!}{2}$. Nota-se na Tabela 2-1 o número de possibilidades para alguns valores de n , juntamente com o tempo estimado para se resolver o problema usando essa estratégia, supondo um computador que possa avaliar 10^{12} soluções por segundo [ROD00].

Tabela 2-1. Explosão combinatória e o tempo estimado.

n	$\frac{(n-1)!}{2}$	Dias	Anos	10^7 Anos
10	$1,81 \times 10^5$	$2,10 \times 10^{-13}$	$5,75 \times 10^{-16}$	$5,75 \times 10^{-23}$
100	$4,67 \times 10^{155}$	$5,40 \times 10^{137}$	$1,48 \times 10^{135}$	$1,48 \times 10^{128}$
1000	$2,01 \times 10^{2564}$	$2,33 \times 10^{2546}$	$6,38 \times 10^{2543}$	$6,38 \times 10^{2536}$

Conforme observado na Tabela 2-1, o tempo necessário para utilização dessa abordagem é proibitivo mesmo para valores pequenos de n . Assim, mecanismos mais eficientes devem ser empregados na resolução deste tipo de problema.

O PCV não é um problema de fácil resolução. A razão disso não está no fato de existirem muitas soluções possíveis. Com efeito, um outro problema bem conhecido é a determinação da árvore de custo mínimo em um grafo completamente conectado. Neste caso, o número de árvores possíveis é, muitas vezes, maior que o número de rotas que satisfazem as restrições do PCV. Contudo, existem maneiras eficientes de encontrar uma solução de custo mínimo para o caso da árvore.

Na Teoria da Complexidade (TC) são fornecidos critérios para avaliação da dificuldade de resolução do PCV e de outros problemas. Inicialmente, em TC, um problema é definido como uma questão geral para a qual deve ser dada uma resposta, podendo tal questão ter muitas variáveis (ou parâmetros), cujos valores estão em aberto. Uma instância de um problema é obtida através da fixação desses valores e da especificação de quais propriedades uma solução para o problema deve possuir [ROD00].

Formalmente, definem-se problemas através de um esquema de representação, formado por seqüências (“palavras”) de 1’s e 0’s que representam as instâncias e as soluções do problema. Com efeito, denomina-se problema a um subconjunto Π de $\{0,1\}^* \times \{0,1\}^*$, onde $\{0,1\}^*$ denota o conjunto de todas as seqüências finitas de 0’s e 1’s. Cada seqüência $\sigma \in \{0,1\}^*$ é denominada instância ou entrada de Π e cada $\tau \in \{0,1\}^*$ tal que $(\sigma, \tau) \in \Pi$ é denominada solução ou saída de Π .

Assume-se que para cada entrada em Π existe pelo menos uma solução. Para um esquema de representação específico, define-se tamanho de uma entrada (L) para uma instância de um problema, como o número de elementos que compõem a seqüência que representa tal entrada. Um algoritmo é uma seqüência de passos que devem ser executados para a obtenção de uma solução para um problema. Neste caso, algoritmos diferentes podem ser propostos para resolver um mesmo problema [COO71].

Para um esquema de representação específico de um problema, a função de complexidade de tempo $f: N \rightarrow N$ de um algoritmo, expressa o máximo de tempo (operações elementares) necessário para resolver qualquer instância de tamanho $n \in N$.

Um algoritmo é denominado algoritmo de tempo polinomial, se sua função de complexidade de tempo f é tal que $f(n) \leq p(n)$ para todo $n \in N$, para algum polinômio P .

Existe uma classe de problemas denominada problemas de decisão. Tais problemas possuem apenas duas soluções possíveis (ou saídas), quais sejam “sim” ou “não”. A classe formada por todos os problemas de decisão que possuem um algoritmo de tempo polinomial é denominada classe P.

Uma outra classe de problemas de decisão é a denominada classe NP. Esta classe é formada por todos os problemas de decisão com a seguinte propriedade: “Se a resposta para uma instância de P é “sim”, então este fato pode ser provado em tempo polinomial”.

Observa-se, neste contexto, que $P \subseteq NP$ e também acredita-se que $P \neq NP$, embora não haja prova desse fato.

Uma transformação polinomial é um algoritmo que dada uma instância codificada de um problema de decisão Π , é capaz de transformá-la em tempo polinomial numa instância codificada Π' tal que: para toda instância $\sigma \in \Pi$ a resposta para σ é “sim”, se e somente a resposta para a transformação de σ em uma instância $\sigma' \in \Pi'$ é “sim”.

Um problema de Otimização Combinatória não é um problema de decisão. No entanto, um problema de Otimização Combinatória pode ser transformado em um problema de decisão através do seguinte argumento:

Seja o problema de otimização regido pela expressão:

$$\min\{cx : x \in F\}, \tag{2.3}$$

onde x é uma representação de F . Por conseguinte, esse problema pode ser substituído pelo problema de decisão: “há um $x \in F$ tal que $cx \leq k$ ”. Supondo que exista um algoritmo capaz de resolver o problema de minimização em tempo polinomial, então o problema de decisão também pode se resolvido em tempo polinomial, do seguinte modo: resolver o problema de minimização, em seguida o de decisão, comparando a solução gerada pelo primeiro com o valor k .

Por outro lado, se existe um algoritmo capaz de resolver o problema de decisão em tempo polinomial, o problema de minimização também pode ser resolvido através de sucessivos questionamentos feitos para diferentes valores de k .

Sejam os problemas Π e Π' . Informalmente, uma redução de Turing de tempo polinomial de Π em Π' é um algoritmo A que resolve Π pelo uso de uma sub-rotina

hipotética A' para resolver e Π' de tal modo que se A' fosse um algoritmo de tempo polinomial para Π' , então A seria um algoritmo de tempo polinomial para Π .

Um problema de decisão Π é dito NP-completo, se P pertence a NP e todo problema em NP pode ser transformado em tempo polinomial para P . Como consequência dessa definição, tem-se que, se um problema NP-completo puder ser resolvido em tempo polinomial então todos os problemas NP também poderão ser resolvidos. Neste sentido, os problemas NP-completos, são os problemas mais difíceis da classe NP.

Um problema Π é chamado NP-fácil se existe um problema $\Pi' \in NP$ tal que Π pode ser Turing reduzido a Π' . Um problema Π é denominado NP-difícil se existe um problema de decisão Π' NP-completo, tal que Π' pode ser Turing reduzido a Π .

O termo Complexidade Computacional está relacionado com o estudo dos problemas e dos algoritmos capazes de resolvê-los, ou seja, a Complexidade Computacional é um ramo da Matemática Computacional que estuda a eficiência de algoritmos.

Em termos de complexidade de um algoritmo nos problemas de otimização pode-se lidar com aspectos de complexidade no espaço (memória necessária para se resolver um problema) e no tempo (necessário para se obter uma solução satisfatória). Do ponto de vista prático, de nada nos adianta um algoritmo “perfeito” se sua implementação computacional demora uma centena de anos para ser processada. Mesmo as tarefas relativamente simples, como o produto de dois números com muitos dígitos, pode demorar alguns minutos para serem concluídas nos atuais computadores. Se considerar-se que alguns algoritmos necessitam multiplicar números muito grandes milhares de vezes esses alguns minutos podem se transformar em um tempo excessivamente longo.

Para medir a eficiência de um algoritmo frequentemente usa-se o tempo teórico que o programa leva para encontrar uma resposta em função dos dados de entrada. Este cálculo é feito associando-se uma unidade de tempo para cada operação básica que o procedimento executa. Se a dependência do tempo com relação aos dados de entrada for polinomial, o programa é considerado rápido. Se, entretanto, a dependência do tempo for exponencial o programa é considerado lento. Pode-se também perguntar sobre os programas que estão entre estas duas classes.

Definição: A classe de algoritmos P é formada pelos procedimentos para os quais existe um polinômio $p(n)$ que limita o número de passos do processamento se este for iniciado com uma entrada de tamanho n .

Por exemplo, o algoritmo da eliminação de Gauss, ou método do escalonamento, usado para resolver sistemas lineares, é um procedimento da classe P. De fato, para se resolver um sistema linear $n \times n$ é necessário $\frac{4n^3 + 9n^2 - 7n}{6}$ operações aritméticas, ou seja, apresenta um custo computacional aproximado de $O(n^3)$.

Por outro lado, o método de Cramer, também utilizado para resolver sistemas lineares, tem custo exponencial. Para um sistema linear $n \times n$ esse método despende aproximadamente $(n+1)!(e-1)$ operações de multiplicação. Para se ter uma idéia desse custo, se um computador que realiza 10^7 milhões de multiplicações por segundo for utilizado para resolver um sistema linear 20×20 seriam necessários mais de $3,2 \times 10^4$ anos de processamento [MAL02].

Os algoritmos NP não se referem aos procedimentos não polinomiais (na verdade, isto é uma conjectura). A leitura correta para procedimentos NP é dizer que se referem aos algoritmos “não-determinísticos polinomiais” no tempo. A classe NP é definida logo a seguir, mas, no entanto, antes será realizado um breve histórico de suas origens.

No início dos anos 1960 foram encontrados muitos algoritmos que resistiam a uma simplificação polinomial, isto é, algoritmos que não admitiam procedimentos análogos na classe P. Nesta época, Steve Cook [COO71] observou um fato simples e ao mesmo tempo surpreendente: se um problema pudesse ser resolvido em tempo polinomial, poderia se verificar também se uma dada possível solução é “correta” em tempo polinomial (diz-se que o algoritmo pode ser certificado em tempo polinomial).

Um exemplo simples retirado de [MAL02] de como esta certificação pode ser realizada é o problema de descobrir se um dado número é composto, ou seja, se ele não é primo. Suponha que seja necessário descobrir se 4294967297 é um número composto. Não existe uma maneira eficiente (rápida) de fazer isto. De fato tal tarefa pode ser realizada pela utilização do crivo de Eratóstenes, testando os possíveis divisores do número, o que pode demandar um tempo computacional excessivo. Entretanto, existe uma maneira sucinta de certificar que aquele número é composto: basta verificar que o produto de 6700417 por 641 é exatamente 4294967297. Assim, se for possível obter uma certificação, pode-se exibir efetivamente sua validade. No entanto, achá-la pode ser extremamente difícil. A fatoração do

número 4294967297 foi encontrada por Leonard Euler, em 1732, ou seja, 92 anos após Pierre de Fermat[SIN98] ter proposto erroneamente a conjectura que tal número era primo.

Definição: A classe dos problemas NP é aquela para as quais se pode verificar, em tempo polinomial, se uma possível solução é correta.

Evidentemente $P \subset NP$. De fato, se um algoritmo pode ser executado em tempo polinomial e um possível candidato S para que a solução esteja disponível, é possível executar o programa, obter uma solução correta C e comparar C com S para certificar que S é de fato solução, sendo que todas as operações são realizadas em tempo polinomial.

2.1.3. Métodos de Resolução do PCV

Pode-se pensar em otimização como sendo o procedimento de busca da melhor solução para um problema, esta possível de ser encontrada em um tempo finito. O compromisso entre a qualidade da solução e o tempo necessário para se obter tal solução é o que determina a qualidade de um processo de otimização.

Existem muitas variantes para o PCV que normalmente foram criadas para poder resolver um determinado problema na qual apenas o PCV clássico não poderia resolver de maneira satisfatória, incluindo-se, neste caso, formulações de PCV simétrico, generalizado, com Backhauls, com janelas de tempo, múltiplo, com a presença de gargalos, com bônus, seletivo, estocástico, entre outros [GOL00].

Neste contexto, os diferentes métodos matemáticos para resolução do PCV podem ser divididos em duas categorias [ROD00]: (i) métodos exatos: estes métodos são aqueles que têm como característica a capacidade de determinar sempre uma solução ótima para o problema; e (ii) métodos aproximados (heurísticas e metaheurísticas): estes são aqueles métodos que não garantem a determinação de soluções ótimas, embora eventualmente as encontrem.

A seguir nas próximas subseções são detalhados alguns aspectos dos métodos de resolução do PCV e também um apanhado da literatura recente de abordagens utilizadas para resolver o PCV.

2.1.3.1. Métodos Exatos

Um procedimento clássico utilizado na resolução de diversos tipos de problemas de otimização é conhecido por *branch-and-bound*.

O princípio do *branch-and-bound* é a enumeração de todas as soluções viáveis de um problema de otimização combinatorial, diga-se um problema de minimização, tal que as propriedades ou os atributos não compartilhados por qualquer solução ótima são detectados tão cedo quanto possível. Um atributo (ou ramo da árvore de enumeração) define um subconjunto do conjunto de todas as soluções viáveis do problema original onde cada elemento do subconjunto satisfaz este atributo [BLA96].

Em síntese, o *branch-and-bound* resolve problemas de otimização discreta, dividindo o conjunto de soluções viáveis em sucessivos subconjuntos menores, calculando limites inferiores para a função objetivo em cada um desses subconjuntos. O *branch-and-bound* utiliza essa informação para descartar alguns desses subconjuntos de futuras considerações. Esses limites são obtidos pela substituição do problema em questão, por um conjunto de subproblemas mais fáceis de serem resolvidos (relaxações). O procedimento termina quando cada subconjunto produziu uma solução viável ou quando se demonstra que não é possível determinar uma solução melhor que uma já obtida. Ao final do procedimento, a melhor solução encontrada é uma solução ótima.

O estado da arte entre os algoritmos exatos é formado por algoritmos que utilizam a abordagem *branch-and-cut*. Um algoritmo de *branch-and-cut* é um algoritmo de *branch-and-bound* nos quais planos são gerados ao longo da árvore de busca. A mudança provocada por essa filosofia está no fato de que a busca por soluções rápidas em cada nó é substituída pela procura por limites mais apertados. Com esse tipo de procedimento, os problemas abrangendo mais de 2000 cidades passaram a ser resolvidos com sucesso, ou seja, obtendo-se o valor ótimo para a função objetivo.

Padberg e Rinald [PAD91] resolveram o problema PR2392 da TSPLIB [REI91] usando um procedimento com esse tipo de abordagem. Para isso, utilizaram uma configuração *hardware* poderosa e também um *software* extremamente complexo. Em relação a complexidade do código computacional, [PAD91] mencionam que “o código fonte do *software* é composto por aproximadamente 120 rotinas com cerca de 8500 linhas de código, não incluídos os comentários e o programa responsável pela resolução do problema de programação linear”. O tempo despendido por este algoritmo na resolução do problema

PR2392 foi de 4,3 horas (executado em um computador IBM 3090/600). Neste caos, a relação entre tempo de execução do *software* e a dimensão do problema abordado apresentou comportamento exponencial. Portanto, limitando a utilização do procedimento a supercomputadores, no caso de PCVs de grande porte.

Mais recentemente, em 1995, Applegate *et al.* [APP95b] determinaram o valor ótimo para diversos problemas da TSPLIB com tamanhos variando entre 225 a 7397 cidades. Para isso utilizaram uma rede de estações de trabalho UNIX (não são mencionados detalhes sobre a quantidade ou tipo das estações e nem sobre o tempo demandado com o procedimento de otimização), utilizando um procedimento que consiste de uma combinação de diversos algoritmos, inclusive o proposto em [PAD91]. Johnson e McGeoch [JOH97] dão uma pista sobre o tempo demandado na execução de tal abordagem computacional: “3 a 4 anos do tempo de CPU em uma rede de máquinas do porte de uma SPARCStation 2”.

O avanço histórico observado na resolução de PCVs simétricos com matrizes de distâncias não aleatórias para o ótimo e em função do número de cidades é o seguinte: 49 cidades em 1954 [DAN54], 120 cidades [GRO80], 318 cidades [CRO80], 2392 cidades [PAD91] e 7397 cidades [APP95b].

Entretanto, existem problemas práticos que apresentam grandeza na ordem de dezenas de milhares de vértices, ou seja, muito acima desses limites. Em relação as dificuldades de utilização de métodos exatos, Junger *et al.* [JUN93] mencionam que “os pesquisadores interessados em resolver problemas práticos com esse tipo de procedimento, têm descrito tais algoritmos como impraticáveis”.

2.1.3.2. Métodos Heurísticos

As soluções exatas para problemas de otimização combinatória raramente existem para todas as instâncias do problema. O PCV é um problema de difícil tratamento onde soluções para um grande número de vértices consomem um longo tempo de processamento. Os algoritmos que fornecem uma solução exata são mais apropriados para a resolução de instâncias menores do problema. Os métodos aproximativos, tais como heurísticas e metaheurísticas, foram então desenvolvidos para a determinação de soluções genéricas. Tais métodos requerem, muitas vezes, custos computacionais menores que os exigidos pelos

métodos exatos. Ou seja, os algoritmos heurísticos ou aproximados, são desenvolvidos para encontrar soluções próximas da ótima com um tempo de processamento aceitável.

Uma heurística é uma técnica de otimização que através de passos muito bem definidos encontra uma solução de boa qualidade de um problema complexo. Neste contexto, do ponto de vista teórico, uma heurística não tem capacidade de encontrar a solução ótima global de um problema complexo. Segundo Reeves em [RAY96]:

“Uma heurística é uma técnica que busca boas soluções, isto é, soluções próximas do ótimo, com um custo computacional razoável sem garantir a otimalidade, e possivelmente a viabilidade. Inoportunamente pode não ser possível determinar quão próximo uma solução heurística em particular está da solução ótima”.

Uma heurística apresenta a vantagem de ser simples de formular e de implementar computacionalmente, simples de entender e, são rápidos e robustos. Uma heurística realiza um conjunto de transições através do espaço de soluções do problema, iniciando o processo de um ponto do espaço de busca e terminado em um ponto de ótimo local.

A diferença entre os diferentes algoritmos heurísticos está relacionada com a escolha do ponto inicial para iniciar as transições, a caracterização da vizinhança e o critério usado para escolher o próximo ponto, isto é, o melhor vizinho. O processo termina quando todos os vizinhos são de pior qualidade.

O algoritmo heurístico mais popular é o construtivo, onde a cada passo é escolhida uma componente da solução e o processo termina quando é encontrada uma solução factível para o problema. Neste sentido, a metaheurística representa uma evolução em relação aos algoritmos heurísticos clássicos.

Na prática, a despeito dessa conceituação pessimista, as técnicas heurísticas têm sido utilizadas com bastante sucesso em vários tipos de problemas [BLU03], [CHA03], [DEC06]. Mais adiante no Capítulo 3, algumas dessas técnicas são comentados em detalhes.

A seguir serão apresentados os fundamentos do uso de FPGAs (*Field Programmable Gate Arrays*) e alguns conceitos relevantes de Mecânica Quântica e Computação Quântica.

2.2. FPGA – Field Programmable Gate Array

A FPGA foi introduzida pela empresa Xilinx Inc. no ano de 1985 [CHA94]. Esta tecnologia consiste em dispositivos lógicos programáveis que suportam a implementação de circuitos lógicos relativamente grandes.

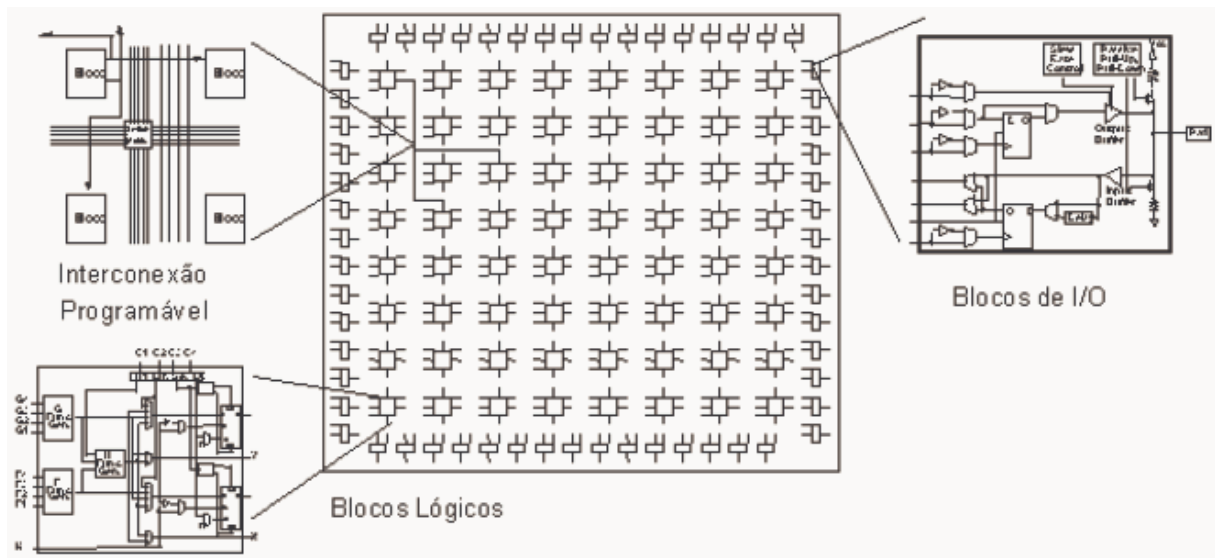


Figura 2.2. Estrutura básica de um *FPGA*

Uma FPGA é formado por um arranjo de células configuráveis, também denominados de bloco lógico, que podem ser utilizados para a implementação de funções lógicas. Uma *FPGA* possui três tipos principais de recursos: blocos lógicos, blocos de entrada e saída, e chaves de interconexão programáveis. Os blocos lógicos formam um arranjo bidimensional e as chaves de interconexão são organizadas como canais de roteamento horizontais e verticais entre as linhas e colunas de blocos lógicos. Cada um destes canais possui chaves programáveis que permitem conectar os blocos lógicos de maneira conveniente, em função da necessidade de cada projeto [MUR95] [OLD95].

Atualmente existem *FPGAs* da ordem de 10 milhões de portas lógicas [XIL03], permitindo a construção de sistemas complexos em um único chip (*SoPC – System on a Programmable Chip*).

Quando um circuito lógico é implementado em uma *FPGA*, os blocos lógicos são programados para realizar as funções necessárias, e os canais de roteamento são estruturados de forma a realizar as interconexões necessárias entre os blocos lógicos. As células de armazenamento dos LUTs (*Look Up Table*) de um *FPGA* são voláteis, o que implica na perda do conteúdo armazenado no caso de falta de alimentação elétrica. Desta forma, o *FPGA* deve

ser programado toda vez que for energizado. Geralmente um pequeno chip de memória PROM(*Programmable Read-Only Memory*) é incluído nas placas de circuito impresso que contem FPGAs. As células de armazenamento são automaticamente carregadas a partir das PROMs toda vez que uma tensão elétrica é aplicada a estes chips.

A seguir é apresentado um resumo de acordo com [JAS04], com as vantagens da utilização de FPGAs. Entre elas pode-se citar:

- Baixo custo de produção e desenvolvimento: quando comparados aos de um ASIC. Componentes dedicados requerem a elaboração de máscaras para a fabricação dos protótipos e dos componentes definitivos, as quais podem ser inutilizadas caso seja necessária uma correção ou atualização do projeto;

- Facilidade de prototipagem: O projeto pode ser realizado em uma estação de trabalho convencional, e facilmente carregado para uma placa de circuito impresso contendo o hardware definitivo do sistema;

- Facilidade para correção de erros de projeto: Como as FPGAs baseadas em SRAM são reconfiguráveis, eventuais erros podem ser prontamente corrigidos pelos projetistas; ao carregar a nova configuração para o dispositivo, o circuito já está pronto para ser testado novamente;

- Capacidade de reconfiguração:. A cada inicialização do sistema, as FPGAs devem ser carregadas com os dados fornecidos por algum componente externo, como uma memória não-volátil, CPLDs, ou um microprocessador;

- Portabilidade: Projetos já implementados podem ser facilmente adaptados para dispositivos mais recentes, ou para similares de outro fabricante. Isto reduz a preocupação com a obsolescência e descontinuidade dos componentes;

- Redução do número de componentes: Uma única FPGA pode integrar um microprocessador, memória RAM, blocos de IP (propriedade intelectual) e lógica de interfaceamento, além de diversos periféricos. A redução do número de componentes traz a vantagem adicional de diminuir o número de pontos de solda e de contatos, aumentando a robustez e a confiabilidade de sistemas destinados à operação em ambientes adversos;

Muitas são as implementações utilizando FPGA aliados a metaheurísticas. Pode-se citar implementações adotando colônia de formigas [SCH04], redes neurais [LIN07], algoritmos genéticos [FRÖ01] e sistemas nebulosos [HSU96].

