UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Uma Fundamentação Teórica para a Complexidade Estrutural de Problemas de Otimização

por

LIARA APARECIDA DOS SANTOS LEAL

Tese de Doutorado
submetida à avaliação, como requisito parcial
para a obtenção do grau de
Doutor em Ciência da Computação.

Prof. Dr. Dalcidio Moraes Claudio
Orientador

Profª. Drª. Laira Vieira Toscani
Co-Orientadora

Porto Alegre, abril de 2002.

CIP – Catalogação na Publicação

Leal, Liara Aparecida dos Santos

Uma Fundamentação Teórica para a Complexidade Estrutural de Problemas de Otimização / por Liara Aparecida dos Santos Leal. – Porto Alegre: PPGC da UFRGS, 2002.
116 f.: il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR – RS, 2002. Orientador: Claudio, Dalcidio Moraes; Co-Orientadora: Toscani, Laira Vieira.

1. Complexidade Estrutural. 2. Problema de Otimização. 3. Algoritmo Aproximativo. 4. Teoria das Categorias. 5. Teoria de Universais. 6. Teoria Categorial da Forma. I. Claudio, Dalcidio Moraes II. Toscani, Laira Vieira. III. Título.

Dedico este trabalho

aos meus pais
Ozi e Ciara,

aos meus filhos
Alexandre e Artur,

e

ao meu marido
Evaldo Dombrowski Leal.

Agradecimentos

Foram os professores Dalcidio Moraes Claudio e Laira Vieira Toscani que me apresentaram os fundamentos da Matemática da Computação e, após um ano como aluna especial, participando dos seminários sobre Teoria dos Domínios (junto com Graçaliz, Renata, Gerardo, Javier, Fabiana e Beatriz), fui por eles convidada para ingressar no curso de doutorado do Programa de Pós-Graduação em Computação, do Instituto de Informática - UFRGS.

Começou então para mim um período de intenso trabalho, dividindo meu tempo entre as aulas ministradas na Faculdade de Matemática – PUCRS e toda a formação necessária para obter o título de doutora em ciência da computação.

No entanto, motivação era o que não faltava, pois podia contar com o carinho e incentivo de toda minha família (especialmente meus pais Ozi e Ciara e meus irmãos Daltro e Cléverson), além do amor incondicional de meus filhos Alexandre e Artur, reforçado pelo companheirismo, apoio e segurança de meu marido Evaldo. Nunca esquecerei todo o esforço e sacrifício que fizeram para que eu pudesse ter uma permanência tranqüila e proveitosa durante os seis meses de visita a Portugal, Alemanha e Escócia. Obrigado pela dádiva de tê-los sempre ao meu lado.

Sou muito grata a Daniele e Martin Escardó, Nancy e Matias pela afetuosa acolhida em Edimburgo. Especial agradecimento ao Prof. Charles Rattray pela atenção e interesse dedicado ao meu trabalho durante visita à Universidade de Stirling, não esquecendo que fui introduzida a ele através do Prof. Blauth, com quem tive muitas e valiosas discussões sobre teoria das categorias.

Merece, também, um agradecimento muito especial o Prof. Edward H. Haeusler, que se mostrou sempre receptivo nas diversas ocasiões em que nos encontramos, enriquecendo a pesquisa com valiosas idéias e sugestões.

Aos colegas da Faculdade de Matemática - PUCRS, especialmente ao Nilton que, como diretor, sempre me apoiou e incentivou, e a todas as pessoas que participaram, vibraram, sofreram e torceram para que eu chegasse até aqui, muito obrigada pelo exemplo, amizade, amor e carinho altruístas. Obrigada aos meus orientadores, meus mestres, meus amigos. Esta conquista também é de vocês.

E, por último, mas principalmente, agradeço a Deus que, por intermédio de minha protetora espiritual N. S. Aparecida, ilumina e conduz todos os meus passos.