2.3. Mecânica Quântica

O mundo quântico é inexplicável em termos de mecânica clássica. As previsões pertencentes a interações da matéria e da luz contidas nas leis de Newton para movimento de partícula e as equações de Maxwell que governam a propagação de campos eletromagnéticos são contraditórias em experimentos realizados em escala microscópica. Obviamente deve haver uma linha-limite no tamanho dos objetos para que o comportamento desses se enquadre na mecânica clássica ou na mecânica quântica. Nessa linha-limite estão objetos que são aproximadamente 100 vezes maiores que o tamanho de um átomo de hidrogênio. Os objetos menores que isso tem seu comportamento descrito na mecânica quântica, enquanto que objetos maiores são tratados pela mecânica newtoniana [VIG04].

A mecânica quântica é uma teoria notável. Parece não haver nenhuma dúvida real que muito da física e tudo da química seriam dedutíveis de deus postulados ou leis. Ela tem respondido corretamente muitas questões e tem dado uma visão profunda dos processos naturais, estando preparada para contribuir muito mais. É uma ampla teoria sobre a qual é baseado muito do nosso conhecimento de mecânica e radiação. É uma teoria recente. Em 1900, a mecânica da matéria, baseada nas leis de movimento de Newton, tinha resistido a alterações durante séculos. Assim, do mesmo modo, a teoria ondulatória da luz, baseada na teoria eletromagnética, resistia sem desafios [POH71].

É verdade que a mecânica quântica ainda não apresentou uma descrição consistente das partículas elementares e campos de interação, entretanto a teoria já se encontra completa para experimentos com átomos, moléculas, núcleos, radiações e sólido. Como resultado do rápido crescimento desta teoria no até a metade do século XX pudemos constatar grandes progressos explosivos¹ neste século [MER61].

2.3.1. Efeito Fotoelétrico

Foi em 1886 e 1887 que Henrich Hertz realizou as experiências que pela primeira vez confirmaram a existência de ondas eletromagnéticas e a teoria de Maxwell sobre propagação da Luz [EIS94].

¹ Provável referência do autor a criação das bombas atômicas utilizadas na 2ª guerra mundial.

A experiência mostra que, quando a luz atinge uma superfície metálica limpa, elétrons podem ser emitidos pela superfície. Os resultados mostram que a energia cinética dos elétrons emitidos varia com a frequência da luz incidente. Variando a intensidade da luz de uma dada frequência, varia somente o número de elétrons emitidos por unidade de tempo, mas não a energia cinética de cada elétron. Observa-se que existe uma energia (ou frequência) mínima crítica que a luz deve ter para os elétrons escaparem da superfície. [POH71]

Em 1905 Einstein colocou em questão a teoria clássica da luz, propôs uma nova teoria, e citou o efeito fotoelétrico como sendo uma aplicação que poderia testar qual teoria estava correta. Einstein propôs que a luz era formada por pacotes de energia, *quanta*, que mais tarde após sua comprovação receberam nome de *fótons*. Cada *quanta* de um comprimento de onda carregava a mesma quantidade de energia. Quando estes pacotes de energias se chocavam a contra a parede de um metal, provocam a expulsão dos elétrons da superfície do mesmo gerando assim corrente. Em 1921 Einstein recebeu o premio Nobel por ter previsto teoricamente a lei do efeito fotoelétrico.

2.3.2. Princípio da Incerteza de Heisenberg

Ao se tratar do comportamento dos elétrons nos átomos, não podemos definir simultaneamente a velocidade e a posição de cada (ou qualquer) elétron. Também não se pode, ainda que em princípio, observar continuamente as órbitas individuais ou movimentos dos elétrons. O princípio da incerteza de Heisenberg consiste num enunciado da mecânica quântica, impondo restrições à precisão com que se podem efetuar medidas simultâneas de uma classe de pares observáveis.

Num certo sentido, é como um tratorista cego usando um trator para detectar bolas de pingue-pongue. Os ensaios são rudimentares em relação aos sistemas de elétrons; são tão primitivos que são perturbados consideravelmente. Há um limite bem definido para o conhecimento dos movimentos como foi mostrado por Heisenberg em 1927 com seu principio da incerteza. [POH71]

Pode-se exprimir o princípio da incerteza nos seguintes termos:

O produto da incerteza associada ao valor de uma coordenada x_i e a incerteza associada ao seu correspondente momento linear p_i não pode ser inferior, em grandeza, à constante de *Planck* normalizada. Em termos matemáticos, exprime-se assim:

$$\Delta x_i \Delta p_i \geq \frac{\hbar}{2}, \quad (2.4)$$

onde \hbar é a Constante de *Planck* (h) dividida por 2π .

Quando se quer encontrar a posição de um elétron, por exemplo, é necessário fazê-lo interagir com algum instrumento de medida, direta ou indiretamente. Por exemplo, faz-se incidir sobre ele algum tipo de radiação. Tanto faz aqui que se considere a radiação do modo clássico - constituída por ondas eletromagnéticas - ou do modo quântico - constituída por fótons. Quer-se determinar a posição do elétron, é necessário que a radiação tenha comprimento de onda da ordem da incerteza com que se quer determinar a posição.

Heisenberg demonstrou que a incerteza quanto à posição multiplicada pela incerteza quanto à velocidade nunca pode ser inferior a certa quantidade - a denominada constante de Planck.

Erwin Schrödinger foi um cientista austríaco que, em meados dos anos 1920, desempenhou um papel muito importante no desenvolvimento das equações da Mecânica Quântica. O gato de Schrödinger foi pensado como exemplo para mostrar claramente as diferenças existentes entre o mundo cotidiano e o mundo quântico. Este experimento hipotético foi pensando por Schrödinger para tentar explicar, de forma inteligente, alguns dos princípios da mecânica quântica definidos na Interpretação de Copenhague.

A Interpretação de Copenhague é uma interpretação da mecânica quântica desenvolvida por Werner Heisenberg e Niels Bohr, em 1927, e é composto de princípios da mecânica quântica. A seguir temos alguns destes importantes princípios:

- Um sistema é completamente descrito por sua função de onda, que representa o conhecimento do observador sobre o sistema;
- A descrição da natureza é probabilística. A probabilidade de um evento acontecer esta relacionada ao quadrado da amplitude da função de onda;
- O princípio de incerteza de Heisenberg assegura que não é possível saber o valor de todas as propriedades do sistema ao mesmo tempo;

Estes princípios geraram algum desconforto na comunidade científica da época, movido principalmente pelo debate entre Niels Bohr e Albert Einstein, após a Interpretação de

Copenhague. Schrödinger então propôs seu experimento como uma forma de resposta a Einstein pelas críticas a Interpretação de Copenhague. O problema pode ser enunciado da seguinte forma:

“Um gato está fechado numa câmara de aço, junto ao dispositivo seguinte (que deve assegurar-se contra uma interferência direta por parte do gato); num contador Geiger há um pedacinho de uma substância radioativa, tão pequeno, que talvez no decorrer de uma hora se desintegre um átomo, mas também poderia ocorrer com igual probabilidade que nenhum átomo se desintegrasse; se ocorre o primeiro, produz-se uma descarga no tubo e mediante um relé solta-se um martelo que rompe um frasquinho de ácido cianídrico. A função de onda do sistema inteiro expressa o estado morto e vivo do gato com probabilidades iguais.”
[SCH35]

Ao tentar descrever o que ocorreu no interior da caixa, servindo-se das leis da mecânica quântica, chega-se a uma conclusão estranha. O gato viria descrito por uma função de onda extremamente complexa resultado da superposição de dois estados, combinando 50% de *gato vivo* e 50% de *gato morto*. Ou seja, aplicando-se o formalismo quântico, o gato estaria por sua vez vivo e morto, e isto correspondente a dois estados indistinguíveis.

A única forma de averiguar o que realmente aconteceu com o gato será realizar uma medida: abrir a caixa e olhar dentro. Sendo assim a função de onda que define o sistema colapsa para um dos estados. Em alguns casos encontrar-se-á o gato vivo e em outros um gato morto.

2.3.3. Função de Onda

A Função de Onda é uma ferramenta matemática da mecânica quântica utilizada para descrever um sistema físico. É uma função complexa que descreve completamente o estado de uma partícula $\Psi(x,t)$.

O seu módulo quadrado é a densidade de probabilidade, que é a probabilidade da partícula ser encontrada, tal que:

$$|\Psi(x,t)|^2 dx = \Psi^*(x,t)\Psi(x,t)dx \quad (2.5)$$

A evolução temporal da função de onda é determinada pela equação de Schrödinger descrita na próxima sub-sessão.

2.3.4. Equação de Schrödinger

A maioria das teorias da Física é baseada em equações fundamentais. Por exemplo, a mecânica de Newton é baseada em $F = ma$, eletrodinâmica clássica é baseada na equação de Maxwell e a teoria da relatividade geral é baseada na equação de Einstein $G_{uv} = -8\pi GT_{uv}$.

A equação fundamental da mecânica quântica é a equação de Schrödinger [RYD79], [EIS94]. Pode-se escrever a equação de Schrödinger para uma partícula de massa m se movimentando em um potencial U em apenas uma dimensão x da seguinte forma:

$$-\frac{\hbar^2}{2m} \frac{\partial^2 \Psi}{\partial x^2} + U\Psi = i\hbar \frac{\partial \Psi}{\partial t} \quad (2.6)$$

onde o símbolo Ψ representa a função de onda.

2.4. Computação Quântica

Pode-se dizer que a teoria de computação quântica iniciou-se nos anos 1980, quando Feynman [FEY82] observou que um sistema quântico de partículas, ao contrário de um sistema clássico, parece não poder ser simulado eficientemente em um computador clássico, e sugeriu um computador que explorasse efeitos da física quântica para contornar o problema. Desde então, até 1994, a teoria de computação quântica desenvolveu-se discretamente, com várias contribuições de Deutsch [DEU85, DEU89], Bernstein e Vazirani [BER97], entre outros, que colaboraram fundamentalmente para a formalização de um modelo computacional quântico.

Um computador quântico é um dispositivo que executa cálculos fazendo uso direto de propriedades da mecânica quântica, tais como sobreposição e emaranhamento. Teoricamente, computadores quânticos podem ser implementados e o mais desenvolvido atualmente trabalha

com poucos *qubits* de informação. O principal ganho desses computadores é a possibilidade de resolver em tempo eficiente, alguns problemas que na computação clássica levariam tempo impraticável, como por exemplo: fatoração, busca de informação em bancos não ordenados, etcétera.

Um bit quântico (*Qubit* ou *Q-bit*) é um sistema que, tal como as partículas/ondas, possui dois estados distintos, habitualmente chamados $|0\rangle$ e $|1\rangle$, que podem representar os valores 0 e 1, mas que, para além destes, podem estar em estados que são sobreposição destes dois. Se esse tipo de bit for controlado, poderemos ter computadores com milhares de vezes mais velocidade que os atuais supercomputadores. Um computador com aproximadamente 200 *qubits* pode processar mais do que qualquer computador atualmente. O incremento de um *qubit* eleva exponencialmente a velocidade dos computadores.

Um *quantum bit*, ou *qubit* (às vezes *qbit*) é uma unidade da informação quântica. Essa informação é descrita por um estado em um sistema mecânico quântico de dois níveis, que seja formalmente equivalente a um espaço bidimensional de vetores de números complexos. Os dois estados da base (ou os vetores) são escritos convencionalmente como $|0\rangle$ e $|1\rangle$. Um *qubit* pode ser pensado como uma versão da mecânica quântica para o tradicional bit utilizando em computadores tradicionais. Um estado puro do *qubit* é a superposição linear quântica de dois estados. Isto significa que cada *qubit* pode ser representado como uma combinação linear de $|0\rangle$ e $|1\rangle$:

$$|\varphi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (2.7)$$

onde α e β são probabilidade da amplitude normalmente representada por números complexos.

$$|\alpha|^2 + |\beta|^2 = 1 \quad (2.8)$$

A probabilidade que o *qubit* estará medido no estado $|0\rangle$ é $|\alpha|^2$ e a probabilidade que estará medida no estado $|1\rangle$ é $|\beta|^2$. Sendo assim a probabilidade total do sistema que está sendo observado ou no estado $|0\rangle$ ou $|1\rangle$ é 1.

Mais detalhes podem ser encontrado em [CHU00], [BEC96] e [VIG04].

Capítulo 3.

3 Metodologia usando Heurísticas e Metaheurísticas

A principal dificuldade encontrada para a resolução dos problemas de otimização combinatorial está no esforço computacional empregado para alcançar a solução ótima. Os problemas de grande porte possuem maior dificuldade na sua resolução quando se utilizam os métodos exatos. Para resolver os problemas de otimização combinatorial de grande porte, os métodos heurísticos ou de aproximação fornecem soluções próximas da ótima com maior rapidez e flexibilidade de implementação.

As heurísticas de construção para o PCV são algoritmos que geram um circuito viável partindo de um conjunto inicial (que pode ser vazio) de vértices e/ou arestas, e modificando esse conjunto a cada iteração utilizando algum critério de escolha. Não há garantia de que este circuito seja ótimo.

Neste contexto, uma metaheurística se diferencia de uma heurística por adicionar mais inteligência ao processo de busca por soluções. Tal metodologia procura escapar de ótimos locais e percorrer áreas mais amplas dentro do espaço de soluções possíveis. O desenvolvimento e o estudo das metaheurísticas tem aprofundado de maneira marcante o conhecimento geral sobre o processo de resolução de problemas complexos em otimização combinatorial. Além disso, as metaheurísticas são pontos de partida promissores para a resolução de novos problemas, sem eliminar a possibilidade de incorporação de conhecimentos específicos para a produção de algoritmos mais sofisticados.

Neste capítulo serão apresentados os fundamentos das heurísticas e metaheurísticas usadas neste trabalho para a resolução do PVC, ou seja, LKH, esta uma heurística de melhoria e o enxame (ou nuvem) de partículas com inspiração quântica, uma metaheurística fundamentada na teoria dos enxames e mecânica quântica.

3.1. Heurísticas ou Algoritmos Aproximados

Chong [CHO01] menciona que as heurísticas (algoritmos aproximados ou heurísticos) para o PCV podem ser divididas em três classes:

- Heurísticas construtivas: buscam construir o circuito Hamiltoniano de maneira gradativa, através de adições sequenciais de componentes individuais (variáveis, nós, arcos) um de cada vez até que uma solução seja obtida. Exemplo: heurísticas de inserção, heurísticas de economias;
- Heurísticas de melhoria: a partir de um circuito inicial, buscam melhorar o custo do mesmo através de trocas de posições dos cidades. Exemplo: heurística k -opt (ou também denominada λ -opt), método Lin-Kernighan [LIN73] e método LKH;
- Heurísticas compostas: combinam elementos dos dois procedimentos anteriores.

Os algoritmos construção de rotas constróem gradualmente uma rota adicionando uma cidade nova em cada etapa. Os algoritmos de melhoria de rota melhoram em cima de uma rota existente executando sucessivas trocas. Os algoritmos compostos combinam estas duas características.

Um exemplo simples de um algoritmo da construção da excursão é o algoritmo denominado Vizinho mais Próximo (*Nearest Neighbor*) [ROS77]: Comece em uma cidade arbitrária. Enquanto houver cidades que ainda não tenham sido visitadas, visite a cidade a mais próxima que não tenha aparecido ainda na rota. Finalmente, retorne à primeira cidade.

Esta aproximação é simples, mas frequentemente “gulosa” (*greedy*). As primeiras distâncias no processo da construção são razoavelmente pequenas, enquanto que as distâncias no fim do processo geralmente serão longas. Existem diversos outros algoritmos de construção de rotas que foram desenvolvidos para remediar este problema. Exemplos podem ser encontrados em [LAW85], [LAP92], [REI94].

3.1.1. Heurística de Melhoria k -opt

Um conjunto importante de heurísticas desenvolvidas para o PCV é formado pelas trocas k -opt ou também denominadas y -opt. Em linhas gerais tratam-se de algoritmos que

partindo de uma solução inicial viável para o PCV, realizam permutações de λ arcos que estão no circuito por outros λ arcos fora deste, buscando a cada troca diminuir o custo total do circuito.

O primeiro mecanismo de trocas, denominado troca *2-opt*, foi proposto por Croes [CRO58]. Neste mecanismo, uma nova solução é gerada através da remoção de dois arcos, resultando em dois caminhos. Um dos caminhos é invertido e, em seguida, reconectado para formar uma nova rota, conforme mostra a Figura 3.1. Essa nova solução passa a ser a solução corrente e o processo se repete até que não seja mais possível realizar uma troca de dois arcos com ganho. Daí o nome troca *2-opt*, *opt* de *optimum*, indicando que ao término do processo a solução resultante não pode mais ser melhorada com qualquer troca de dois arcos.

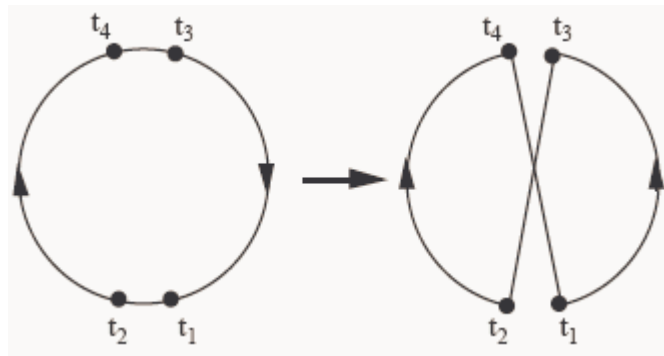


Figura 3.1. Troca *2-opt*.

Shen Lin [LIN65] propõe uma ampliação da vizinhança de busca. No caso das trocas *2-opt*, uma solução corrente tem sua vizinhança representada por todas as soluções alcançáveis, considerando a troca de dois arcos. [LIN65] propõe uma vizinhança mais alargada ao considerar todas as soluções alcançáveis com uma troca de três arcos, conforme mostra a Figura 3.2. Denominou esse mecanismo de troca *3-opt*. Em sua avaliação da qualidade desse procedimento, observou que, em média, os resultados gerados por trocas *3-opt* eram consideravelmente melhores que os da troca *2-opt*, e que a probabilidade de se encontrar valores ótimos também é muito maior. Observou também, que o tempo de execução da busca *3-opt* era maior que o obtido por *2-opt* por um fator de cinco.

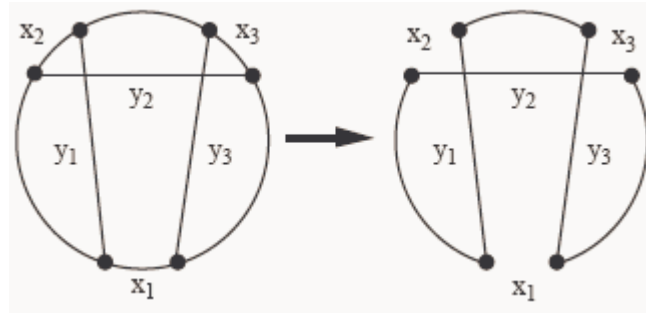


Figura 3.2. Troca 3-*opt*.

Lin sugeriu ainda, que durante o processo de busca das trocas, dever-se-ia considerar a primeira troca favorável em qualquer estágio, em vez de buscar a troca de maior ganho possível na vizinhança (busca gulosa), pois o tempo necessário para esse tipo de busca seria demasiadamente alto.

Lin e Kernighan [LIN73] propõem um algoritmo onde o número de arcos trocados em cada passo é variável. As trocas de arcos são realizadas segundo um critério de ganho que restringe o tamanho da vizinhança de busca. As soluções produzidas são de alta qualidade, superando as obtidas através de trocas 3-*opt* padrão.

3.1.2. Heurística de Melhoria Lin-Kernighan

A heurística Lin-Kernighan é considerada um dos métodos mais eficazes para gerar soluções ótimas ou próximo-do-ótimo para o PCV simétrico. Entretanto, o projeto e a execução de um algoritmo baseado nesta heurística não são triviais. Há muitas decisões do projeto e de execução que devem ser feitas, e a maioria de decisões têm uma influência acentuada no desempenho.

O algoritmo 2-*opt* é um exemplo especial - do algoritmo do k -*opt*, onde em cada etapa λ ligações da rota atual são substituídos por λ ligações de tal maneira que uma rota mais curta é conseguida. Ou seja, em cada etapa, passo, uma rota mais curta é obtida removendo λ ligações e unindo os trajetos resultantes em uma nova maneira, possivelmente invertendo um ou mais deles.

O algoritmo de melhoria k -*opt* é baseado no conceito k -*optimality*, tal que:

“Uma rota é dita ótima (*k-optimal* / *k-opt*) se for impossível obter uma nova rota mais curta, substituindo qualquer λ ligações atuais por outro conjunto de λ ligações.”

A partir desta definição pode-se concluir que qualquer rota *k-optimal* é também *k'-optimal* quando $1 \leq k' \leq k$. Também é possível verificar que uma rota contendo n cidades será ótima se e somente se a rota em questão for *n-optimal*.

No entanto, o número de operações para testar todas as λ -trocas aumenta rapidamente conforme o número de cidades aumenta. Em uma implementação, simples o algoritmo para testar as k -trocas tem a complexidade de $O(n^k)$. Como consequência os valores $k=2$ e $k=3$ são os mais utilizados. Em [CHR72] é possível ver $k=4$ e $k=5$ sendo utilizado.

Entretanto, um inconveniente deste algoritmo é que λ deve ser especificado no começo da execução. É difícil saber que λ se usar para conseguir o melhor custo benefício entre tempo de execução e qualidade da solução.

Lin e Kernighan [LIN73] removeram este inconveniente do algoritmo *k-opt* introduzindo *k-opt variável*. Este algoritmo pode variar o valor de λ em tempo de execução, decidindo em cada iteração qual o valor que λ deve assumir. A cada iteração o algoritmo busca, em valores crescentes de λ , qual variação forma a rota mais curta. Dada um número de trocas r , uma série de testes é executada para verificar quando $r+1$ trocas deve ser considerado. Isto é verificado até que alguma condição de parada seja satisfeita.

A cada passo o algoritmo considera um número crescente de possíveis trocas, começando em $r = 2$. Estas trocas são escolhidas de uma forma que a viabilidade da rota possa ser obtida em qualquer estágio do processo. Se a pesquisa por uma rota menor obter sucesso então a rota anterior é descartada e substituída pela nova rota.

O algoritmo Lin-Kernighan pertence à classe de algoritmos denominados Algoritmos de Otimização Local (*local optimization algorithms*) [JOH90], [JOH 97].

Resumidamente, o algoritmo Lin-Kernighan (LK) consiste dos seguintes passos:

- (i) Gerar um circuito aleatório T inicial.
- (ii) Fazer $G^* = 0$. [G^* representa o melhor ganho alcançado até o momento]. Escolha qualquer nó t_1 e seja x_1 um dos arcos de T adjacentes a t_1 . Adotar $i = 1$.
- (iii) Do outro ponto final t_2 de x_1 escolher y_1 para t_3 com $g_1 = |x_1| - |y_1| > 0$. Se não existe y_1 , vá para o (vi)(d).
- (iv) Fazer $i = i + 1$. Escolher x_i [que no momento liga t_{2i-1} a t_{2i}] e y_i como segue:

(a) x_i é escolhido de modo que, se t_{2i} é ligado com t_1 , a configuração resultante é um circuito.

(b) y_i é algum arco disponível no ponto final t_{2i} compartilhado com x_i , sujeito a (c), (d) e (e). Se não existir y_i , vá para o (v).

(c) Para garantir que os x 's e y 's são disjuntos, x_i não pode ser um arco previamente unido (isto é, um y_j , $j < i$), e similarmente y_i não pode ser um arco previamente quebrado.

$$(d) G_i = \sum_{j=1}^i g_j = \sum_{j=1}^i (|x_j| - |y_j|) > 0 \text{ [Consiste do Critério de ganho].}$$

(e) Em seqüência para garantir que o critério de viabilidade de (a) possa ser satisfeita para $i+1$, o y_i escolhido deve permitir a quebra de x_{i+1} .

(f) Antes de y_i ser construído, deve-se verificar se fechando a ligação de t_{2i} par t_1 será observado ganho maior que o melhor alcançado anteriormente. Faça y_i^* ser um arco conectando t_{2i} a t_1 e faça $g_i^* = |y_i^*| - |x_i|$. Se $G_{i-1} + g_i^* > G^*$, faça $G^* = G_{i-1} + g_i^*$ e $k = i$.

(v) Finalizada a construção de x_i e y_i dentro dos passos de (ii) a (iv) quando não houver mais arcos satisfazendo a os critérios de (iv)(c) a (iv)(e), ou quando $G_i \leq G^*$. Se $G^* > 0$ então considere o novo circuito T' faça $f(T') = f(T) - G^*$ e $T \leftarrow T'$ e vá para o passo 2.

(vi) Se $G^* = 0$, um recuo limitado é realizado como segue:

(a) Repita os passos (iv) e (v) escolhendo y_2 's na ordem crescente do comprimento. Contudo que o critério de ganho $g_1 + g_2 > 0$ seja satisfeito.

(b) Se todas as escolhas de y_2 no (iv)(b) são exauridas sem ganho, retorne ao passo 4(a) e tente escolher outro x_2 .

(c) Se isto também falha, um passo atrás no passo (iii) é promovido, aonde os y_i 's são examinados na ordem do aumento do comprimento.

(d) Se os y_i 's são também exauridos sem ganho, tenta-se alternar x_1 no passo (ii).

(e) Se isto também falhar, um novo t_1 é selecionado, e repete-se o (ii).

(vii) O procedimento termina quando todos os n valores de t_1 tiverem sido examinados sem ganho.

Em linhas gerais, o algoritmo LK parte de um circuito inicial T e de um vértice inicial t_1 . Estando no passo i , uma troca é feita através da seqüência remove-se o arco (t_1, t_{2i}) , e adiciona-se o arco (t_{2i}, t_{2i+1}) ; o arco (t_{2i+1}, t_{2i+2}) é escolhido para ser removido, se com sua remoção e com a adição do arco (t_{2i+2}, t_1) um circuito seja formado. O arco (t_{2i+2}, t_1) é removido se e quando o passo $i+1$ é executado.

O número de inserções e remoções no processo de busca é limitado pelo critério de ganho de LK. Esse critério, basicamente limita a seqüência de trocas àquelas que gerem ganhos positivos (redução do tamanho da rota) a cada passo da seqüência, isto é, são desconsideradas, em princípio, seqüências onde algumas trocas resultem em ganhos positivos e outras em ganhos negativos, mas cujo ganho total da seqüência seja positivo.

Um outro mecanismo presente no algoritmo é a análise de soluções alternativas nos níveis 1 e 2, sempre que uma nova solução é gerada com ganho zero (passo (vi)). Isso é feito através da escolha de novos arcos para troca.

3.1.3. Lin-Kernighan Helsgaun

Keld Helsgaun [HEL00], faz uma revisão do algoritmo Lin-Kernighan original, propondo e desenvolvendo modificações que melhoram seu desempenho. O aumento na eficiência é conseguido, primeiramente, por uma revisão de regras da heurística de Lin e Kernighan para restringir e dirigir a busca. Mesmo que suas regras pareçam naturais, uma análise crítica mostra que estas apresentam deficiências consideráveis.

Analisando as regras do algoritmo LK que são utilizadas para o refinamento da busca, além de poupar tempo computacional, Helsgaun identificou um problema que poderia levar o algoritmo a encontrar soluções pobres para problemas maiores. Uma das regras heurísticas propostas por Lin e Kernighan[LIN73] limita o refinamento da busca apenas aos cinco vizinhos mais próximos. Neste caso se um vizinho que irá propiciar o resultado ótimo não for levando em consideração como um destes cinco vizinhos, a busca não levará ao resultado

ótimo. Helsgaun exemplifica com o problema att532 da TSPLIB proposto por Padberg e Rinaldi[PAD87] onde o melhor vizinho só é encontrado na vigésima segunda busca.

Considerando isso, Helsgaun apresenta uma medida de proximidade que melhor reflete as possibilidades da obtenção de uma aresta que faça parte do percurso ótimo. Esta medida, chamada α -nearness, é baseada na análise de sensibilidade usando uma árvore geradora mínima *1-trees*.

Nos movimentos básicos, pelo menos duas pequenas modificações no esquema geral são realizadas. Primeiro, em um caso especial, o primeiro movimento de uma seqüência deve ser um movimento $4-opt$ seqüencial; os movimentos seguintes devem ainda ser os movimentos $2-opt$. Em segundo, os movimentos $4-opt$ não-seqüenciais são tentados quando o percurso pode não ser melhorado por movimentos seqüenciais.

Além disso, o algoritmo modificado (LK-H) revisa a estrutura básica da busca em outros pontos. O primeiro e principal ponto, é a mudança do movimento básico de $4-opt$ para um movimento $5-opt$ seqüencial. Além deste, experiências computacionais mostraram que o *backtracking* não é muito necessário neste algoritmo e sua remoção reduz o tempo de execução, não compromete o desempenho final do algoritmo e simplifica extremamente a execução do mesmo.

Em relação a percursos iniciais, o algoritmo Lin-Kernighan aplica trocas de arestas diversas vezes no mesmo problema usando percursos iniciais diferentes. As experiências com várias execuções do algoritmo LK-H mostraram que a qualidade das soluções finais não depende fortemente das soluções iniciais. Entretanto, a redução significativa no tempo pode ser conseguida escolhendo soluções iniciais mais próximas do ótimo obtidas a partir de heurísticas construtivas, por exemplo. Desta forma, Helsgaun, em seu trabalho, apresenta também uma heurística simples que utilizou na construção de soluções iniciais em sua versão do Lin-Kernighan [PRE06].

Por outro lado, [NGU06] afirma que a utilização do movimento $5-opt$ como movimento básico eleva grandemente o tempo computacional necessário, embora as soluções encontradas sejam de muito boa qualidade. Além disso, o consumo de memória e tempo necessário ao pré-processamento do algoritmo LK-H são muito elevados quando comparados a outras implementações do LK.

3.1.4. Procedimentos de Busca Local

A busca local, também referida na literatura como busca na vizinhança, é a estratégia base de muitos dos métodos heurísticos utilizados na solução de problemas de otimização. Seja um problema de otimização representado pelo par (S, g) , onde S é um conjunto de soluções viáveis² e g a função objetivo que associa cada elemento $s \in S$ a um número real. Em um problema de minimização, o objetivo é encontrar um elemento $s^* \in S$, tal que $g(s^*) \leq g(s) \forall s \in S$.

Uma vizinhança é definida através de uma função $N: S \rightarrow 2^S$, que associa cada elemento $s \in S$ a um conjunto de soluções alcançáveis com um movimento simples. Entenda-se como movimento simples, uma pequena modificação aplicada em uma solução corrente.

No caso do PCV, por exemplo, para uma solução corrente viável (uma rota válida) a vizinhança pode ser definida como: “todas as rotas geráveis pela substituição de dois arcos presentes na rota corrente por outros dois arcos que não estão nesta rota e que gerem também uma rota viável”.

Uma solução x é chamada de mínimo local com respeito à g , se $g(x) \leq g(y) \forall y \in N(x)$. Os algoritmos de busca local são processos de otimização iterativos. Iniciam com uma solução, e iterativamente, procuram na vizinhança desta solução uma outra de menor custo (ou que gere expectativa de ganho futuro). Se tal solução é encontrada a solução inicial é substituída por esta, e a busca continua. Alguns dos métodos baseados em metaheurísticas (*simulated annealing*, algoritmos genéticos, busca tabu, busca local dirigida, entre outros) e busca gulosa podem ser configurados como procedimentos de busca local.

3.2. Metaheurísticas

As metaheurísticas representam um conjunto de técnicas de otimização adaptadas para lidarem com problemas complexos e que apresentam características de explosão combinatória. Essas técnicas foram desenvolvidas nas duas décadas passadas [OCH94] e são consideradas como as técnicas que devem apresentar maior evolução e utilização na resolução de problemas de otimização matemática e que requerem tempo de processamento elevado.

A metaheurística pode ser considerada como uma evolução [ROM04] dos algoritmos heurísticos. Uma heurística é uma técnica de otimização que através de passos muito bem definidos encontra uma solução de boa qualidade de um problema complexo. Neste contexto, do ponto de vista teórico, uma heurística não tem capacidade de encontrar a solução ótima global de um problema complexo.

Geralmente em problemas de grande porte e complexos, uma heurística encontra apenas um ótimo local geralmente de pobre qualidade. Entretanto, uma heurística apresenta a vantagem de ser muito simples e formular e de programar computacionalmente, muito simples de entender e, são rápidos e robustos. Uma heurística realiza um conjunto de transições através do espaço de soluções do problema, iniciando o processo de um ponto do espaço de busca e terminado em um ponto de ótimo local.

A diferença entre os diferentes algoritmos heurísticos está relacionada com a escolha do ponto inicial para iniciar as transições, a caracterização da vizinhança e o critério usado para escolher o próximo ponto, isto é, o melhor vizinho. O processo termina quando todos os vizinhos são de pior qualidade.

O algoritmo heurístico mais popular é o construtivo, onde a cada fase (iteração ou etapa) é escolhida uma componente da solução e o processo termina quando é encontrada uma solução factível para o problema. Neste sentido, a metaheurística representa uma evolução em relação aos algoritmos heurísticos clássicos.

Dentre os procedimentos que podem ser classificados como metaheurísticas (ou heurísticas inteligentes) que surgiram ao longo das últimas décadas, destacam-se abordagens bio-inspiradas da computação evolutiva (por exemplo, algoritmos genéticos) [JUN02], [TSA04], *simulated annealing* [REE96], colônia de formigas [ROO00], [DOR04], enxame (ou nuvem) de partículas [CLE05], [LOP05], sistemas imunológicos artificiais [JIA00], redes neurais artificiais [SPE89], [SIQ05], sistemas nebulosos [PAN04a], busca tabu [GLO98], busca dispersa (*scatter search*) [LAG03], transgenética computacional [RAM05], *Greedy Randomized Adaptive Search Procedure* (GRASP) [FEO95], *Variable Neighborhood Search* (VNS) [MLA97], entre outras [BLU03], [CHA03], [DEC06]. Estas técnicas, por serem de propósito geral podem por meio de adaptações serem usadas para a procura de soluções aceitáveis para vários problemas de Otimização Combinatória, como é o caso do PCV. A seguir são apresentados alguns fundamentos sobre as heurísticas mencionadas.

Computação evolutiva: O paradigma computacional da computação evolutiva ou evolucionária abrange algoritmos bio-inspirados (uma analogia a teoria evolucionista

NeoDarwiniana) como um processo adaptativo de busca e otimização que possibilite implementações computacionais baseados em teorias de seleção natural e genética. A computação evolutiva sugere um mecanismo em que uma população de indivíduos (soluções potenciais de um problema de otimização) visa melhorar, em média, a sua adequação (*fitness*) em relação ao ambiente, ou seja, o seu desempenho geral com respeito a um dado problema, usando operadores de cruzamento (*crossover*) e/ou mutação. A computação evolutiva abrange as técnicas clássicas de algoritmos genéticas, estratégias evolutivas, programação genética, programação genética e sistemas classificadores, entre outros.

Simulated annealing: O nome *annealing* (anelamento ou recozimento) está relacionado ao processo de aquecimento de um sólido até atingir seu ponto de fusão, seguido de um resfriamento até atingir seu estado sólido. O *simulated annealing* é um algoritmo inspirado em sistemas físicos que inicia com uma solução inicial, gerando outras soluções vizinhas baseado no clássico algoritmo Metropolis, e calcula os custos de todas elas, se o custo for menor, então esta nova solução é aceita, caso contrário a solução pode ser descartada ou não. O *simulated annealing* permite a aceitação de uma configuração que forneça um “pior” valor para a função objetivo evitando, assim, a convergência para um mínimo local. Essa aceitação é determinada por um número aleatório e é controlada através de uma probabilidade.

Colônia de formigas: As formigas são insetos sociais que possuem um sistema complexo de organização e divisão de tarefas, cuja principal função é garantir a sobrevivência do formigueiro. A metaheurística da colônia de formigas foi inspirada na observação das colônias de formigas reais, em particular em como elas encontram o menor caminho entre a fonte de alimentos e o formigueiro. Inicialmente, as formigas percorrem de modo aleatório a região próxima ao formigueiro em busca do alimento. Cada formiga, enquanto percorre o seu caminho, deposita sobre o solo uma substância denominada feromônio, formando um caminho ou rastro de feromônio. As formigas subsequentes detectam a presença desta substância e tendem a escolher o caminho marcado com a maior concentração de feromônio. O feromônio, portanto, além de possibilitar a formação de um caminho de volta para a formiga, também tem a função de informar as outras formigas sobre quais os melhores caminhos até o alimento. Depois de algum tempo, os caminhos mais eficientes – ou de menor distância percorrida até o alimento – acumulam uma quantidade maior de feromônio. Inversamente, os caminhos menos eficientes – ou de maior distância percorrida até o alimento – apresentam uma pequena concentração de feromônio, devido ao menor número de formigas

que passaram por ele e ao processo de evaporação natural do feromônio. No problema de otimização que o formigueiro se defronta, cada formiga é capaz de construir uma solução completa do problema; contudo, a melhor solução só é obtida mediante cruzamento das diversas soluções encontradas.

Enxame (ou nuvem) de partículas - PSO: O algoritmo PSO é um procedimento de otimização estocástica inspirado em princípios de cooperação e comportamento em sociedade de enxames, cardume de peixes e bandos de pássaros. O algoritmo PSO foi proposto por Kennedy e Eberhart [KEN95] e rapidamente o algoritmo popularizou-se como um otimizador global devido sua eficiência e custo computacional relativamente baixo. Como o algoritmo PSO é uma técnica da computação evolutiva (obs.: alguns autores a definem como uma abordagem da inteligência coletiva), este trabalha a partir de uma população de partículas fazendo-os evoluir até que se ache uma solução aceitável. Entretanto, diferentemente dos algoritmos evolutivos clássicos, que utilizam uma população inicial de soluções (indivíduos) gerada aleatoriamente com uma distribuição uniforme, o algoritmo PSO trabalha com uma população de partículas em um enxame.

Sistemas imunológicos artificiais: O sistema imunológico é um mecanismo biológico capaz de reconhecer e eliminar elementos causadores de patologias. Os sistemas imunológicos artificiais fazem uso de fundamentos do sistema imunológico para desenvolver novas técnicas e algoritmos, incluindo-se métodos de seleção negativa, seleção clonal, medula óssea e rede imunológica [LVC06].

Redes neurais artificiais: Uma rede neural artificial pode ser definida como sendo uma estrutura de processamento composta por um número de unidades interconectadas (neurônios artificiais). Cada unidade apresenta um comportamento específico de entrada/saída (computação local), determinado pela sua função de transferência, pelas interconexões com outras unidades, dentro de um raio de vizinhança, e possivelmente pelas entradas externas capazes de aprender a partir do ambiente[LVC06].

Sistemas nebulosos: Os sistemas nebulosos, também conhecidos como sistemas de inferência nebulosa, ou sistemas nebulosos baseados em regras, ou modelos nebulosos, representam a mais importante ferramenta de modelagem baseada na teoria dos conjuntos nebulosos. Um sistema de inferência nebulosa é um mapeamento ou função de um espaço de alternativas de entrada para um espaço de saída. A estrutura básica de um sistema nebuloso clássico possui três componentes conceituais: uma base de regras, que contém o conjunto de regras nebulosas; uma base de dados, que define as funções de pertinência usadas nas regras

nebulosas; e um mecanismo de inferência, que realiza um procedimento de inferência (raciocínio nebuloso) para obter a saída ou conclusão, baseado em regras e fatos conhecidos.

Busca tabu: A busca tabu utiliza exploração reativa e memória flexível para guiar a busca no espaço de soluções. Através da exploração reativa, determina-se uma direção de busca baseada em propriedades da solução corrente e da história da busca. A memória flexível consiste de estruturas de memória de curto e longo prazo que armazenam a história da busca. A memória de curto prazo armazena atributos de soluções visitadas em passado recente. Estes atributos são armazenados numa lista tabu para impedir o retorno a soluções visitadas. A memória de longo prazo contém uma história seletiva de soluções e seus atributos encontrados durante o processo de busca, e é utilizada em estratégias de diversificação e intensificação da busca.

Busca dispersa (*scatter search*): O algoritmo de busca dispersa combina soluções de um conjunto referência para criar novas soluções. Este algoritmo consiste de uma abordagem de otimização que está estruturada em procedimentos de [MAR06]: (i) geração de diversificação, (ii) determinação do conjunto de referência, (iii) geração de subconjuntos e (iv) método de melhoria. Muitas vezes, esta abordagem é combinada com busca tabu e/ou algoritmo de reconexão de caminhos (*path relinking*).

Transgenética computacional: Os algoritmos transgenéticos definem um processo evolucionário desdobrado em três níveis, nos quais as informações são armazenadas e gerenciadas. A população de cromossomos está no primeiro nível e representa a memória corrente do processo de busca [RAM05]. O segundo nível é composto por uma população denominada de vetores transgenéticos, cuja natureza difere daquela dos cromossomos. Essa população é provida de ferramentas capazes de promover a diversificação e intensificação da busca no espaço de soluções. O terceiro nível é composto por regras que comandam a interação entre essas duas populações [GOL01].

GRASP: O método GRASP foi proposto por Feo e Resende [FEO95] e engloba um algoritmo iterativo onde cada iteração possui duas fases: uma da construção, na qual a solução viável é construída, seguida de uma fase de busca local que consiste em um procedimento de refinamento da solução gerada na fase de construção [MOT01].

VNS: A metaheurística VNS baseia-se em uma sistemática troca de estruturas de vizinhanças associada a um algoritmo de busca local. De forma diferenciada de outras metaheurísticas baseadas em métodos de busca local, o VNS segue uma trajetória, mas explora incrementalmente vizinhanças distantes da solução atual [MOT01].

3.3. Algoritmo de Enxame de Partículas (Algoritmo PSO)

O algoritmo PSO foi proposto, em 1995, por James Kennedy e por Russell Eberhart é baseado em comportamentos coletivos de pássaros [KEN95]. O algoritmo PSO constitui uma técnica da inteligência coletiva baseada em uma população de soluções e transições aleatórias [KEN99]. O algoritmo PSO apresenta características similares as técnicas da computação evolutiva, que são baseadas em uma população de soluções. Entretanto, a PSO é motivada pela simulação de comportamento social e cooperação entre agentes em vez da sobrevivência do indivíduo mais apto como nos algoritmos evolutivos. No algoritmo PSO, cada solução candidata (denominada partícula) possui associada uma velocidade. A velocidade é ajustada através de uma equação de atualização que considera a experiência da partícula correspondente e a experiência das outras partículas presentes na população.

O algoritmo PSO tem-se mostrado eficiente já é comparável no desempenho com os algoritmos tradicionais de otimização, tais como o algoritmo *simulated annealing* e os algoritmos genéticos [HUA05], [NGU06].

O conceito do algoritmo PSO consiste de, a cada passo iterativo, mudar a velocidade de cada partícula em direção as localizações do *pbest* (melhor posição) e do *gbest* (melhor partícula). A rapidez do procedimento de busca é ponderada através de um termo gerado de forma aleatória, sendo este vinculado de forma separada as localizações do *pbest* e do *gbest*. O procedimento para implementação do algoritmo PSO é regido pelas seguintes etapas [TEB06], [HER06], [COE06], [COE07]:

(i) iniciar uma população (matriz) de partículas, com posições e velocidades em um espaço de problema n dimensional, aleatoriamente com distribuição uniforme;

(ii) para cada partícula, avaliar a função de aptidão (função objetivo a ser minimizada, no contexto deste trabalho de otimização para PCV);

(iii) comparar a avaliação da função de aptidão da partícula com o *pbest* da partícula. Se o valor corrente é melhor que *pbest*, então o valor de *pbest* passa a ser igual ao valor da

função de aptidão da partícula, e a localização do $pbest$ passa a ser igual a localização atual no espaço n dimensional;

(iv) comparar a avaliação da função de aptidão com o prévio melhor valor de aptidão da população. Se o valor atual é melhor que o $gbest$, atualizar o valor de $gbest$ para o índice e valor da partícula atual;

(v) modificar a velocidade e a posição da partícula de acordo com as equações (3.1) e (3.2), respectivamente:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot ud \cdot [p_i(t) - x_i(t)] + c_2 \cdot Ud \cdot [p_g(t) - x_i(t)] \quad (3.1)$$

$$x_i(t+1) = x_i(t) + \Delta t \cdot v_i(t+1). \quad (3.2)$$

onde Δt é igual a 1.

(vi) ir para a etapa (ii) até que um critério de parada seja encontrado, usualmente um valor de erro pré-definido ou um número máximo de iterações (gerações).

As notações usadas são: t é a iteração (geração), $x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T$ armazena a posição da i -ésima partícula, $v_i = [v_{i1}, v_{i2}, \dots, v_{in}]^T$ armazena a velocidade da i -ésima partícula e $p_i = [p_{i1}, p_{i2}, \dots, p_{in}]^T$ representa a posição do melhor valor de aptidão da i -ésima partícula. O índice g representa o índice da melhor particular entre todas as partículas do grupo. A variável w é a ponderação de inércia, c_1 e c_2 são constantes positivas; ud e Ud são duas funções para geração de números aleatórios com distribuição uniforme no intervalo $[0,1]$, respectivamente. O tamanho da população é selecionado dependendo do problema.

As velocidades das partículas em cada dimensão são limitadas a um valor máximo de velocidade, V_{max} . O V_{max} é importante, pois determina a resolução que a região próxima às soluções atuais é procurada. Se V_{max} é alto, o algoritmo PSO facilita a busca global, enquanto um valor V_{max} pequeno enfatiza as buscas locais. A primeira parte na equação (3.1) é um termo de momento da partícula. A ponderação de inércia w representa o grau de momento da partícula. A segunda parte consiste da parte “cognitiva”, que representa o “conhecimento”

independente da partícula. A terceira parte é a “social”, que representa a colaboração entre as partículas.

As constantes c_1 e c_2 representam a ponderação das partes de “cognição” e “social”, estas influenciam cada partícula em direção a $pbest$ e a $gbest$. Estes parâmetros são usualmente ajustados por heurísticas de tentativa e erro.

3.3.1. QPSO (*Quantum Particle Swarm Optimization*)

O algoritmo QPSO permite às partículas a se moverem seguindo regras definidas na mecânica quântica ao invés de clássica movimentação aleatória Newtoniana [SUN04b].

No modelo quântico do PSO o estado de cada partícula é representado por uma função de onda $\Psi(\bar{x}, t)$ ao invés de posição e velocidade como no modelo convencional. O comportamento dinâmico da partícula é vastamente divergente do comportamento tradicional do PSO onde os valores exatos de velocidade e posição não podem ser determinados simultaneamente. Pode-se apenas aprender a probabilidade da partícula estar em uma determinada posição através da probabilidade da sua função densidade $|\Psi(\bar{x}, t)|^2$, a qual depende no campo potencial em que a partícula se encontra.

Em [SUN04a], [SUN04b], [SUN05], a função onda da partícula pode ser definida como:

$$\Psi(x) = \frac{1}{\sqrt{L}} \cdot e^{\frac{-\|p-x\|}{L}} \quad (3.3)$$

E a densidade de probabilidade é dada pela seguinte expressão:

$$Q(x) = |\Psi(x)|^2 = \frac{1}{L} \cdot e^{\frac{-2\|p-x\|}{L}} \quad (3.4)$$

O parâmetro L depende da intensidade da energia do campo potencial, o qual define o escopo de procura da partícula e pode ser denominado Criatividade ou Imaginação da partícula [SUN04b].

No modelo quântico inspirado do PSO, espaço de busca e o espaço de solução é diferente em termos de qualidade. A função onda ou a função probabilidade descreve o estado da partícula em um espaço de busca quântico, não provendo nenhuma informação sobre a posição da partícula, o que é mandatário para se calcular a função custo.

Neste contexto a transformação de estado entre estes dois espaços é essencial. Em termos de mecânica quântica a transformação de um estado quântico para um estado clássico, definido na mecânica convencional, é conhecida como colapso, que na natureza é a medida da posição da partícula. Podem-se evidenciar as diferenças entre o modelo convencional do PSO e o modelo quântico inspirado na Figura 3.3 [SUN04b].

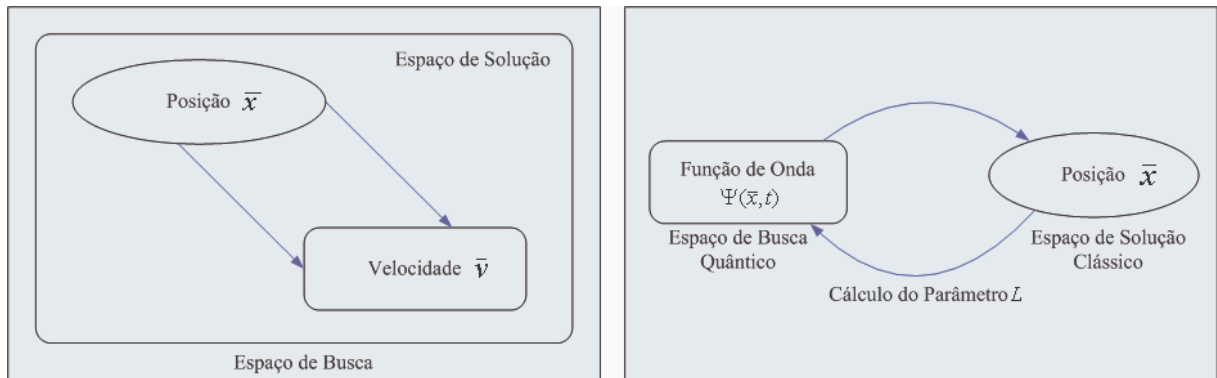


Figura 3.3. Espaço de Busca PSO e QPSO.

Devido à natureza quântica das equações as aferições utilizando computadores convencionais devem utilizar o Método de simulação estocástico de Monte Carlo (MMC).

A posição da partícula pode ser definida por:

$$x(t) = p \pm \frac{L}{2} \ln\left(\frac{1}{n}\right) \quad (3.5)$$

Em [SUN04b], o parâmetro L é calculado como sendo:

$$L(t+1) = 2 \cdot \alpha \cdot |p - x(t)| \quad (3.6)$$

Então a equação iterativa do QPSO é definida como:

$$x(t+1) = p \pm \alpha \cdot |p - x(t)| \cdot \ln\left(\frac{1}{u}\right), \quad u = rand(0,1) \quad (3.7)$$

que substitui equação (3.2) do modelo de algoritmo PSO convencional. Em termos de evolução do conhecimento de um organismo social, existem dois tipos de pensamentos relacionados à forma de como os indivíduos de uma população adquirem o conhecimento. O primeiro através do *pbest* o melhor valor encontrado pelo indivíduo e o *gbest* a melhor solução encontrada no enxame (população). Cada partícula procura na direção da posição atual do indivíduo para o ponto *p* que está situado entre o *pbest* e o *gbest*. O ponto *p* é conhecido como LIP (*Learning Inclination Point*) do indivíduo. A tendência de aprendizado de cada indivíduo faz com que cada um procure na vizinha do seu LIP, que é determinado pelo *pbest* e o *gbest*.

No algoritmo PSO quântico inspirado, a cada iteração, cada partícula grava seu *pbest* e compara com todas as outras partículas da população para obter o *gbest*. Para executar o próximo passo, o parâmetro *L* é calculado. Considera-se o parâmetro *L* como sendo Criatividade ou Imaginação da partícula, por isso se caracteriza como o escopo de busca de conhecimento da partícula. Quanto maior o valor de *L*, mais facilmente a partícula adquire novo conhecimento. No QPSO, o fator Criatividade da partícula é calculado como sendo a diferença entre a posição atual da partícula e o seu LIP como mostra a equação (3.6).

Em [SHI99] a posição da média melhor (*mbest*) é introduzida ao PSO e em [SUN05] utilizado no QPSO. O *mbest* é definido como sendo a média do *pbest* de todas as partículas do enxame (população), dada pela expressão:

$$mbest = \frac{1}{M} \sum_{i=1}^M p_i = \left(\frac{1}{M} \sum_{i=1}^M p_{i1}, \frac{1}{M} \sum_{i=1}^M p_{i2}, \dots, \frac{1}{M} \sum_{i=1}^M p_{id} \right) \quad (3.8)$$

onde *M* é o tamanho da população e *p_i* é o *pbest* da partícula *i*, sendo assim *L* pode ser redefinido como:

$$L(t+1) = 2 \cdot \alpha \cdot |mbest - x(t)| \quad (3.9)$$

e

$$x(t+1) = P \pm \alpha \cdot |mbest - x(t)| \cdot \ln\left(\frac{1}{u}\right), \quad u = rand(0,1) \quad (3.10)$$

O pseudocódigo do algoritmo QPSO é descrito a seguir:

População Inicial: Aleatória x_i

Faça

Para $i = 1$ até o tamanho da população M

Se $f(x_i) < f(p_i)$ então $p_i = x_i$

$p_g = \min(p_i)$

Calcula $mbest$ utilizando equação (3.8)

Para $d = 1$ até a dimensão D

$f\ddot{i}_1 = rand(0,1), f\ddot{i}_2 = rand(0,1)$

$$p = \frac{(f\ddot{i}_1 \cdot p_{id} + f\ddot{i}_2 \cdot p_{gd})}{(f\ddot{i}_1 + f\ddot{i}_2)}$$

$u = rand(0,1)$

Se $rand(0,1) > 0.5$

$$x_{id} = p - \beta \cdot abs(mbest_d - x_{id}) \cdot \ln\left(\frac{1}{u}\right)$$

Senão

$$x_{id} = p + \beta \cdot abs(mbest_d - x_{id}) \cdot \ln\left(\frac{1}{u}\right)$$

Até o critério de parada ser atingido

Algoritmo 3.1 Pseudocódigo QPSO.

3.3.1.1. Vantagens do QPSO

O modelo quântico do QPSO apresenta algumas vantagens em relação ao modelo tradicional. Podem-se citar algumas peculiaridades do QPSO de acordo com [SUN04b] que são as seguintes:

- Sistemas quânticos são sistemas complexos e não lineares que se baseiam no Princípio da Sobreposição de Estados, ou seja, os modelos quânticos têm muito mais estados que o modelo convencional;

- Sistemas quânticos são sistemas incertos, sendo assim bem diferente do sistema clássico estocástico. Antes da medição a partícula pode estar em qualquer estado com certa probabilidade, não existe uma trajetória definida;

3.3.1.2. Discretização

Até o momento o algoritmo de QPSO apresentado funciona para problemas contínuos e não problemas discretos como é o caso do PCV. Para que se possa utilizar o algoritmo proposto em problemas discretos algumas alterações devem ser implementadas.

Considera-se uma população inicial M de tamanho quatro e dimensão quatro representada pela matriz:

$$M = \begin{bmatrix} 1,02 & 3,56 & -0,16 & 4,5 \\ 1,14 & -0,10 & 5,27 & 1,65 \\ 1,63 & 1,52 & 0,48 & -1,24 \\ 1,99 & 0,97 & -1,82 & 2,24 \end{bmatrix}$$

onde cada linha da matriz M mencionada representa uma possível solução de um problema contínuo como, por exemplo, a minimização da função esfera $f(x) = \sum_{i=1}^n x_i^2$. Para se conhecer o $gbest$ desta população inicial basta calcular o $pbest$ de cada partícula (linha) e então verificar qual o menor. Ou seja, a partícula que possuir o menor $pbest$ possui também o $gbest$. Vale lembrar que dependendo da função custo associada ao problema a dimensão da população pode ser fixa ou, como no exemplo citado, pode ser variável. Pode se então aplicar o algoritmo QPSO proposto, neste caso.

Agora imagine-se o PCV neste contexto. Uma vez que a matriz M é gerada por uma distribuição uniforme, a pergunta é como pode-se calcular a função objetivo associada ao PCV, definido na equação (2.1), uma vez que as cidades (vértices do grafo) são números pertencentes ao conjunto dos inteiros positivos.

Como solução e proposta deste trabalho devem-se discretizar os valores contínuos da matriz M a fim de se poder calcular a função objetivo ou custo. Deve se salientar que a matriz M não é modificada e seus valores persistem na execução do algoritmo do QPSO. A

discretização é realizada apenas no momento de se calcular a função objetivo (equação (2.1)) para uma determinada solução. Sendo assim cada linha da matriz M , uma possível solução para problemas com quatro cidades, deve ser discretizada. Para tanto utiliza-se a seguinte regra: O menor valor da linha representará a primeira cidade o segundo menor valor a segunda cidade e assim sucessivamente. Este tipo de abordagem tem sido proposta na literatura em [TAS03]. Neste contexto, alguns trabalhos têm sido apresentados na literatura recente sobre a aplicação de abordagens de PSO para problemas combinatoriais, tais como [WAN03], [PAN04a], [PAN04b], [MAC05], [LOP05], [SOU06], mas nenhum deles usando QPSO nem FPGA.

Para a primeira linha da matriz M tem-se então:

$$[1,02 \quad 3,56 \quad -0,16 \quad 4,5] \Rightarrow [2 \quad 3 \quad 1 \quad 4]$$

onde $[2 \quad 3 \quad 1 \quad 4]$ representa uma solução para o problema do PCV de quatro cidades.

Pode-se observar a matriz M totalmente discretizada a seguir:

$$M = \begin{bmatrix} 1,02 & 3,56 & -0,16 & 4,5 \\ 1,14 & -0,10 & 5,27 & 1,65 \\ 1,63 & 1,52 & 0,48 & -1,24 \\ 1,99 & 0,97 & -1,82 & 2,24 \end{bmatrix} \Rightarrow M_{discreta} = \begin{bmatrix} 2 & 3 & 1 & 4 \\ 2 & 1 & 4 & 3 \\ 4 & 3 & 2 & 1 \\ 3 & 2 & 1 & 4 \end{bmatrix} \begin{matrix} tour(1) \\ tour(2) \\ \vdots \end{matrix}$$

Neste caso, existe um problema evidente derivado da utilização desta simples abordagem, que é evidenciado quando tem-se os valores repetidos na matriz M . Este fato se torna comum quando tem-se instâncias do PCV com um número de cidades elevadas. A duplicata de valores pode não afetar a execução para alguns problemas discretos, entretanto no caso do PCV não pode-se conseguir vértices repetidos na solução. Neste contexto, deve-se lembrar da definição do PCV:

*“Dado um número N de cidades e os custos (distância) de viajar de uma cidade a outra, qual é a rota mais barata em que se visita cada cidade **exatamente uma vez** e retorna então à cidade de origem?”*.

Para solucionar este problema utilizasse então o seguinte algoritmo representado pelo pseudocódigo a seguir no Algoritmo 3.2.

Para $i = 1$ até tamanho da população M

LinhaCont = *Linha_M*[i]

LinhaOrd = *Qsort*(*LinhaCont*)

Para $j = 1$ até o tamanho da *LinhaOrd*

LinhaTemp[j] = j ;

Fim_Para

Marca = *Zeros*(*Tamanho*(*LinhaOrd*))

LinhaDisc = *Zeros*(*Tamanho*(*LinhaOrd*))

Para $k = 1$ até o tamanho da *LinhaOrd*

Se *LinhaCont*[k] = *LinhaOrd*[k] então:

Se $(k+1) < \text{Tamanho}(\text{LinhaOrd})$ e *LinhaOrd*[k] = *LinhaOrd*[$k+1$]
então

Incrementa *Marca*[k]

Fim_Se

LinhaDisc[k] = $k + \text{Marca}[k]$

Fim_Se

Fim_Para

Linha_Mdiscreta[i] = *LinhaDisc*

Fim_Para

Algoritmo 3.2 Discretização da Matriz de População.

Capítulo 4.

4 Plataforma de Desenvolvimento

Ao se projetar uma solução embarcada devem se considerar as exigências de projeto que exigem processar as várias entradas para produzir as saídas esperadas.

Existem várias opções aceitáveis quando se trata de processadores. A primeira e mais utilizada solução é a de um microprocessador discreto. Os microprocessadores discretos são de uso geral e produzidos em grande escala. Facilmente encontrados no mercado e vastamente utilizado na indústria. Dentro dos mais utilizado tem-se também os processadores *hard core*. O processador *hard core* é um processador customizado em bolachas de silício projetado exclusivamente para um determinado projeto podendo, em alguns casos ser, implementado em uma FPGA.

Uma solução que vem se tornando um crescente ultimamente é o uso de processadores *soft core*. Um processador *soft core* é uma solução que pode ser implementada utilizando apenas portas lógicas de uma FPGA. Devido a esta implementação os processadores *soft core* não apresentam o mesmo desempenho de processadores *hard-core* ou discretos. Entretanto em muitos sistemas embarcados o alto desempenho que pode ser obtido pelas soluções mencionadas não é mandatário e pode ser substituída por flexibilidade e expansão de funcionalidades.

Processadores *soft core* são apropriados para sistemas simples, onde existem apenas funcionalidades de manipulação de entradas e saídas (GPIO, *General Purpose Input Output*), porém também são apropriados para sistemas complexos, onde um sistema operacional é incorporado diversas interfaces podem ser utilizadas.

Existem várias razões para escolher um processador *soft core* ao invés de processadores *hard core* ou discretos. Uma delas, talvez a principal, que pode ser citada é que

este tipo de processador pode ser reconfigurado de acordo com as necessidades e devido à própria natureza reconfigurável da FPGA.

4.1. Hardware

A seguir os detalhes de implementação usando placa Celoxica RC200 aliada ao processador *soft core* MicroBlaze, são apresentados.

4.1.1. Placa Celoxica RC200

A placa da Celoxica RC200 é um modelo completo em relação ao número de periféricos e interfaces de IO disponíveis. A placa possui interfaces paralela, serial (RS232), *ethernet* e *bluetooth* além de um barramento ATA (*Advanced Technology Attachment*) que pode ser utilizado como GPIO. A Figura 4.1 mostra uma visão do *hardware* utilizado.

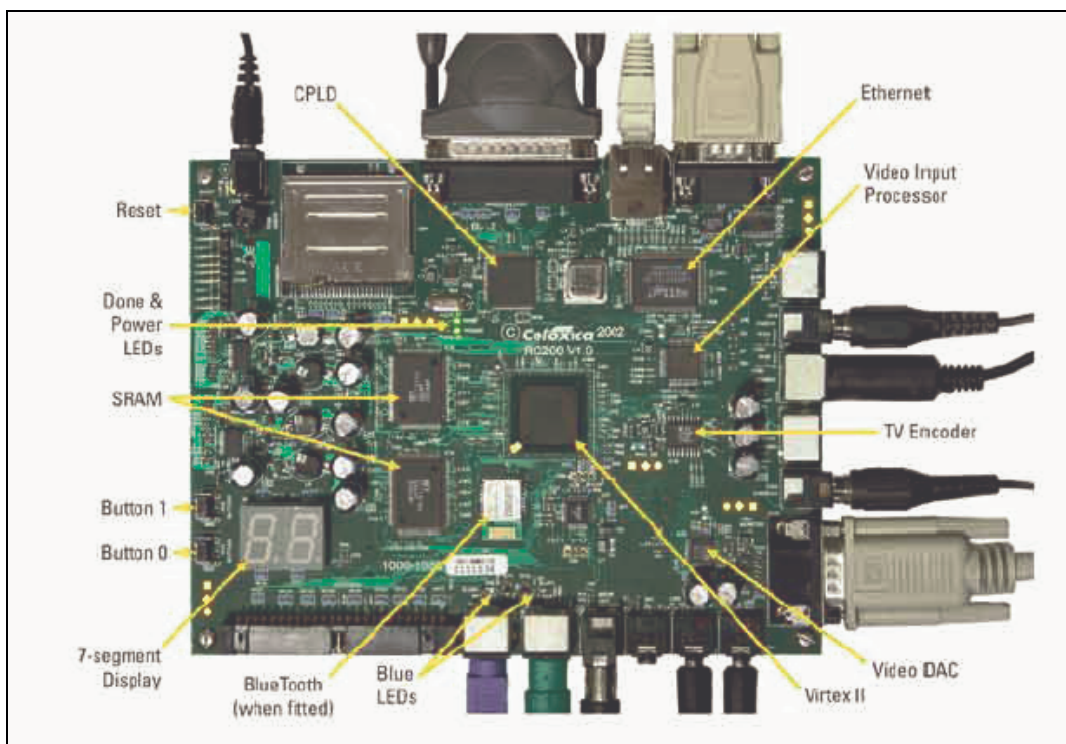


Figura 4.1. Placa Celoxica RC200.

A placa também possui dois blocos de memória SRAM de 8Mb além de possuir CPLD para armazenamento de programas não voláteis, uma vez que a FPGA é volátil, e também uma leitora de *Smart Flash*. A seguir a Figura 4.2 mostra o *hardware* na forma de blocos lógicos.

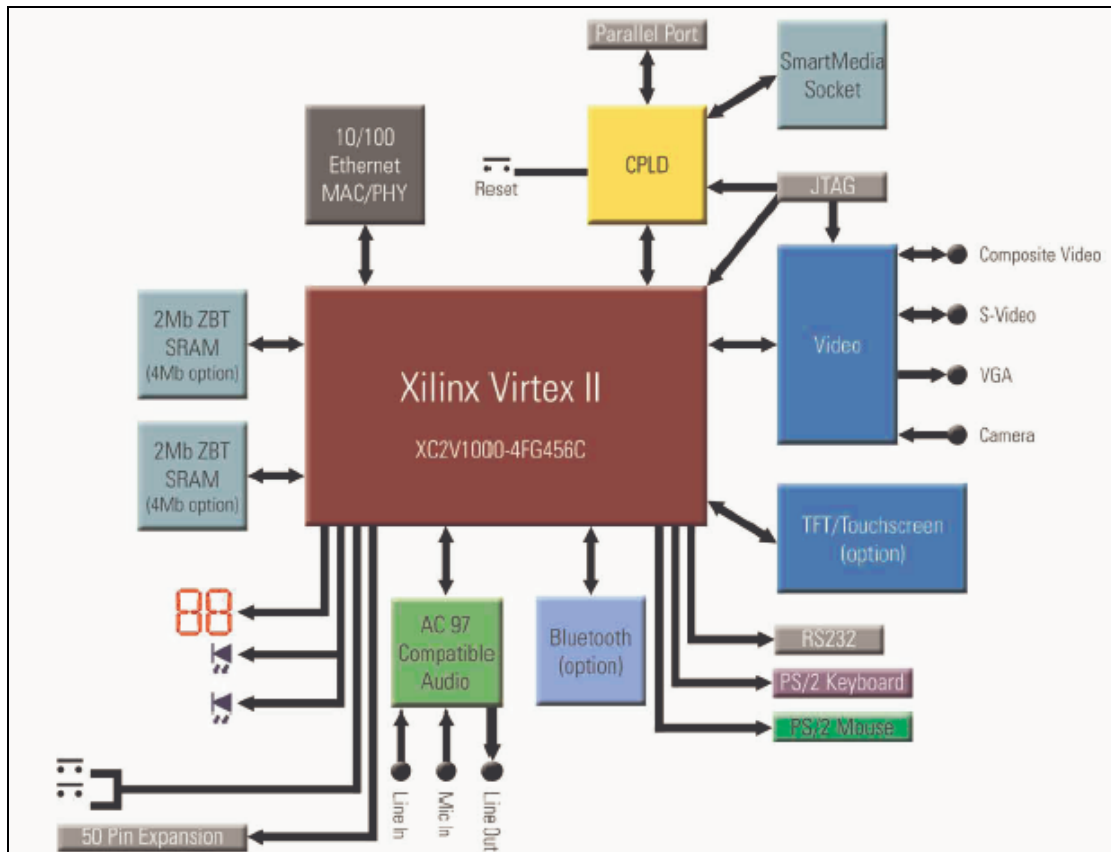


Figura 4.2. Placa Celoxica RC200 (Blocos Lógicos).

4.1.2. MicroBlaze

O MicroBlaze é um processador *soft core* fornecido pela Xilinx otimizado para uso de FPGAs do mesmo fabricante. É baseado em uma arquitetura RISC e tem a palavra, tanto para dados como para instruções definidas em 32bits. Dependendo da FPGA onde for programado o MicroBlaze pode atingir velocidade de até 210MHz, entretanto neste trabalho devido às limitações da placa a frequência de *clock* utilizada foi de 50MHz.

O processador pode ser conectado a um barramento OPB podendo assim acessar um vasto leque de módulos diferente chamados de IPs. Os módulos IPs são análogos a *device drivers* quando falamos de sistemas operacionais. Cada IP é responsável por prover uma

Em projetos de missão crítica co-processadores podem ser facilmente adicionados ao projeto do MicroBlaze via interface FSL.

No escopo deste trabalho apenas alguns periféricos foram mapeados para serem utilizados na resolução do problema do PCV.

São eles:

- Bancos de Memória SRAM 8Mb via OPB;
- UART (Porta Serial RS232) via OPB;
- Modulo de Debug via FSL;
- BRAM via LMB.

A Figura 4.4 a seguir mostra este mapeamento utilizando o *software* de desenvolvimento da Xilinx, *Xilinx Platform Studio* v.8.2. A Figura 4.5 mostra o endereçamento de *hardware* utilizado neste trabalho.

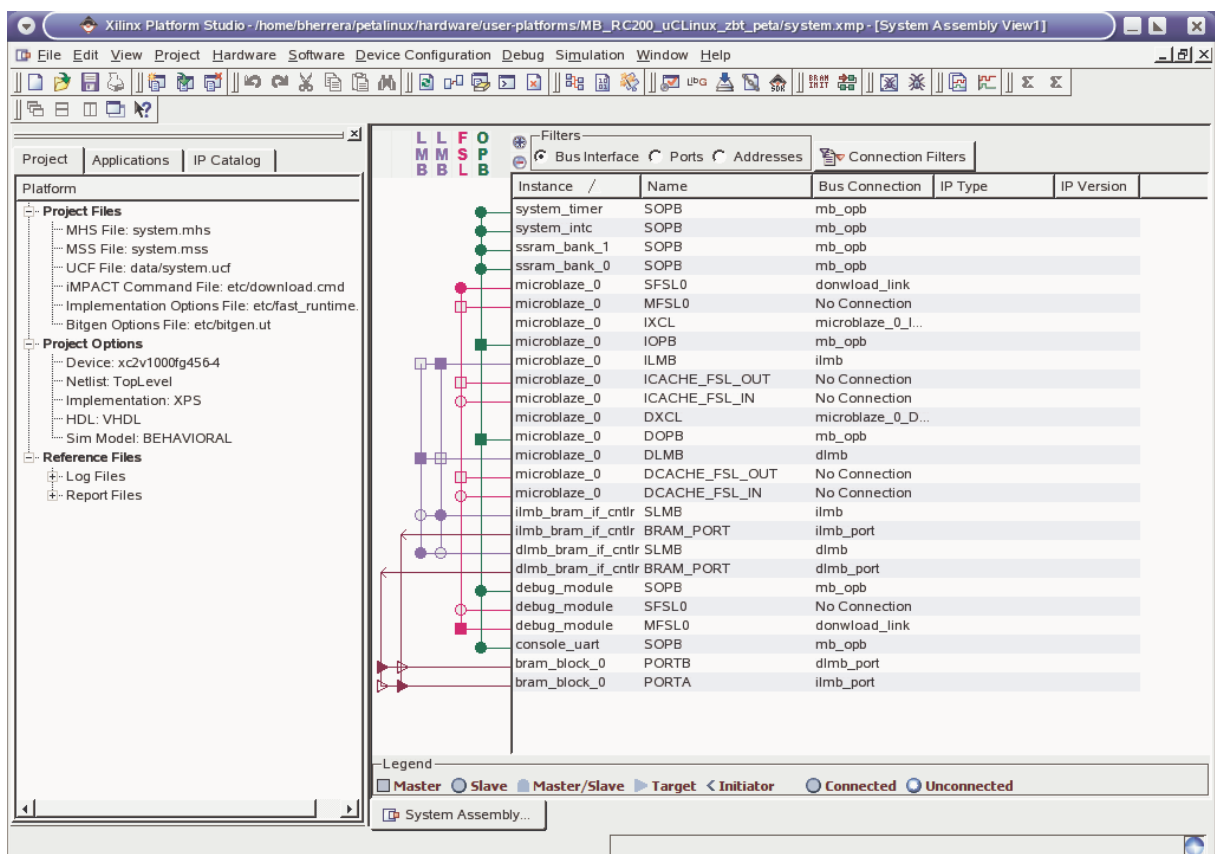


Figura 4.4. Barramento de periféricos *Microblaze*

Mesmo não possuindo MMU e não sendo capaz de suportar sistemas operacionais de grande porte, por exemplo, Windows ou Linux, existem vários *ports* de sistemas embarcados para o este processador. Dentre eles podemos citar o porte de um RTOS o *FreeRTOS* e o escolhido para ser utilizado neste trabalho o *uCLinux*.

A Xilinx inclui como parte do kit de desenvolvimento embarcado um conjunto de ferramentas GNU como o compilador GNU C(*mb-gcc*) e do depurador GDB (*mb-gdb*) dentre outras ferramentas para simulação e teste da FPGA.

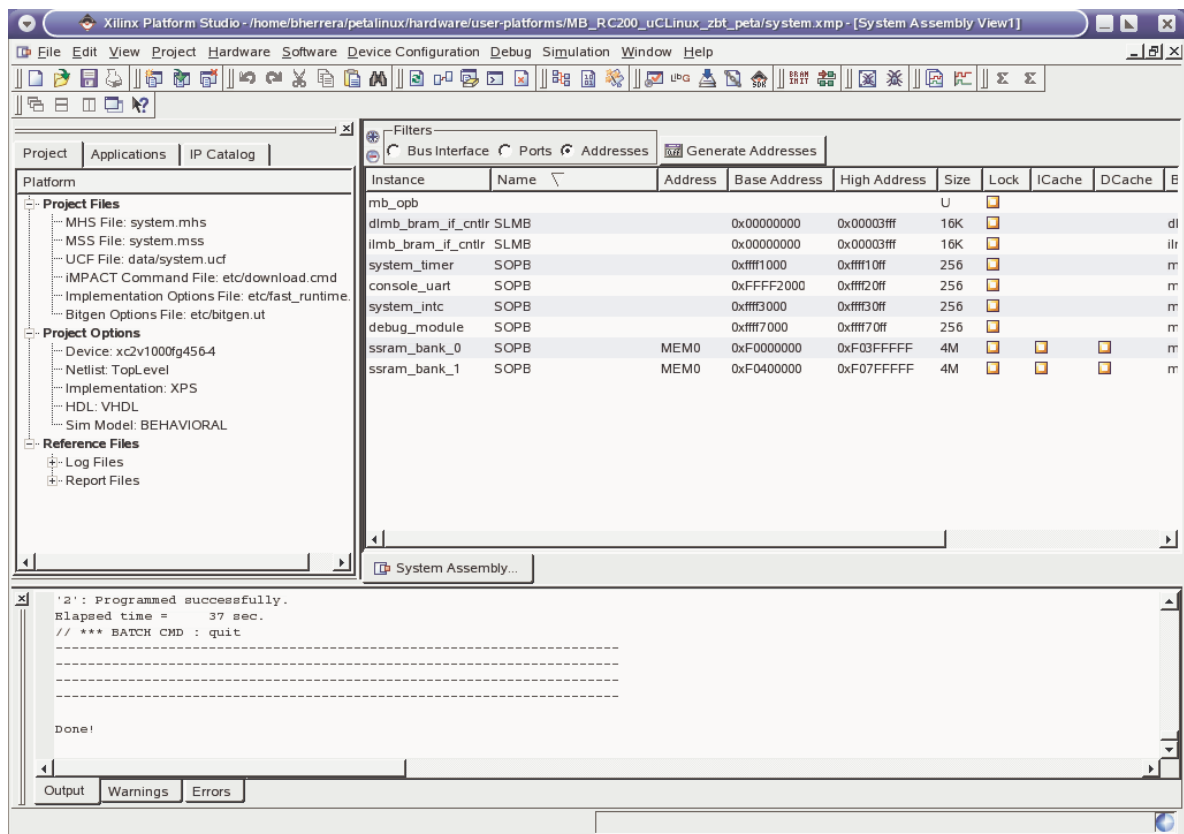


Figura 4.5. Endereçamento de periféricos *Microblaze*.

4.2. Software

A seguir os detalhes da compilação (porte) do *kernel* do *uCLinux* para placa Celoxica RC200 utilizando o processador *soft core* *MicroBlaze* são detalhados.

A versão original do *uCLinux* foi derivado do *kernel* do Linux 2.0, inicialmente desenvolvido para micro controladores sem unidades de gerenciamento de memória (*MMUs* – *Memory Magement Units*). Entretanto, os projetos embarcados utilizando sistema operacional Linux e micro-controladores vêm aumentando consideravelmente nos últimos anos, tanto pelo

reconhecimento da comunidade científica como por se caracterizar como um sistema robusto e confiável. Existem hoje diversos portes do sistema para as mais diversas arquiteturas de *hardware*.

Atualmente uCLinux como sistema operacional possibilita a escolha em três versões diferentes do kernel do Linux (2.0, 2.4 e 2.6) bem como uma vasta coleção de aplicações, bibliotecas e ferramenta de desenvolvimento.

Apesar do fato de já existir um porte do uCLinux para o *soft core* utilizado neste trabalho (MicroBlaze), algumas customizações se fazem necessárias uma vez que o endereçamento de periféricos de hardware varia de placa para placa. Como contribuição deste trabalho foi realizado o porte uCLinux para a placa RC200 da Celoxica, se caracterizando como o único existente até a presente data, que seja de conhecimento do autor. É possível ver a tela de *boot/login* do sistema na Figura 4.6.

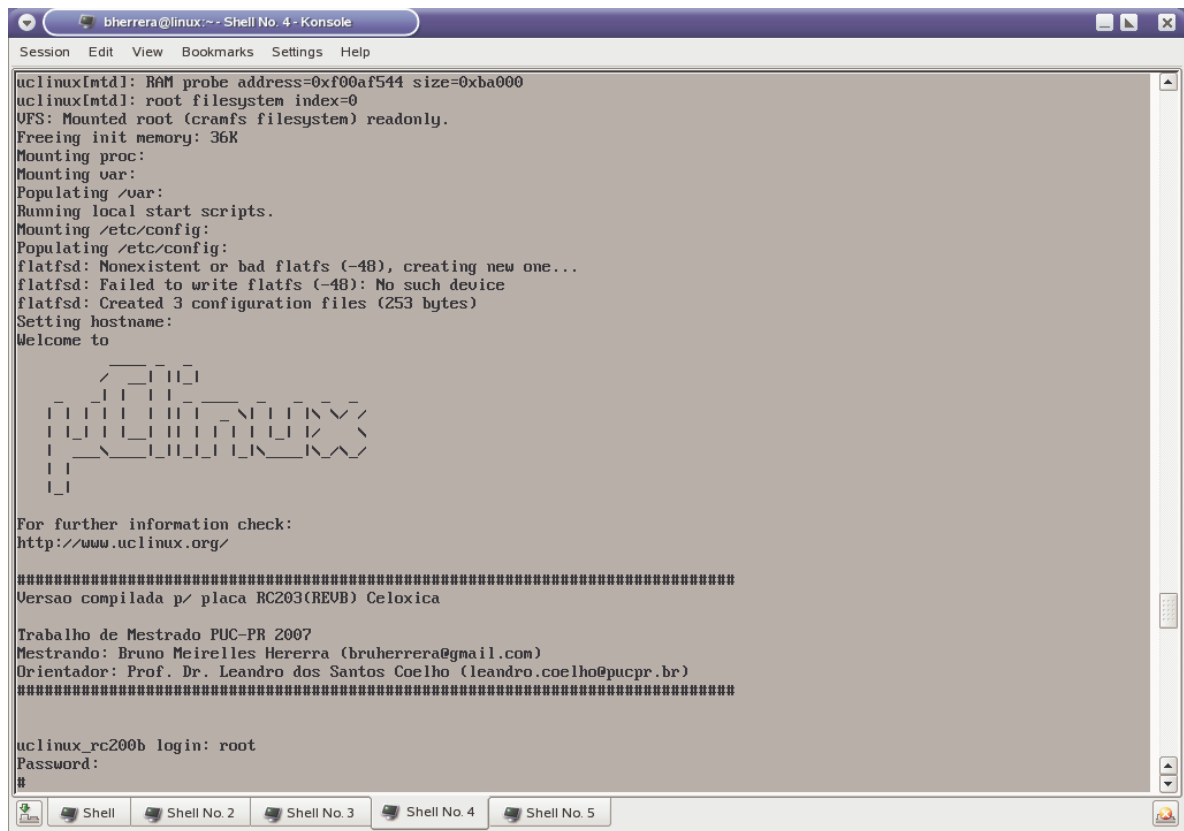


Figura 4.6. Boot uCLinux compilado para placa RC200.

Capítulo 5.

5 Implementação Computacional e Análise de Resultados

Neste capítulo será apresentada uma análise estatística dos resultados obtidos através dos experimentos realizados utilizando a heurística de melhoria LKH e a metaheurística QPSO previamente fundamentadas nos capítulos 3 e 4. Os resultados foram obtidos através de diversas execuções dos algoritmos tanto em um PC (processador de 2.8GHz com 512MB de RAM (*Random Access Memory*) quanto na plataforma embarcada RC200 (50MHz com 8MB de RAM).

5.1. Forma de Execução

Os algoritmos executados foram aplicados a instâncias do repositório TSPLIB. O tamanho das instâncias foi selecionado de forma a termos instâncias de tamanhos variados, mas seguindo um padrão. Instâncias pequenas, médias e grandes foram selecionadas para testar as abordagens de otimização selecionadas, para o PCV.

Para cada instância do problema o algoritmo foi executado 30 vezes utilizando sementes diferentes em cada uma delas. A forma de execução adotada foi a seguinte:

- (i) QPSO gera uma população inicial aleatória;
- (ii) O melhor indivíduo (*tour*) desta população inicial gravado em arquivo com extensão .ini (Utilizado a posteriori pelo Alea+LKH);
- (iii) QPSO executa até encontrar o melhor global para população gerada aleatoriamente;
- (iv) O melhor global (*tour*) é gravado em um arquivo com extensão .final;
- (v) Executa-se o LKH passando como *tour* inicial o arquivo .ini (Abordagem Alea + LK);
- (vi) Executa-se o LKH passando como *tour* inicial o arquivo .final (Abordagem QPSO + LK);
- (vii) Executa-se o LKH com parâmetros *default*(LKH puro).

O mesmo procedimento foi adotado tanto no PC(*Personal Computer*) quanto na plataforma embarcada. No PC com sistema operacional Windows XP Professional, os algoritmos de otimização foram compilados utilizando o compilador C/C++ MinGW. Para a plataforma embarcada, os códigos foram compilados utilizando Suse Linux 10.1 com um compilador GCC para MicroBlaze (mb-gcc).

Os tempos mostrados nas Tabelas 5-1 e 5-2 a seguir foi calculado com base na média das 30 execuções. O tamanho da população foi fixo em 100 partículas e o critério de parada era o valor ótimo mencionado na TSPLIB para o problema em questão. No caso do QPSO sem LKH o critério de parada era número fixo de iteração previamente definido como sendo de 1000.

5.2. Resultados para o PCV Simétrico

Na Tabela 5-1 são apresentados os resultados para os algoritmos: (i) QPSO+LKH, (ii) Aleat + LKH e (iii) LKH, para 10 problemas teste da TSPLIB [REI94]. Nesta tabela usou-se a notação % para representar quantos % o valor obtido está distante do valor ótimo para o problema teste.

Nota-se pelos resultados da Tabela 5-1 que para o problema swiss42 [REI94], os algoritmos QPSO+LKH, Alea+LKH e LKH obtiveram desempenho similar em termos de análise estatística com a obtenção do melhor valor (valor ótimo) para a função objetivo de 1273.

Em relação ao problema Gr229, o algoritmo Ale+LKH não conseguiu obter o valor ótimo de 134602, ficando 0,010% deste valor. No entanto, o QPSO+LKH e o LKH obtiveram o valor ótimo. Mas deve-se ressaltar que em média o LKH foi levemente superior ao QPSO+LKH.

Para o problema pcb442 observa-se que este cenário o QPSO+LKH foi o algoritmo mais rápido. No entanto, as três abordagens de otimização obtiveram o valor ótimo para a função objetivo que é de 50778.

Baseado nos resultados de simulação para o problema Gr666 apresentados na Tabela 5-1 nota-se que o menor desvio padrão foi obtido pelo QPSO+LKH, mas a média dos resultados da função objetivo para o QPSO+LKH e LKH foi idêntica.

Quanto ao problema dsj1000, todos os algoritmos testados obtiveram o ótimo. No entanto, em termos de convergência, o LKH apresentou a melhor média dos resultados para a função objetivo.

Em relação ao problema pr1002, os algoritmos de otimização obtiveram o valor ótimo para o PCV, mas o Alea+LKH foi o algoritmo mais rápido. No caso do pcb1173, o Alea+LKH foi a abordagem de otimização com melhor média de valor obtidos para a função objetivo.

Para os problemas d1291 e u1817, o LKH foi o método com melhor média, mas lembrando que o QPSO+LKH obteve o valor ótimo em pelo menos um dos 30 experimentos realizados. Nota-se, também, que o Alea+LKH obtém o ótimo para o d1291, mas o melhor resultado dele fica a 0,042% do ótimo para o problema u1817.

Para o PCV simétrico de maior porte testado nesta dissertação nota-se que o QPSO+LKH foi o método com melhor e mais rápido que o Alea+LKH e o LKH.

Nas Figura 5.1 a 5.9 são apresentados os resultados em termos da convergência para os resultados da Tabela 5-1 . A partir destas figuras, nota-se que o QPSO+LKH foi mais rápido que o Alea+LKH, mas inferior quanto a convergência quando comparado ao LKH clássico.

Na Figura 5.10 é mostrado a melhor rota encontrada para o problema teste pcb1173.

Tabela 5-1. Resultados de Otimização das Instâncias para o PCV Simétrico.

Instância (Ótimo)	Método	Mínimo / (%)		Média / (%)		Máximo / (%)		Mediana / (%)		Desvio Padrão	Tempo(s)
swiss42 (1.273)	QPSO+LKH	1.273	0,000	1.273,000	0,000	1.273	0,000	1.273,0	0,000	0,00	≈ 0
	Alea+LKH	1.273	0,000	1.273,000	0,000	1.273	0,000	1.273,0	0,000	0,00	≈ 0
	LKH	1.273	0,000	1.273,000	0,000	1.273	0,000	1.273,0	0,000	0,00	≈ 0
Gr229 (134.602)	QPSO+LKH	134.602	0,000	134.615,533	0,010	134.616	0,010	134.616,0	0,010	2,56	1,8
	Alea+LKH	134.616	0,010	134.616,000	0,010	134.616	0,010	134.616,0	0,010	0,00	1,7
	LKH	134.602	0,000	134.613,200	0,008	134.616	0,010	134.616,0	0,010	5,70	0,1
pcb442 (50.778)	QPSO+LKH	50.778	0,000	50.778,233	0,000	50.785	0,014	50.778,0	0,000	1,28	0,3
	Alea+LKH	50.778	0,000	50.778,233	0,000	50.785	0,014	50.778,0	0,000	1,28	2,5
	LKH	50.778	0,000	50.778,233	0,000	50.785	0,014	50.778,0	0,000	1,28	0,8
Gr666 (294.358)	QPSO+LKH	294.358	0,000	294.444,667	0,029	294.476	0,040	294.476,0	0,040	52,63	15,9
	Alea+LKH	294.358	0,000	294.426,833	0,023	294.476	0,040	294.476,0	0,040	60,76	11,8
	LKH	294.358	0,000	294.426,833	0,023	294.476	0,040	294.476,0	0,040	60,76	6,6
dsj1000 (18.659.688)	QPSO+LKH	18.660.188	0,003	18.664.570,200	0,026	18.681.851	0,119	18.660.188,0	0,003	8.944,22	34
	Alea+LKH	18.660.188	0,003	18.666.030,933	0,034	18.682.099	0,120	18.660.188,0	0,003	9.914,04	25,26
	LKH	18.660.188	0,003	18.664.537,133	0,026	18.681.851	0,119	18.660.188,0	0,003	8.961,00	28,16
Pr1002 (259.045)	QPSO+LKH	259.045	0,000	259.045,000	0,000	259.045	0,000	259.045,0	0,000	0,00	2,6
	Alea+LKH	259.045	0,000	259.045,000	0,000	259.045	0,000	259.045,0	0,000	0,00	1,1
	LKH	259.045	0,000	259.045,000	0,000	259.045	0,000	259.045,0	0,000	0,00	2,8
pcb1173 (56.892)	QPSO+LKH	56.892	0,000	56.893,000	0,002	56.897	0,009	56.892,0	0,000	2,07	8,3
	Alea+LKH	56.892	0,000	56.892,333	0,001	56.897	0,009	56.892,0	0,000	1,29	1,3
	LKH	56.892	0,000	56.893,000	0,002	56.897	0,009	56.892,0	0,000	2,07	9,2
d1291 (50.801)	QPSO+LKH	50.801	0,000	50.849,750	0,096	50.886	0,167	50.868,5	0,133	42,07	40,4
	Alea+LKH	50.801	0,000	50.843,500	0,084	50.886	0,167	50.843,5	0,084	45,43	40,2
	LKH	50.801	0,000	50.830,100	0,057	50.886	0,167	50.801,0	0,000	39,60	79,8
u1817 (50.201)	QPSO+LKH	57.201	0,000	57.242,833	0,073	57.313	0,196	57.243,0	0,073	37,24	75,5
	Alea+LKH	57.225	0,042	57.246,250	0,079	57.272	0,124	57.245,5	0,078	16,06	93,1
	LKH	57.201	0,000	57.237,083	0,063	57.272	0,124	57.236,5	0,062	17,33	117
Fl3795 (28.772)	QPSO+LKH	28.772	0,000	28.779,231	0,025	28.813	0,143	28.772,0	0,000	11,83	1310,33
	Alea+LKH	28.772	0,000	28.788,692	0,058	28.813	0,143	28.785,0	0,045	17,75	1787
	LKH	28.772	0,000	28.808,846	0,128	28.881	0,379	28.813,0	0,143	27,93	2367,3

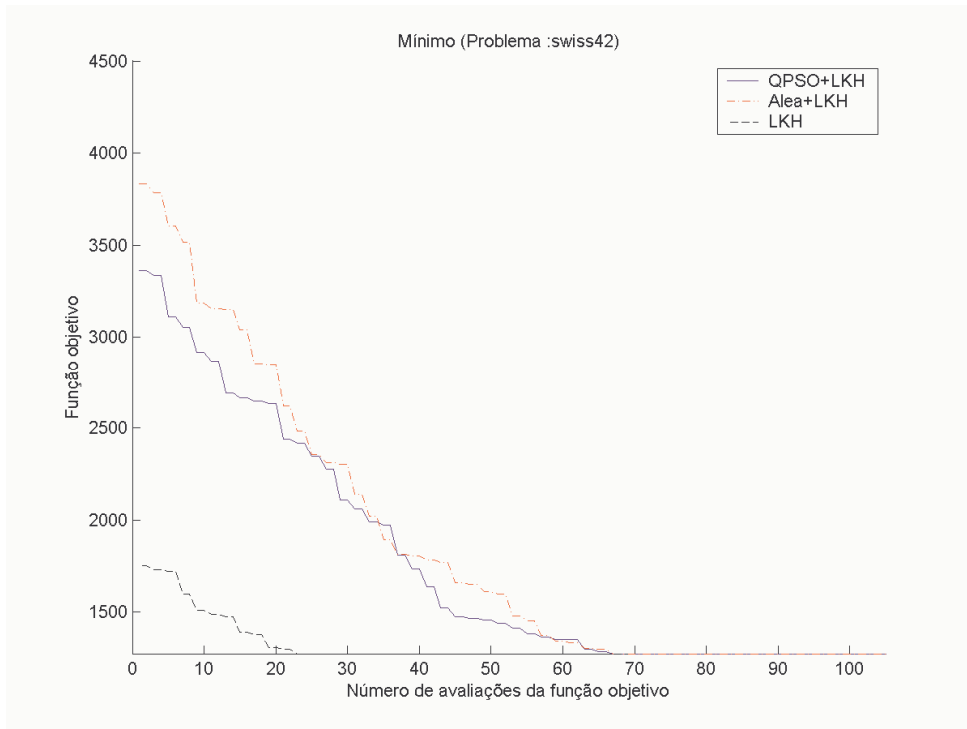


Figura 5.1. Instância swiss42 (Mínimo).

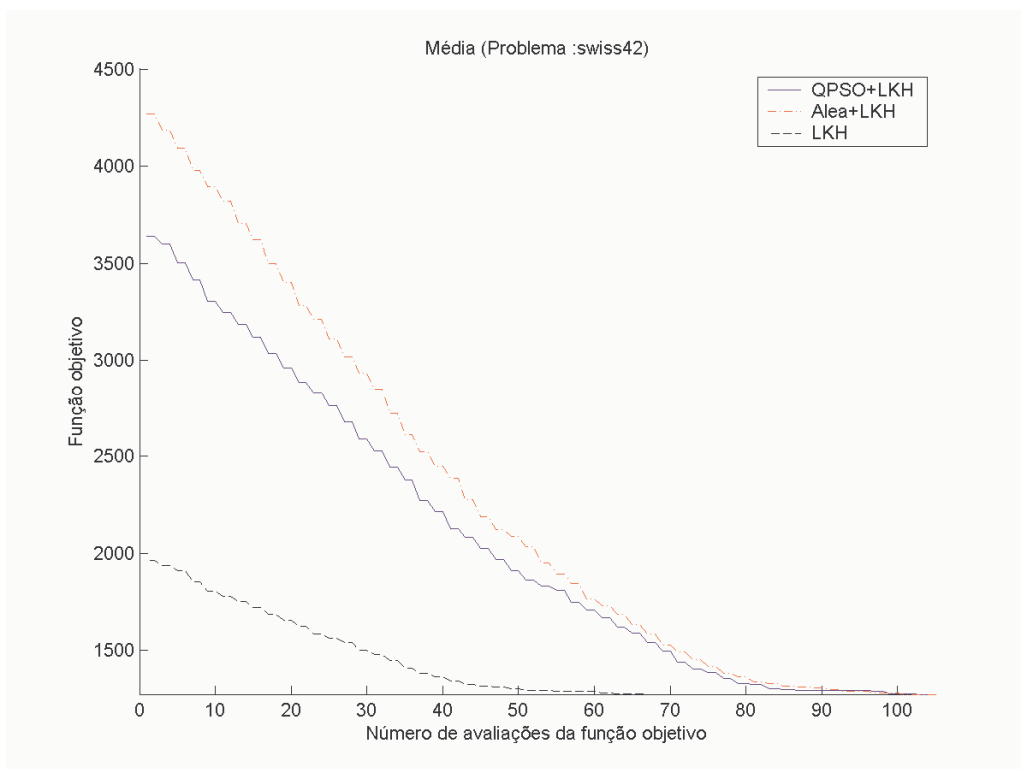


Figura 5.2. Instância swiss42 (Média).

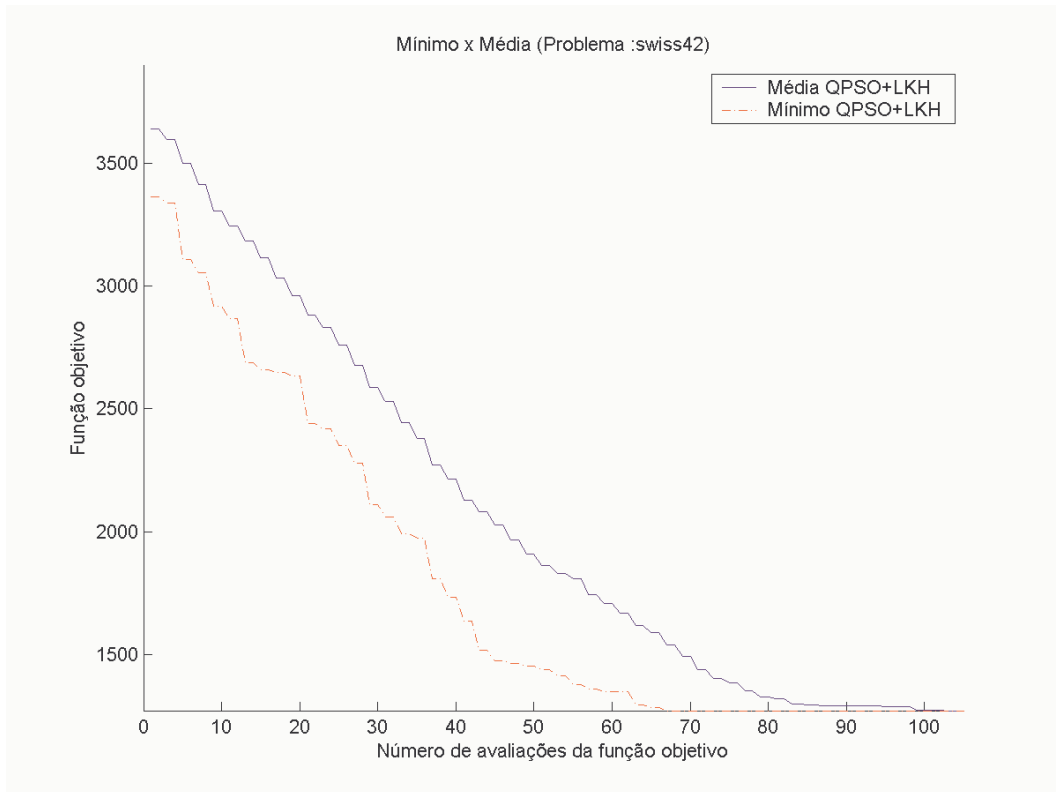


Figura 5.3. Instância swiss42 (Mínimo e Média).

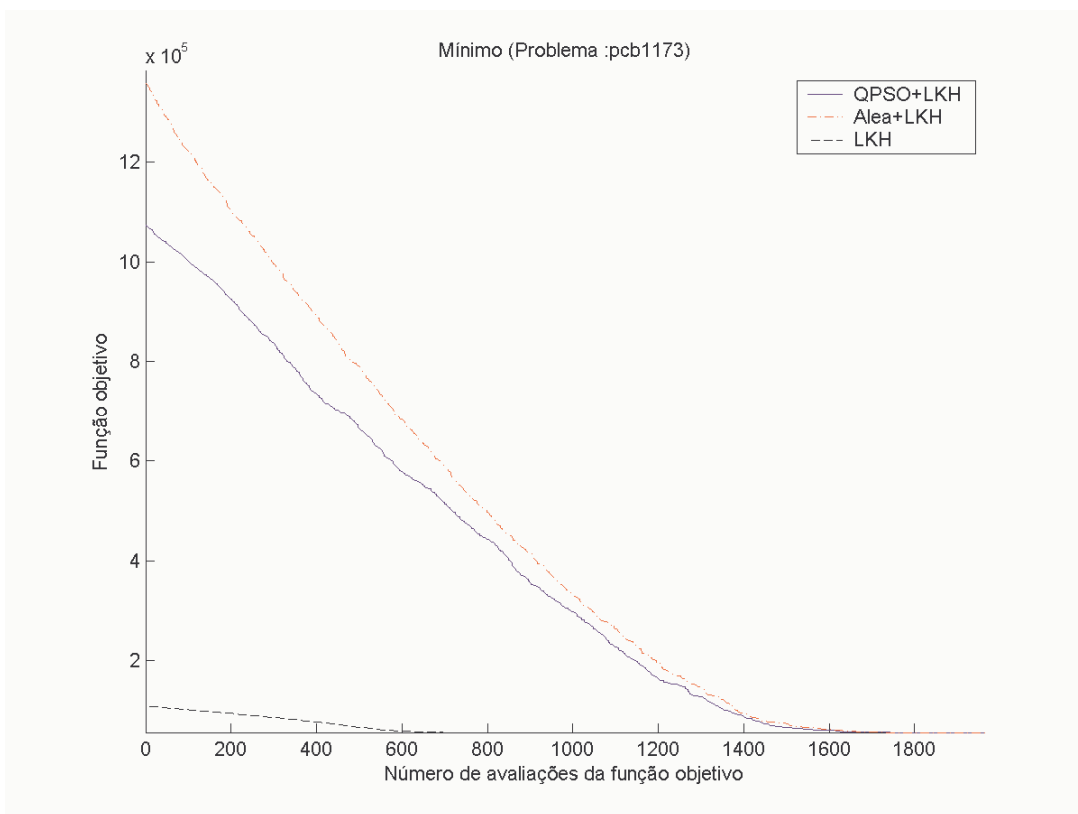


Figura 5.4. Instância pcb1173 (Mínimo).

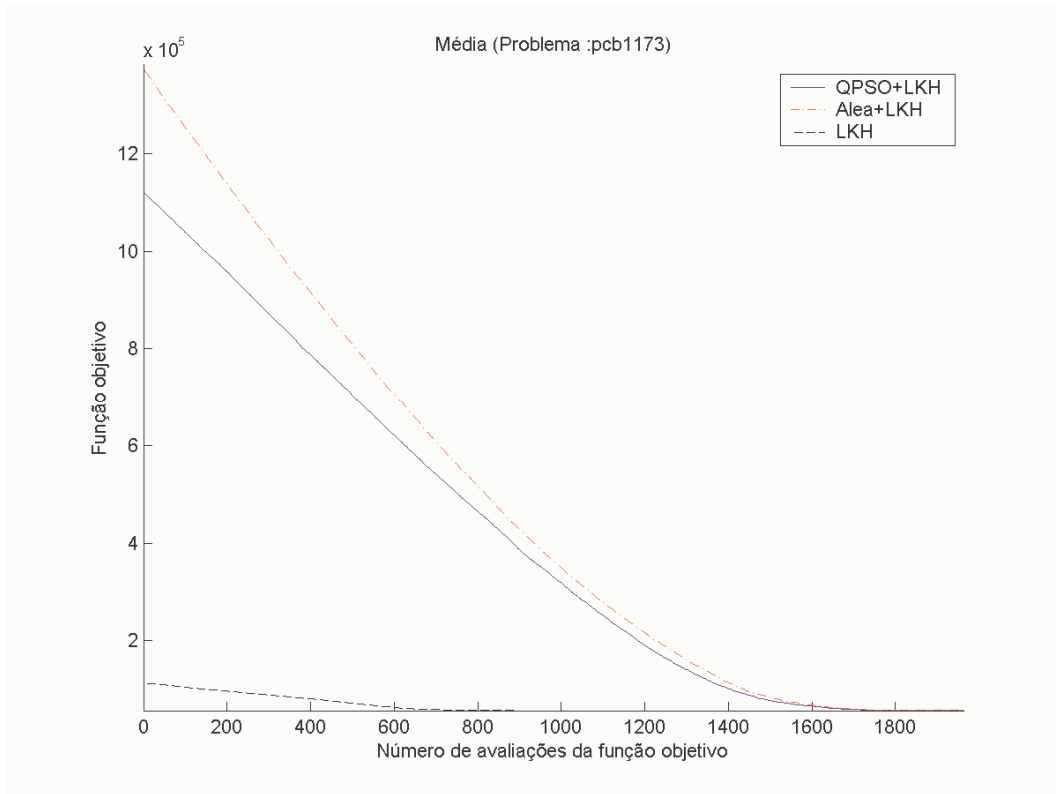


Figura 5.5. Instância pcb1173 (Média).

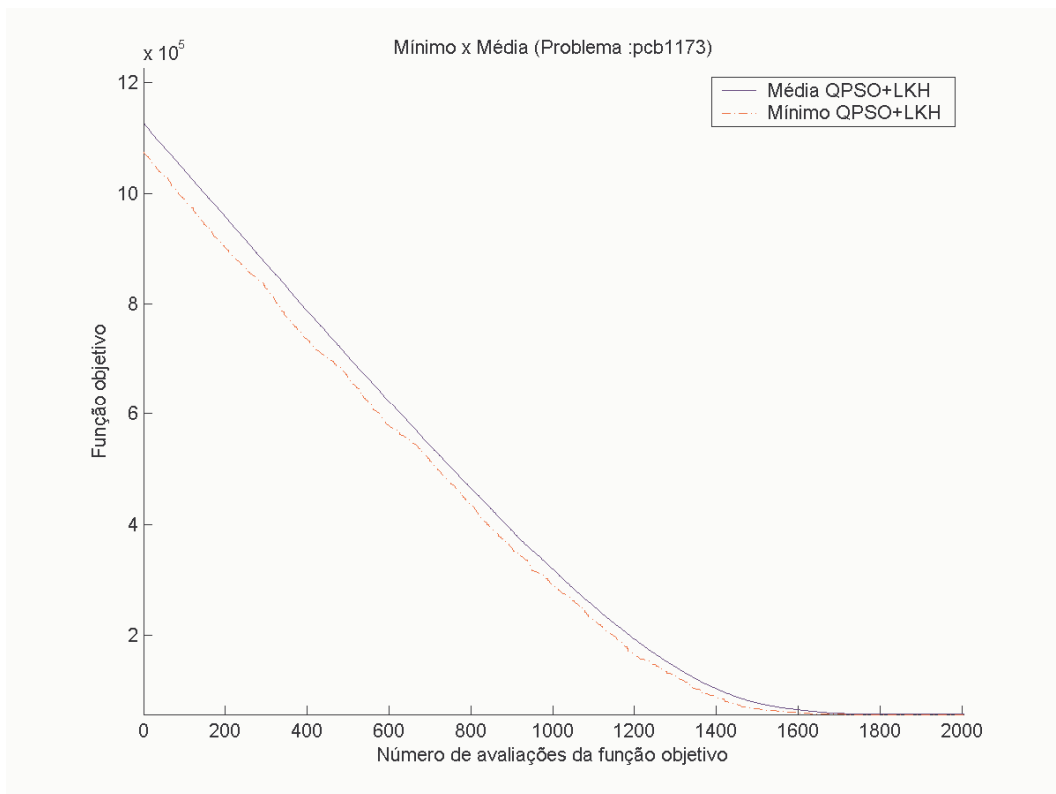


Figura 5.6. Instância pcb1173 (Mínimo e Média).

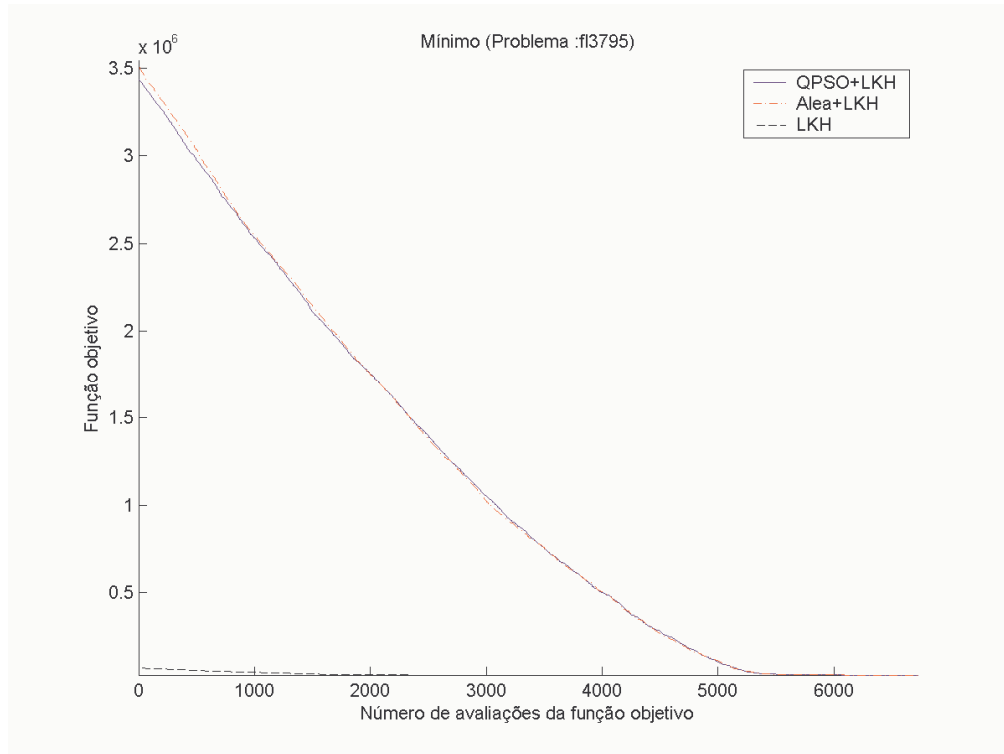


Figura 5.7. Instância fl3795 (Mínimo).

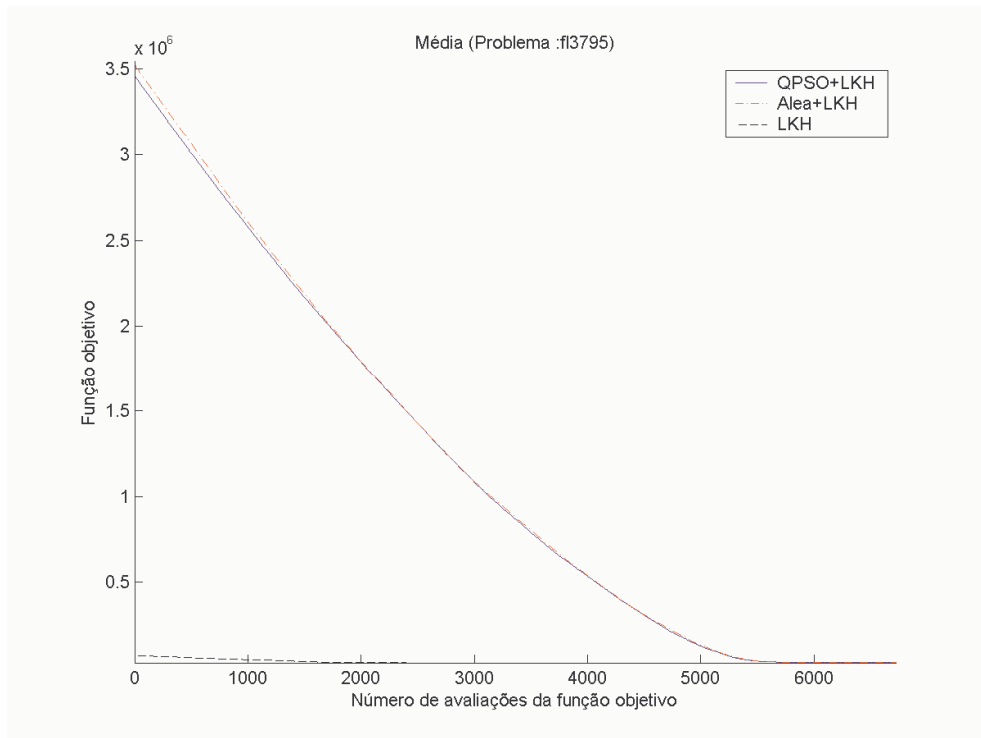


Figura 5.8. Instância fl3795 (Média).

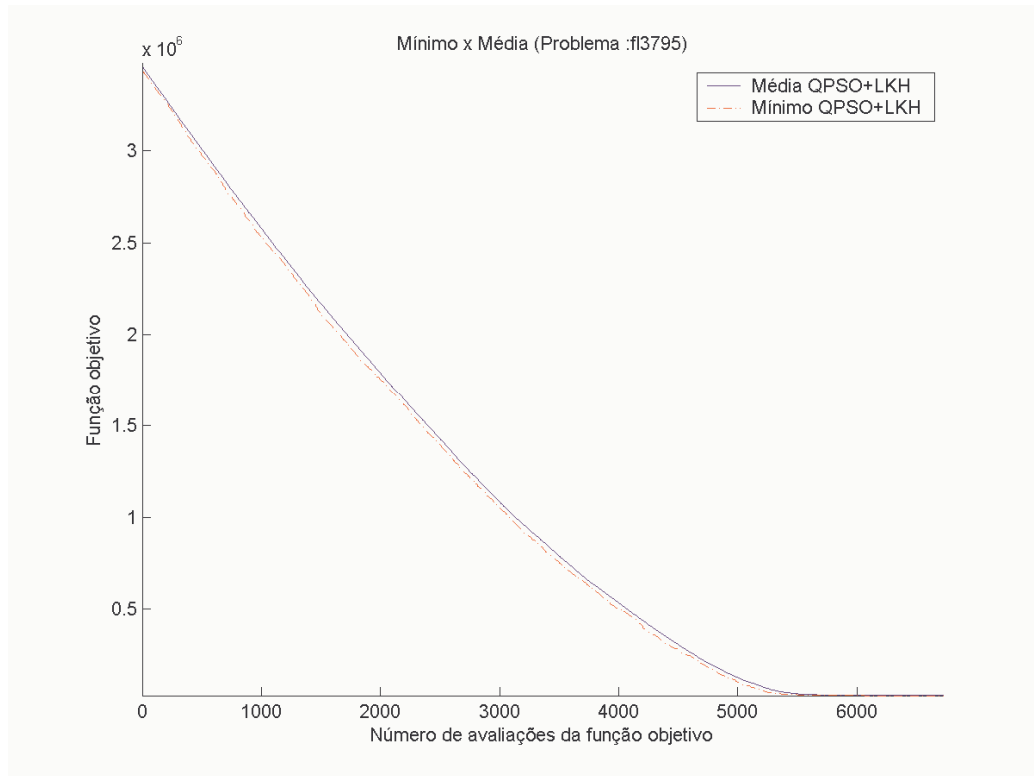


Figura 5.9. Instância fl3795 (Mínimo e Média).

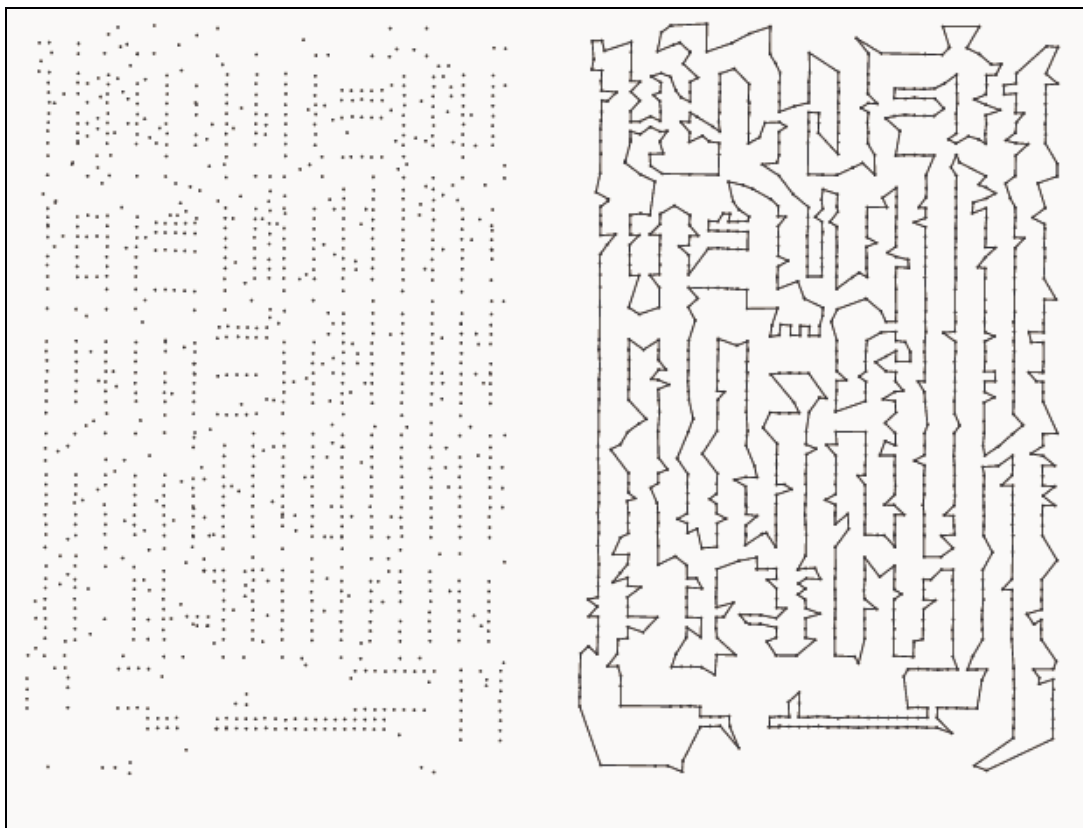


Figura 5.10. Exemplo da melhor rota encontrada para o problema pcb1173.

5.3. Problemas Assimétricos

Na Tabela 5-2 são apresentados os resultados para os algoritmos: (i) QPSO+LKH, (ii) Aleat + LKH e (iii) LKH, para 4 problemas teste de PCV assimétrico da TSPLIB. Da mesma forma que na Tabela 5-1, na Tabela 5-2 usou-se a notação % para representar quantos % o valor obtido está distante do valor ótimo para o problema teste.

Nota-se pelos resultados da Tabela 5-2 que para os problemas ftv38 e rg443, todos os algoritmos obtiveram o valor ótimo, mas o LKH foi o mais rápido. No caso do ftv323, o Alea+LKH foi o algoritmo mais lento dos algoritmos testados.

Tabela 5-2. Resultados de Otimização das Instâncias para o PCV Assimétrico.

Instância (Ótimo)	Método	Mínimo / (%)		Média / (%)		Máximo / (%)		Mediana / (%)		Desvio. Padrão	Tempo(s)
ftv38 (1.530)	QPSO+LKH	1.530	0,00	1.532,00	0,13	1.530,53	0,03	1.530,00	0,00	0,90	0,1
	Alea+LKH	1.530	0,00	1.532,00	0,13	1.530,33	0,02	1.530,00	0,00	0,76	0,1
	LKH	1.530	0,00	1.532,00	0,13	1.530,20	0,01	1.530,00	0,00	0,61	≈ 0
ftv170 (2.755)	QPSO+LKH	2.755	0,00	2.755,00	0,00	2.755,00	0,00	2.755,00	0,00	0,00	0,1
	Alea+LKH	2.755	0,00	2.755,00	0,00	2.755,00	0,00	2.755,00	0,00	0,00	0,4
	LKH	2.755	0,00	2.755,00	0,00	2.755,00	0,00	2.755,00	0,00	0,00	0,1
rg323 (1.326)	QPSO+LKH	1.326	0,00	1.327,73	0,13	1.328,00	0,15	1.328,00	0,15	0,70	29,1
	Alea+LKH	1.326	0,00	1.327,33	0,10	1.328,00	0,15	1.328,00	0,15	0,98	24,9
	LKH	1.326	0,00	1.327,07	0,08	1.328,00	0,15	1.328,00	0,15	1,03	20,2
rg443 (2.720)	QPSO+LKH	2.720	0,00	2.720,00	0,00	2.720,00	0,00	2.720,00	0,00	0,00	163
	Alea+LKH	2.720	0,00	2.720,00	0,00	2.720,00	0,00	2.720,00	0,00	0,00	164
	LKH	2.720	0,00	2.720,00	0,00	2.720,00	0,00	2.720,00	0,00	0,00	55

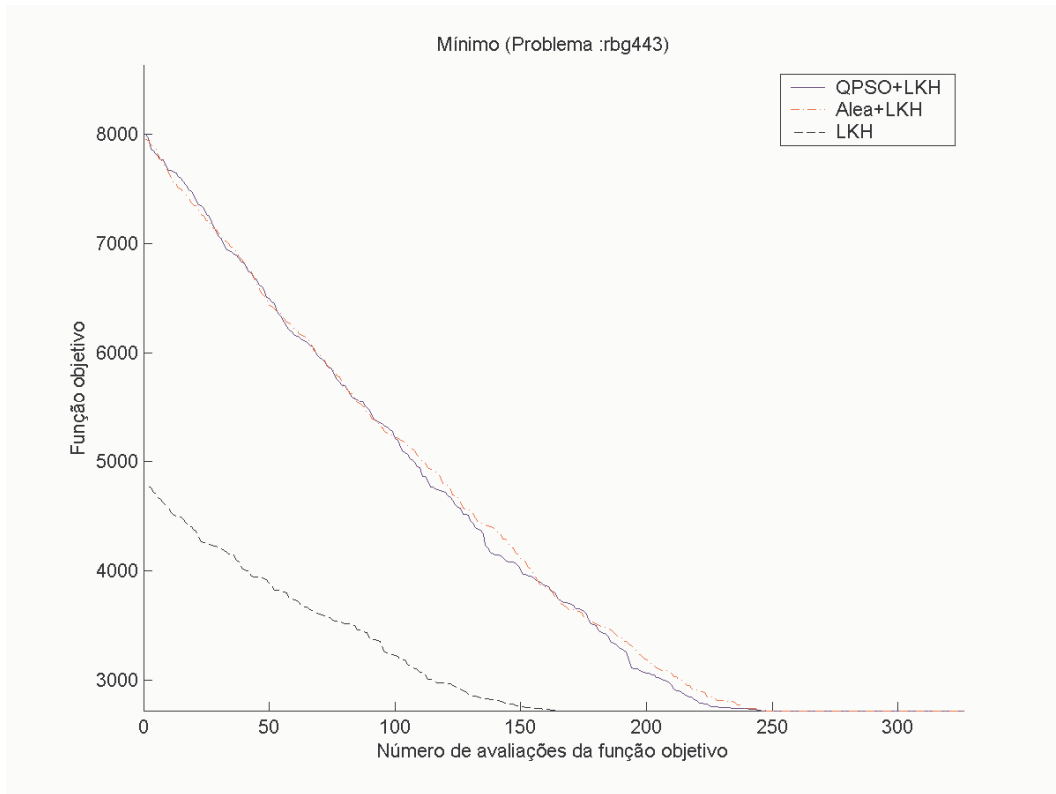


Figura 5.11. Instância rbg443 (Mínimo).

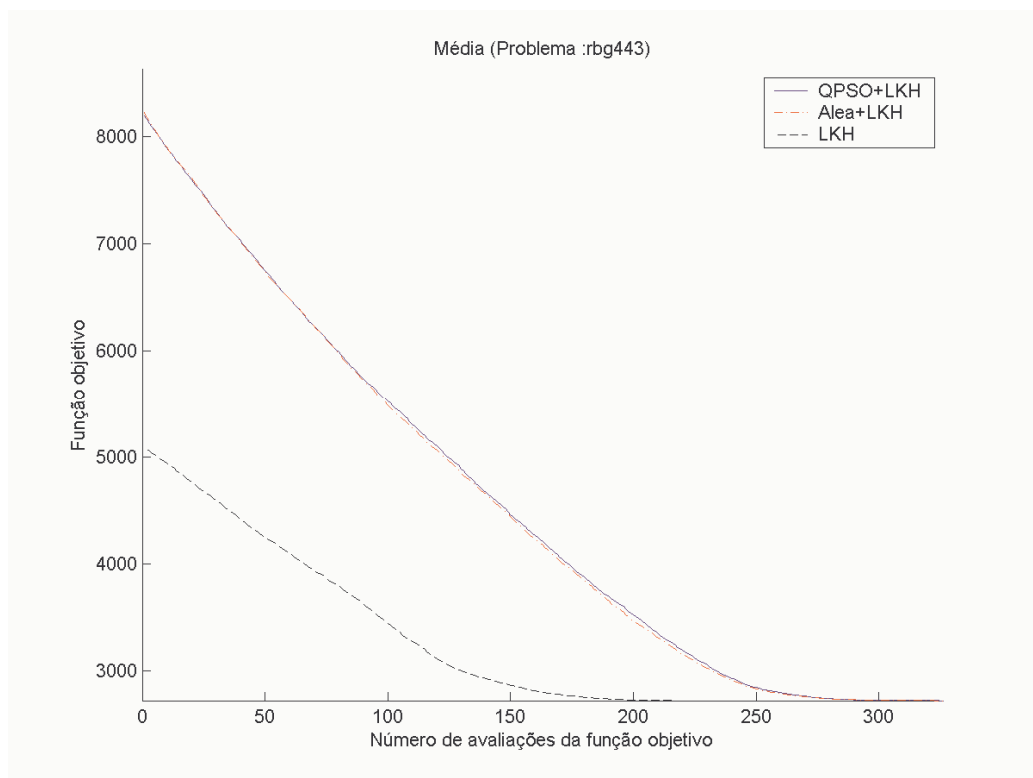


Figura 5.12. Instância rbg443 (Média).

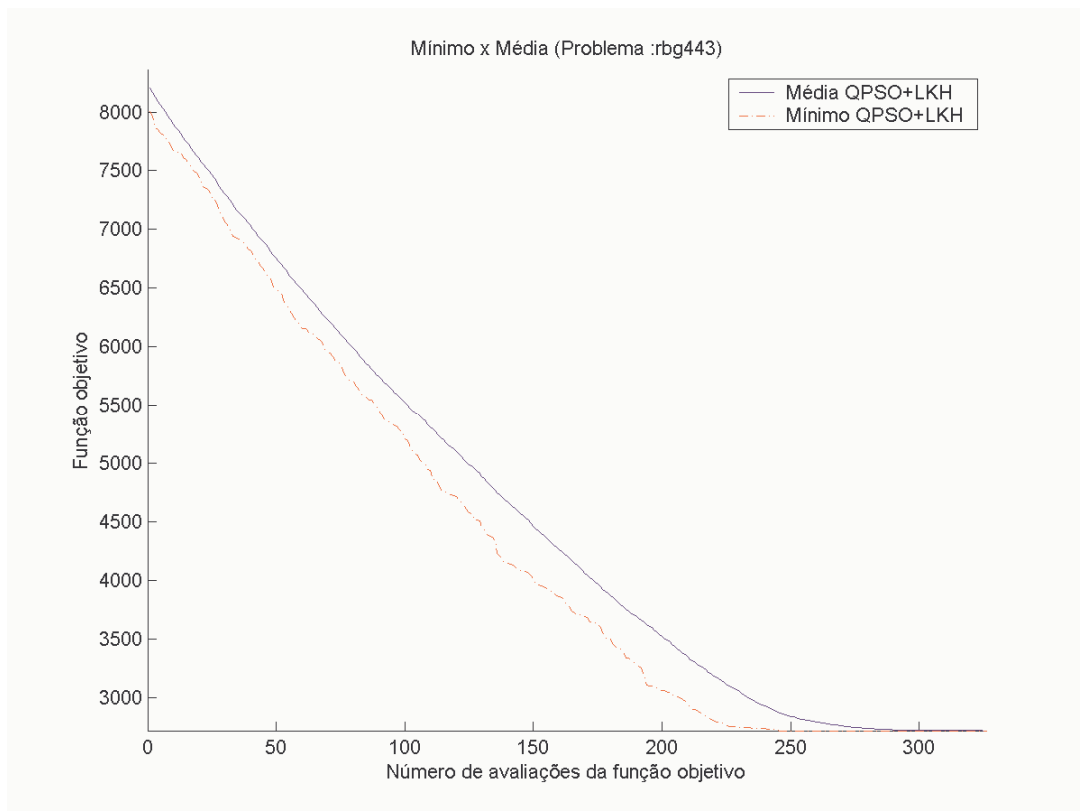


Figura 5.13. Instância rbg443 (Mínimo e Média).

5.4. Execuções Embarcadas na FPGA

Devido a limitações de memória RAM, não é possível executar testes com instâncias com muitos vértices. No entanto foram escolhidas duas instâncias do TSPLIB para execuções em plataforma embarcada. Uma instancia simétrica contendo 42 vértices (swiss42) e uma instancia assimétrica contendo 170 vértices (ftv170). Na Tabela 5-3 são apresentados os resultados das instâncias para o PCV embarcado na FPGA.

Tabela 5-3. Resultados das Instâncias em Execuções Embarcadas.

Instância (Ótimo)	Método	Mínimo / (%)	Média / (%)	Máximo / (%)	Mediana / (%)	Desvio. Padrão	Tempo(s)
swiss42 (1.273)	QPSO+LKH	1.273 0	1.273 0	1.273 0	1.273 0	0	2,1
	Alea+LKH	1.273 0	1.273 0	1.273 0	1.273 0	0	2,4
	LKH	1.273 0	1.273 0	1.273 0	1.273 0	0	1,8
ftv170 (2.755)	QPSO+LKH	2.755 0	2.755 0	2.755 0	2.755 0	0	43,2
	Alea+LKH	2.755 0	2.755 0	2.755 0	2.755 0	0	45,8
	LKH	2.755 0	2.755 0	2.755 0	2.755 0	0	43

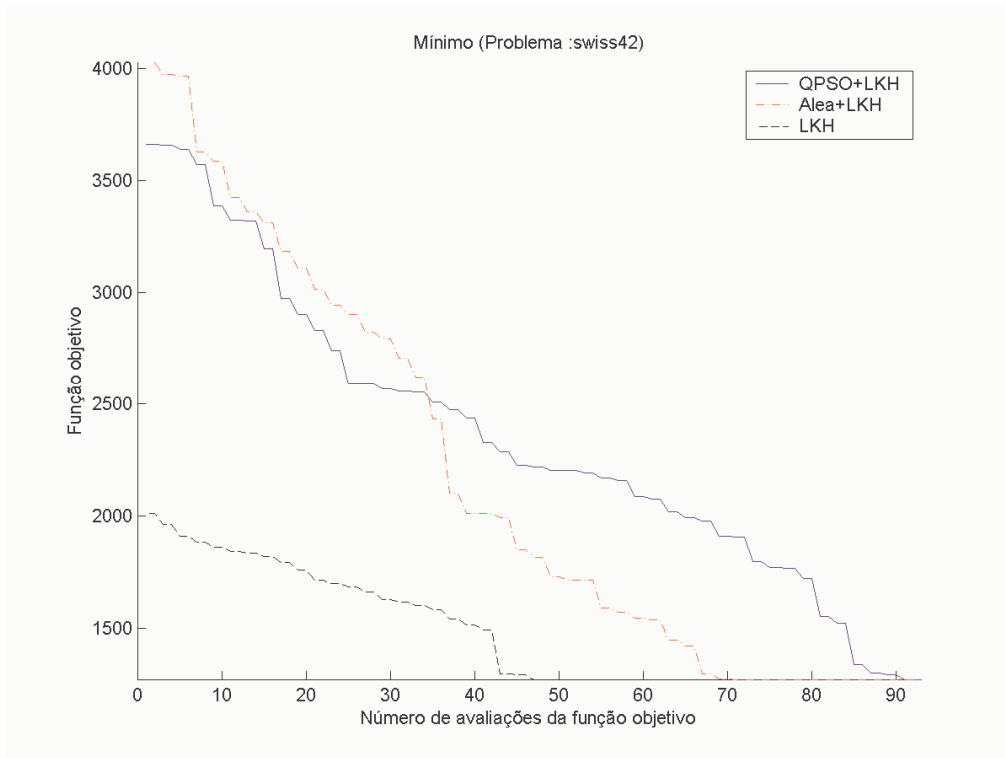


Figura 5.14. Mínimo obtido para instância swiss42 em FPGA.

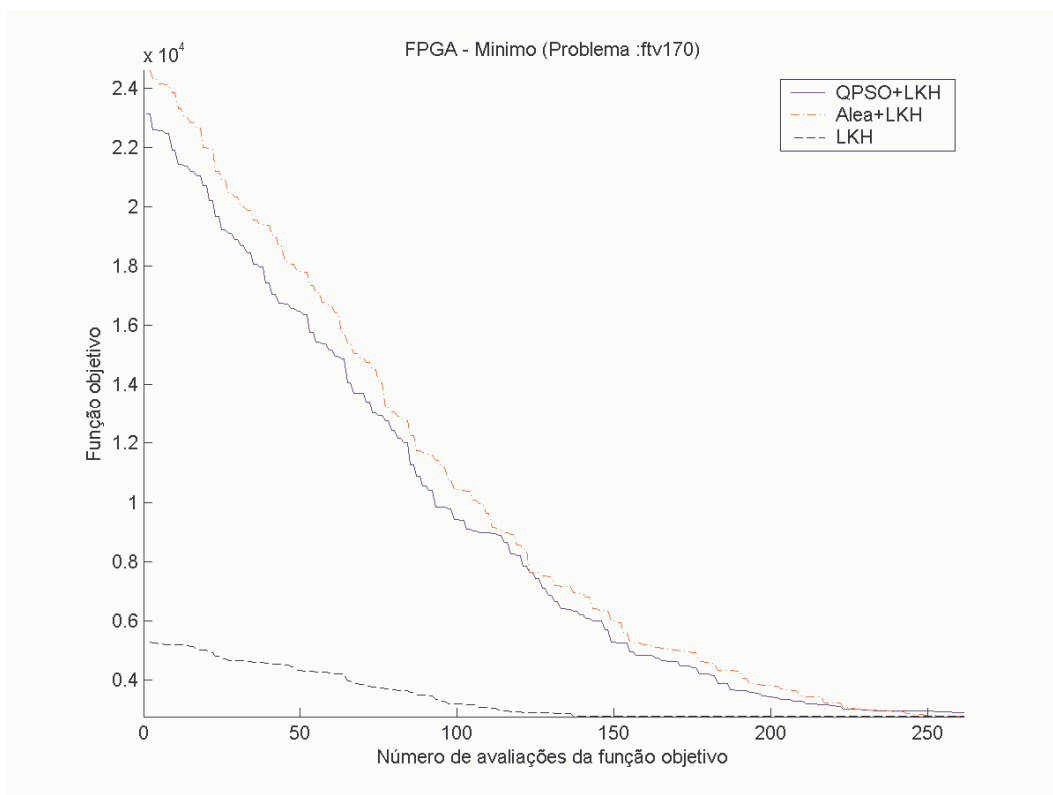


Figura 5.15. Mínimo obtido para instância ftv130 em FPGA.

Conclusões e Trabalhos Futuros

“Os Algoritmos com complexidade polinomial são aceitos como de origem de problemas solúveis na prática e, portanto, algoritmos computacionalmente viáveis” [LEW00]. Os problemas, como o do caixeiro viajante, apresentam algoritmos cuja complexidade não é limitada por um polinômio. Para problemas desse tipo, o espaço das soluções cresce de forma explosiva à medida que a quantidade de cidades cresce e, portanto seus algoritmos são computacionalmente inviáveis.

A Otimização utilizando outras abordagens recentes como uso de conceitos de enxame de partículas e mecânica quântica em problemas de PCV são métodos de otimização baseados na simulação da interação social entre os indivíduos em uma população. Nela, cada elemento desloca-se em um hiperespaço, atraído por posições (soluções promissoras) que ele considera promissoras. Essas posições promissoras são, por exemplo, a melhor posição que ele obteve e a melhor posição observada na sua vizinhança.

A heurística Lin-Kernighan é considerada um dos métodos mais eficazes para gerar soluções ótimas ou próximo-do-ótimo para o PCV simétrico. Entretanto, o projeto e a execução de um algoritmo baseado nesta heurística não é trivial. Há muitas decisões do projeto e de execução que devem ser feitas, e a maioria de decisões têm uma influência significativa no seu desempenho [HEL00].

Assim como a heurística Lin-Kernighan, a metaheurística QPSO apresenta vários parâmetros de configuração e implementação que afetam diretamente o desempenho do algoritmo proposto. Mesmo não tendo apresentando resultados satisfatório para instância pequenas do problema com poucas cidades (<1000), o algoritmo apresenta resultados promissores para instancias maiores (superiores as 1000 cidades), faixa cuja qual LKH não produz resultados tão eficientes [NGU07].

Novas investigações podem ser realizada, variando os parâmetros do algoritmo QPSO e assim adequá-los a cada uma das instâncias a ser testada. Pode variar o tamanho da população, numero de iterações e obviamente o LIP, isto provavelmente levaria a uma melhora de desempenho já que cada problema, mesmo com função objetivo igual, apresenta

variação de comportamento. Pode-se, por exemplo, ter instancias onde clusterizações funcionam bem como é o caso do algoritmo proposto por [NET99].

Mesmo de elevada complexidade computacional os algoritmos apresentados neste trabalho puderam ser implementados e testados em ambiente embarcação constituído de um processador *soft core* (MicroBlaze) implementando em uma FPGA com sistema *uCLinux kernel 2.4*. Entretanto foi evidente a perda de desempenho quando falamos em tempo de execução foi evidente, porém a qualidade dos resultados se manteve. Isto porem viabiliza a utilização de sistemas embarcados para resolução de instâncias pequenas de problemas de otimização, substituindo assim o PC convencional em tarefas onde tempo de resposta possa ser tolerado. Pode se ainda pensar em paralelismo e processamento distribuído, onde teria-se vários sistemas embarcados, por exemplo, placas RC200, se comunicado via interface *ethernet* cooperando mutuamente para resolver um determinado problema.

Deve-se ainda salientar o desenvolvimento de novas técnicas quânticas inspiradas utilizando simulação computacional. Um computador quântico é um dispositivo que executa cálculos fazendo uso direto de propriedades da mecânica quântica, tais como sobreposição e emaranhamento. Teoricamente, computadores quânticos podem ser implementados e o mais desenvolvido atualmente trabalha com poucos *qubits* de informação. Além disso, pensa-se na possível utilização de algoritmos quânticos como algoritmo de Shor e Grover para implementação de técnicas futuras para problemas de otimização.

Referências

- [AAR97] AARTS, E. H. L., LENSTRA, J.K. Local Search in Combinatorial Optimization, John Wiley & Sons, Chichester, England, 1997.
- [APP95a] APPLGATE, D., BIXBY, R. E., CHVÁTAL, V., COOK, W., Finding Cuts in the TSP: a Preliminary Report, Report No. 9505, Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), Rutgers University, New Brunswick, NJ, USA, 1995.
- [APP95b] APPLGATE, D., BIXBY, R. E., CHVÁTAL, V., COOK, W., Traveling Salesman Problem, Disponível em: <http://www.tsp.gatech.edu>. Acesso em: Março de 2007.
- [APP98] APPLGATE, D., BIXBY, R. E., CHVÁTAL, V., COOK, W., On the Solution of Traveling Salesman Problems, Technical Report CRPC-TR98744, Center for Research on Parallel Computation, Rice University, Houston, USA, pp. 645-656, 1998.
- [BEC96] BECKMAN, D. CHARI, A. N., DEVABHAKTUNI, S., PRESKILL, J., Efficient Networks for Quantum Factoring, Phys, vol. 54, pp. 1034-1063, 1996.
- [BEN80] BENIOFF, P., The computer as a Physical System: a Microscopic Quantum Mechanical Hamiltonian Model of Computers as Represented by Turing Machines, J. Stat Phys, vol. 22, pp. 563-591, 1980.
- [BER97] BERNSTEIN, E., VAZIRANI, U, Quantum complexity theory, Proceedings of the 25th ACM Symposium on the Theory of Computation, ACM Press, New York, NY, USA, vol. 26, no. 5, pp. 1411-1473, 1997.

- [BLA96] BLAZEWICZ, J., ECKER, K. H., PESCH, E. SCHMIDT, G., WEGLARZ, J., Scheduling Computer and Manufacturing Process, Springer-Verlag, Berlin, Germany, pp. 461, 1996.
- [BLU03] BLUM, C., ROLI, A., Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, ACM Computing Surveys, vol. 35, no. 5, pp. 268-308, 2003.
- [BOC58] BOCK, F., An Algorithm for Solving Traveling-Salesman and Related Network Optimization Problems, 14th National Meeting of ORSA, 1958.
- [CAM94] CAMPELLO, R. E., MACULAN, N. Algoritmos e Heurísticas: Desenvolvimento e Avaliação de Performance, Editora da Universidade Federal Fluminense, Niterói, RJ, 1994.
- [CHA94] CHAN, P. K., MOURAD, S., Digital Design Using Field-Programmable Gate Arrays, Prentice Hall, NJ, USA, 1994.
- [CHA03] CHAVES, A. A., Modelagem Exata e Heurística para Resolução do Problema do Caixeiro Viajante com Coleta de Prêmios, Trabalho de Conclusão, Curso de Ciência da Computação, Departamento de Computação, Universidade Federal de Ouro Preto, MG, 2003.
- [CHA05] CHAVES, A. A., Heurísticas Híbridas com Busca através de Agrupamentos para o Problema do Caixeiro Viajante com Coleta de Prêmios, Dissertação de mestrado, Pós-Graduação em Computação Aplicada, Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, 2005.
- [CHO01] CHONG, Y. N., Heuristic Algorithms for Routing Problems. PhD Thesis, School of Mathematics and Statistics, Curtin University of Technology, Australia, 2001.
- [CHR69] CHRISTOFIDES, N., EILON, S., An Algorithm for the Vehicle-Dispatching Problem. Operational Research Quaterly, vol. 20, pp. 309-318, 1969.

- [CHR72] CHRISTOFIDES, N., EILON, S, Algorithms for Large-scale Traveling Salesman Problems, Operational Research Quaterly, vol. 23, pp. 511-518, 1972.
- [CHR75] CHRISTOFIDES, N., Graph Theory: An Algorithm Approach. Academic Press. Inc., 1975.
- [CHU00] CHUANG, I., NIELSEN, M., Quantum Computation and Quantum Information, Cambridge University Press, Cambridge, England, 2000.
- [CLE05] CLERC, M., Particle Swarm Optimization, ISTE Ltd, London, UK, 2005.
- [COE06] COELHO, L.S., HERRERA, B. M., Fuzzy Modeling Using Chaotic Particle Swarm Approaches Applied to a Yo-yo Motion System. Proceedings of IEEE International Conference on Fuzzy Systems, Vancouver, BC, Canada, pp. 10508-10513, 2006.
- [COE07] COELHO, L. S., A Quantum Particle Swarm Optimizer with Chaotic Mutation Operator, Chaos, Solitons and Fractals, 2007 (aceito para futura publicação).
- [COE07] COELHO, L.S., HERRERA, B. M., Fuzzy Identification Based on a Chaotic Particle Swarm Optimization Approach Applied to a Nonlinear Yo-yo Motion System, IEEE Transactions on Industrial Electronics, 2007.
- [COO71] COOK, S., The complexity of theorem-proving procedures, Proceedings of the 3rd Symposium on the Theory of Computing, New York, NY, USA, pp. 151-158, 1971.
- [CRO58] CROES, G., A Method for Solving Traveling Salesman Problems, Operations Research, vol. 6, pp.791-8112, 1958.
- [CRO80] CROWDER, H., PADBERG, M. W., Solving Large-scale Symmetric Traveling Salesman Problems to Optimality, Management Science, vol. 26, pp. 495-509, 1980.

- [DAN54] DANTZIG, G. B., FULKERSON, D. R., JOHNSON, S. M., Solution of a Large-scale Traveling Salesman Problem, *Operations Research*, vol. 2, pp. 393-41, 1954.
- [DEC06] DE CASTRO, L. N., *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*, Chapman & Hall/CRC, Boca Raton, CA, USA, 2006.
- [DEU85] DEUTSCH, D., Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer, *Proc. Royal Soc. London, Ser. A*, vol. 400, no. 1818, pp. 97-117, 1985.
- [DEU89] DEUTSCH, D., Quantum Computational Networks, *Proc. Roy. Soc. London, Ser. A*, vol. 425, no. 1868, pp.73-90, 1989.
- [DOR04] DORIGO, M., STÜTZLE, T. *Ant Colony Optimization*, A Bradford Book, The MIT Press, London, UK, 2004.
- [EIS94] EISBERG, R., RESNICK, R., *Física Quântica – Átomos, Moléculas, Sólidos Núcleos e Partículas*, Editora Campus, Rio de Janeiro, RJ, 1994.
- [FEO95] FEO, T. A., RESENDE, M. G. C., Greedy Randomized Adaptive Search Procedures, *Journal of Global Optimization*, vol. 6, pp. 109-133, 1995.
- [FEY82] FEYMANN, R. P., Simulating physics with computers, *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467-488, 1982.
- [FEY96] FEYMAN, R., *Feynman Lectures in Computation*. Addison-Wesley Publishing Company, 1996.
- [FIC04] FICHER, T., MERZ, P., Embedding a Chained Lin-Kernighan Algorithm into a Distributed Algorithm, Technische Universität Kaiserslautern – Fachbereich Informatik 2004. Disponível em <http://dag.informatik.uni-kl.de/papers/tr331-04.pdf>.

- [FRÖ01] FRÖHLICH, H., KOIR, A., ZAJC, B., Optimization of FPGA configurations using parallel genetic algorithm, *Information Sciences*, vol. 133, no. 3-4, pp. 195-219, 2001.
- [GAR79] GAREY, M. R. e JOHNSON, D. S. *Computers and Intractability. A Guide to the Theory of NP-Completeness*, Freeman, New York, USA, 1979.
- [GIL02] GIL, A. C., *Como Elaborar Projetos de Pesquisa*. 4^a ed., São Paulo: Atlas, 2002.
- [GLO98] GLOVER, F., LAGUNA, M., *Tabu Search*, Kluwer Academic Pub. 1998.
- [GRO88] GRÖTSCHEL, M., LOVÁSZ, L., SCHRIJVER, A, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, London, UK, 1988.
- [GOL00] GOLDBARG, M. C., LUNA, H. P. L. *Otimização Combinatória e Programação Linear: Modelos e Algoritmos*. Rio de Janeiro, RJ, Campus, 2000.
- [GOL01] GOLDBARG, M. C., GOUVÊA, E. F., SILVA, L. M., *A Transgenetic Approach for the Graph Coloring Problem*, *Proceedings of European Operational Research Conference*, Rotterdam, Netherland, 2001.
- [HEL00] HELSGAUN, K., *An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic*, *European Journal of Operational Research*, vol. 126, pp. 106-130, 2000.
- [HER06] HERRERA, B. M., COELHO, L. S., *Nonlinear Identification of a Yo-yo System Using Fuzzy Model and Fast Particle Swarm Optimization*, *Applied Soft Computing Technologies: The Challenge of Complexity*, A. Abraham, B. De Baets M. Köppen, B. Nickolay (editors), Springer, London, UK, pp. 302-316, 2006.
- [HOF00] HOFFMAN, K. L. *Combinatorial Optimization: Current Successes and Directions for the Future*, *Journal of Computational and Applied Mathematics*, vol. 124, pp. 341-360, 2000.

- [HOG00] HOGG, T., PORTNOV, D. S., Quantum Optimization, Information Sciences, vol. 128, pp. 181-197, 2000.
- [HSU96] HSU, C., YAMADA, H. F., SHIDA, K., A fuzzy Self-tuning Parallel Genetic algorithm for Optimization, Computers & Industrial Engineering, vol. 30, no. 4, pp. 883-893, 1996.
- [HUA05] HUANG, D.S., ZHANG, X.-P, HUANG, G.-B. (Eds.): ICIC 2005, Part I, LNCS Springer-Verlag Berlin Heidelberg 3644, pp. 420-428, 2005.
- [ISH98] ISHIBUCHI, H., MURATA, T., Multi-objective Genetic Local Search Algorithm and its Application to Flowshop Scheduling, IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, vol. 28, no. 3, pp. 392-403, 1998.
- [JAS02] JASZKIEWICZ, A. Genetic Local Search for Multi-objective Combinatorial Optimization, European Journal of Operational Research, vol. 137, pp. 50-71, 2002.
- [JAS04] JASINSKI, R. P., Implementação de uma Cpu Tolerante a Falhas em Lógica Programável, Dissertação de Mestrado, Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, Centro Federal de Educação Tecnológica do Paraná, Curitiba, PR, 2004.
- [JIA00] JIAO, L., WANG, L., A Novel Genetic Algorithm Based on Immunity. IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans, vol. 30, no. 5, pp. 552-561, 2000.
- [JOH90] JOHNSON, D. S. Local Optimization and the Traveling Salesman Problem, Lecture Notes in Computer Science, vol. 442, pp. 446-461, 1990.
- [JOH97] JOHNSON, D. S. & McGEOCH, L. A., The Traveling Salesman Problem: A Case Study in Local Optimization in E. H. L. Aarts, J. K. Lenstra (eds.), Local Search in Combinatorial Optimization, Wiley, New York, NY, USA, 1997.

- [JUN93] JÜNGER, M., REINELT, G. e THIENEL, S. Provably Good Solutions for Traveling Salesman Problem, *Zeitschrift für Operations Research*, vol. 40, pp. 183-217, 1993.
- [JUN02] JUNG, S., MOON, B. -R. Toward Minimal Restriction of Genetic Encoding and Crossover for the Two-Dimensional Euclidian TSP, *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 6, pp. 557-565, 2002.
- [KAR71] HELD, M., KARP, R. M., The Traveling-Salesman Problem and the Minimum Spanning Trees: Part II, *Mathematical Programming*, vol. 6, pp. 6-25, 1971.
- [KAR72] KARP, R. M., Reducibility Among Combinatorial Problems, In: Miller, R. E., Thatcher, J. W. (Ed.), *Complexity of Computer Computation*, Plenum Press, New York, NY, USA, 1972.
- [KAR77] KARP, R. M., Probabilistic Analysis of Partitioning Algorithms for the Traveling-Salesman Problem in the Plane, *Mathematics of Operations Research*, vol. 2, pp. 209-224, 1977.
- [KEN95] KENNEDY, J., EBERHART, R., Particle Swarm Optimization, *Proceedings of IEEE International Conference on Neural Networks*, Perth, Australia, pp. 1942-1948, 1995.
- [LAG03] LAGUNA, M., MARTÍ, R., *Scatter Search: Methodology and Implementations in C*, Kluwer Academic Publishers, Boston, USA, 2003.
- [LAP92] LAPORTE, G., The Traveling Salesman Problem: An overview of exact and approximate algorithms, *European Journal of Operational Research*, vol. 59, pp.231-247, 1992.
- [LAW85] LAWLER, E. L., LENSTRA, J. K., KAN RINNOOY, A. H. G., SHMOYS(eds) D. B. S, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, New York, NY, USA, 1985.

- [LEV73] LEVIN, L., Universal Search Problems, Problemy. Peredaci Informacii, vol. 9, no. 3, pp. 265-266, 1973.
- [LIN65] LIN, S., Computer Solutions for the Traveling Salesman Problem, Bell. Systems Technology Journal, vol. 44, pp. 2245-2269, 1965.
- [LIN73] LIN, S., KERNIGHAN, B. W., An Effective Heuristic Algorithm for the Traveling Salesman Problem, Operations Research, vol. 21, pp. 498- 516, 1973.
- [LIN07] LIN, C., TSAI, H., FPGA implementation of a wavelet neural network with particle swarm optimization learning, Mathematical and Computer Modelling, 2007 (aceito para futura publicação).
- [LIU05] LUI, J., XU, W., SUN, J., Quantum-behaved Particle Swarm Optimization with Mutation Operator, Proceedings of 17th International Conference on Tools with Artificial Intelligence, Hong Kong, China, 2005.
- [LOP05] LOPES, H. S., COELHO, L. S., Particle Swarm Optimization with Fast Local Search for the Blind Traveling Salesman Problem, Proceedings of 5th International Conference on Hybrid Intelligent Systems, Rio de Janeiro, RJ, pp. 245-250, 2005.
- [LOU01] LOURENÇO, H. R., PAIXÃO, J. P., PORTUGAL, R., Multiobjective Metaheuristics for the Bus-driver Scheduling Problem, Barcelona, Spain, vol. 35, no. 3, pp. 331-343, 2001.
- [LVC06] LVCon - LABORATÓRIO VIRTUAL EM COMPUTAÇÃO NATURAL, Acesso: 20/08/2007, <http://lsin.unisantos.br/lvcon/lvcon>.
- [MAC05] MACHADO, T. R., LOPES, H. S., A Hybrid Particle Swarm Optimization Model for the Traveling Salesman Problem. Natural Computing Algorithms, Ribeiro, H., Albrecht, R. F., Dobnikar, A. (eds.), Springer, New York, NY, USA, pp. 255-258, 2005.

- [MAL02] MALAGUTTI, P. L., Malagutti, P versus NP, 2002. Disponível em:
<http://www.dm.ufscar.br/hp/hp501/hp501001/hp501001.htm>
- [MAR06] MARTÍ, R., LAGUNA, M., GLOVER, F. Principles of Scatter Search, European Journal of Operational Research, vol. 169, no. 2, pp. 359-372, 2006.
- [MLA97] MLADENOVIC, N., HANSEN, P., Variable Neighborhood Search, Computers and Operations Research, vol. 24, pp. 1097-1100, 1997.
- [MEL01] MELO, V. A., Metaheurísticas para o Problema do Caixeiro Viajante com Coleta de Prêmios, Dissertação de Mestrado, Instituto de Computação, Universidade Federal Fluminense, Niterói, Rio de Janeiro, RJ, 2001.
- [MER02] MERKLE, D., MIDDENDORF, M., Fast Ant Colony Optimization on Runtime Reconfigurable Processor Arrays, Genetic Programming and Evolvable Machines, vol. 3, no. 4, pp. 345-361, 2002.
- [MER61] MERZBACHER, E., Quantum Mechanics, John Wiley & Sons, INC, New York, NY, USA. 1961.
- [MIL91] MILLER, D. L., PEKONY, J. F., Exact solution of large asymmetric traveling salesman problems, Science, 251, pp. 754-761, 1991.
- [MOT01] MOTTA, L. C. S. Novas Abordagens para o Problema de Recobrimento de Rotas, Dissertação de Mestrado, Pós-Graduação em Ciência da Computação, Universidade Federal Fluminense, Niterói, RJ, 2001.
- [MUR95] MURGAI, R., BRAYTON, R. K., SANGIOVANNI-VINCENTELLI, A., Logic synthesis for field-programmable gate arrays, Kluwer Academic Publisher, 1995.
- [NEM88] NEMHAUSER, G. L., WOLSEY, A. L., Integer and Combinatorial Optimization. Wiley, New York, NY, USA, 1988.

- [NET99] NETO, D. M., Efficient Cluster Compensation for Lin-Kernighan Heuristics, PhD Thesis, Department of Computer Science, University of Toronto, Canada, 1999.
- [NGU07] NGUYEN, H. D, YOSHIHARA, I., YAMAMORI, M, Implementation of an Effective Hybrid GA for Large-Scale Traveling Salesman Problems, IEEE Transaction on System, Man, and Cybernetics-PART B: Cybernetics, vol. 37, no. 1, pp 92-99, 2007.
- [NGU06] NGUYEN, H. D., YOSHIHARA, I., YAMAMORI, K., YASUNAGA, M., Lin-Kernighan Variant, 2006.
Disponível em: <http://public.research.att.com/~dsj/chtsp/nguyen.txt>
- [NOR00] NORBRY, J., Quantum Mechanics Physics Department, University of Wisconsin-Milwaukee, Milwaukee, 2000.
- [OCH94] OCHI, L. S. Conhecimento heurístico: aplicações em problemas de otimização, XIV Congresso da Sociedade Brasileira de Computação, Caxambu, MG, 1994.
- [OLD95] OLDFIELD, J. V., DORF, R. C., Field-programmable gate arrays, John Wiley & Sons, Inc., 1995.
- [OSM96] OSMAN, I. H., LAPORTE, G., Metaheuristics: A bibliography, Annals of Operations Research 63, pp. 513-623, 1996.
- [PAD91] PADBERG, M.W., RINALDI, G., A Branch and Cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems, SIAM Review, vol.33, pp.60-100, 1991.
- [PAD87] PADBERG, M.W., RINALDI, G., Optimization of a 532-city symmetric traveling salesman problem by branch and cut, Operations Research Letters, no. 6, pp. 1-7, 1987.
- [PAN04a] PANG, W. J., WANG, K.-P., ZHOU, C. -G., DONG, L.-J., Fuzzy Discrete Particle Swarm Optimization for Solving Traveling Salesman Problem, Proceedings of 4th

International Conference on Computer and Information Technology, Washington, DC, USA, pp. 796-800, 2004.

[PAN04b] PANG, W., WANG, K. -P., ZHOU, C. -G., DONG, L. -J., LIU, M., ZHANG, H. -Y., WANG, J.-Y., Modified Particle Swarm Optimization Based on Space Transformation for Solving Traveling Salesman Problem, Proceedings of the Third International Conference on Machine Learning and Cybernetics, vol. 4, pp. 2342-2346, 2004.

[PAN05] PANG, X. F., Quantum Mechanics in Nonlinear Systems, World Scientific Publishing Company, River Edge, NJ, USA, 2005.

[PAP82] PAPADIMITRIOU, C. H., STEIGLITZ, K., Combinatorial Optimization - Algorithms and Complexity, Dover Publications, Inc., New York, NY, SA, 1982.

[PRE06] PRESTES, Á. N., Uma Análise Experimental de Abordagens Heurísticas Aplicadas ao Problema do Caixeiro Viajante. Dissertação de Mestrado, Programa de Pós-Graduação em Sistemas de Computação, Universidade Federal do Rio Grande do Norte, Natal, RN, 2006.

[PRO02] PROTOPESCU, V., BARHEN, J., Solving a Class of Continuous Global Optimization Problems Using Quantum Algorithms, Physics Letters A, vol. 296, pp. 9-14, 2002.

[POH71] POHL, H. A., Introdução à Mecânica Quântica, Editora Edgard Blücher Ltda, São Paulo, SP, 1971.

[RAM2005] RAMOS, I. C. O., Metodologia Estatística na Solução do Problema do Caixeiro Viajante e na Avaliação de Algoritmos: Um Estudo Aplicado à Transgenética Computacional. Tese de Doutorado, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Universidade Federal do Rio Grande do Norte, Natal, RN, 2005.

- [RAY96] RAYWARD-SMITH, V. J., REEVES, C. R., OSMAN, I. H., SMITH, G. D.,
Modern Heuristic Search Methods, John Wiley & Sons, 1996.
- [REI91] REINELT, G., TSPLIB - A Traveling Salesman Problem Library, ORSA Journal on
Computing, vol. 3, no. 4, pp. 376-384, 1991. Avaliada em <http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB/TSPLIB.html>.
- [REI94] REINELT, G., The Traveling Salesman: Computational Solutions for TSP
Applications, Lecture Notes in Computer Science, 840 (1994).
- [RIB02] RIBEIRO, C. C., Problema do Caixeiro Viajante, Notas de aula, Disciplina
Complexidade de Algoritmos, Dissertação de Mestrado, Mestrado em Ciência da
Computação, Departamento de Computação e Estatística, Universidade Federal do Mato
Grosso, Campo Grande, MT, 2002.
- [ROD00] RODRIGUES, M. A. P., Problema do Caixeiro Viajante: Um Algoritmo para
Resolução de Problemas de Grande Porte, Dissertação de Mestrado, Mestrado em
Engenharia de Produção e Sistemas, Florianópolis, SC, 2000.
- [ROM04] ROMERO, R., MONTOVANI, J. R. S. Introdução a Metaheurísticas, Anais do III
Congresso Temático de Dinâmica e Controle da SBMAC, UNESP, Ilha Solteira, SP,
2004.
- [ROO00] ROOK, C., Investigation into the Use of Ant Algorithms for the Traveling
Salesman Problem and Game Playing. Thesis of Master Science, University of
Northumbria, Newcastle, UK, 2000.
- [ROS77] ROSENKRANTZ, D. E., STREAMS, R. E., LEWIS II, P. M., An analysis of
several heuristics for the traveling salesman problem, SIAM J. Comput., vol.6, pp.563-
581, 1977.
- [RYD79] RYDNIK, V., ABC's of Quantum Mechanics, Peace Publishers, Moscow, U.R.S.S,
1979.

- [SCH04] SCHEUERMANN, B., SO, K., GUNTSCH, M., MIDDENDORF, M., DIESSEL, O., ELGINDY, H., SCHMECK, H., FPGA Implementation of Population-based Ant Colony Optimization, *Applied Soft Computing*, vol. 4, pp. 303-322, 2004.
- [SCH96] SCHOONDERWOERD, O., HOLLAND, BRUTEN, J., ROTHKRANTZ, L., Ant based Load Balancing in Telecommunications Networks, *Adaptive Behavior*, vol. 5, no. 2, pp. 169-207, 1996.
- [SCH35] SCHRÖDINGER, E. "Die gegenwärtige Situation in der Quantenmechanik" ("A situação atual da mecânica quântica."), *Naturwissenschaften*, vol. 23, no. 807, pp.823-844, 1935
- [SHI99] SHI, Y., EBERHART, R. Empirical study of particle swarm optimization Proc. Of Congress on Evolutionary Computation, pp. 1945-1950, 1999.
- [SIQ05] SIQUEIRA, P. H., Uma Nova Abordagem na Resolução do Problema do Caixeiro Viajante, Dissertação de mestrado, Pós-Graduação em Métodos Numéricos em Engenharia, Universidade Federal do Paraná, Curitiba, PR, 2005.
- [SIN98] SINGH, S., O Último Teorema de Fermat, Ed. Record, Rio de Janeiro, RJ 1998
- [SMA98] SMALE, S., Mathematical Problems for the Next Century, *Mathematical Intelligencer*, vol.20, no. 2, pp. 7-15, 1998.
- [SOU06] SOUZA, G. R., Uma Abordagem por Nuvem de Partículas para Problemas de Otimização Combinatória, Dissertação de mestrado, Pós-Graduação em Sistemas e Computação, Universidade Federal do Rio Grande do Norte, Natal, RN, 2006.
- [SPE89] SPEARS, W. M., Using Neural Networks and Genetic Algorithms as Heuristics for NP-Complete Problems, Thesis of Master Degree of Science in Computer Science, George Mason University, Fairfax, Virginia, USA, 1989.

- [SUN04a] SUN, J., FENG, B., XU, W., Particle Swarm Optimization with Particles Having Quantum Behavior, Proceedings of Congress on Evolutionary Computation, Portland, Oregon, USA, pp. 325-331, 2004.
- [SUN04b] SUN, J., FENG, B., XU, W., A Global Search Strategy of Quantum-Behaved Particle Swarm Optimization, Proceedings of IEEE Congress on Cybernetics and Intelligent Systems, Singapore, pp. 111-116, 2004.
- [SUN05] SUN, J., XU, W., FENG, B., Adaptive Parameter Control for Quantum-behaved Particle Swarm Optimization on Individual Level, Proceedings of IEEE International Conference on Systems, Man and Cybernetics, Big Island, HI, USA, pp. 3049-3054, 2005.
- [TAS03] TASGERITEN, M. F., LIANG, Y. -C., a Binary Particle Swarm Optimization for Lot Sizing Problem, Journal of Economic and Social Research, vol. 5, no. 2, 2003.
- [TEB06] TEBALDI, A., COELHO, L. S., LOPES JR, V., Detecção de Falhas em Estruturas Inteligentes Usando Otimização por Nuvem de Partículas: Fundamentos e Estudos de Casos, SBA Controle & Automação, vol. 17, no. 3, pp. 312-330, 2006.
- [TSA04] TSAI, H. -K., YANG, J. -M., TSAI, Y. -F., KAO, C. -Y., An Evolutionary Algorithm for Large Traveling Salesman Problems, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 34, no. 4, pp. 1718-1729, 2004.
- [VIG04] VIGNATI, A. L., NETTO, F. S., BITTENCOURT, L. F., Uma Introdução à Computação Quântica, Departamento de Informática, Monografia de Conclusão de Curso, Universidade Federal do Paraná, Curitiba, PR, 2004.
- [WAL01] WALSHAW, C., A multilevel Lin-Kernighan-Helsgaun Algorithm for the Travelling Salesman Problem, Mathematical Research Report 01/IM/80, Comp. Math. Sci, University of Greenwich, UK, 2000.

- [WAN03] WANG, K. -P., HUNAG, L., ZHOU, C. -G., PANG, C. -C., WEI, P., Particle Swarm Optimization for Traveling Salesman Problem, Proceedings of the 2nd International Conference on Machine Learning and Cybernetics, Xi'an, China, pp. 1583-1585, 2003.
- [WIL71] WILSON, E. O., The Insect Societies, Belknap Press of Harvard University Press, Cambridge, MA, 1971.
- [YIN02] YIN R. K., Estudo de Caso: Planejamento e Métodos, 2^a-ed., Porto Alegre, RS, Ed. Bookman, 2001.
- [ZEB02] ZEBULUM, R. S., PACHECO, M. A. C., VELLASCO, M. M. B. R. Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms, The CRC Press, Boca Raton, CA, USA, 2002.