Sumário

Lista de Abreviaturas

AG	Algoritmos Genéticos
CPO	Ordem Parcial Completa (do inglês: Complete Partial Order)
CUSL	Semi-reticulado condicionalmente superior com menor elemento (do inglês: Conditional Upper Semi-Lattice with least element)
MINSAT	Satisfabilidade Mínima
MTD	Máquina de Turing Determinística
MTND	Máquina de Turing Não-Determinística
PRAM	Máquina Paralela de Acesso Randômico (do inglês: Parallel Random Acess Machine)
SAT	Satisfabilidade
sss	se e somente se
VLSI	Integração em escala muito grande (do inglês: Very Large Scale Integration)

Lista de Símbolos

Σ	Conjunto finito de símbolos de uma linguagem.
Σ^*	Conjunto de palavras finitas de símbolos de Σ.
L_M	Linguagem reconhecida pela máquina M.
$\boldsymbol{0}[g(n)]$	Classe de ordem assintótica de g.
$\mathbf{N}$	Conjunto dos números naturais.
$\mathbf{Z}^+$	Conjunto dos números inteiros positivos
$\mathbb{R}$	Conjunto dos números reais
P	Classe de problemas de decisão resolvíveis por algoritmo determinístico de tempo polinomial.
NP	Classe de problemas de decisão resolvíveis por algoritmo não-determinístico de tempo polinomial.
FP	Classe de todos os problemas resolvíveis em tempo polinomial.
Co-P	Classe de problemas complementares à classe P.
NC	Classe de problemas paralelizáveis em P.
Co-NP	Classe de problemas complementares à classe NP.
$L_1 \propto L_2$	Linguagem L_1 polinomialmente transformável na linguagem L_2.
$\Pi_1 \propto \Pi_2$	Problema de decisão Π_1 polinomialmente transformável no problema de decisão Π_2.
NP-Completo	Classe de problemas em NP que são completos para transformação polinomial.
NPI	Classe de problemas NP-intermediário.
NP-Difícil	Classe de problemas de busca que são completos para transformações polinomiais generalizadas.
$OPT_\Pi(I)$	Valor da solução ótima para a entrada I do problema Π.
$R_a(I)$	Razão de qualidade do algoritmo aproximativo a para a entrada I.
R_a	Qualidade absoluta do algoritmo aproximativo a.
$R_a^{¥}$	Qualidade assintótica do algoritmo aproximativo a.
NPO	Classe de problemas de otimização correspondente à classe NP.
APX	Classe de problemas aproximáveis em NPO.
PAS	Classe de problemas de aproximação polinomial em NPO.
FPAS	Classe de problemas de aproximação totalmente polinomial em NPO.
PO	Classe de problemas em NPO resolvíveis em tempo polinomial.

$\propto_{APX}$	Relação de APX-redutibilidade.
NPO-Completo	Classe de problemas em NPO que são completos para APX-redutibilidade.
$[D \to D]$	Espaço das funções contínuas sobre D.
$\cong$	Relação de isomorfismo
$\sqsubseteq$	Relação de ordem parcial
$\bot$	Menor elemento em relação a ordem parcial $\sqsubseteq$.
$\bigsqcup A$	Supremo do conjunto A
D_c	Conjunto dos elementos finitos do cpo D.
aprox(x)	Conjunto dos aproximantes finitos de x.
[a]	Ideal principal gerado por a.
$\overline{P}$	Conjunto de ideais do cusl P.
$\mathbb{A}$	Conjunto de algoritmos aproximativos para um problema de otimização.
$\mathbb{A}_c$	Conjunto dos algoritmos finitos de $\mathbb{A}$
$\overline{A}$	Conjunto de ideais do cusl $\mathbb{A}$
$\varepsilon_a(I)$	Requerimento de qualidade do algoritmo aproximativo a para a entrada I
$\approx_\varepsilon$	Relação de equivalência definida em termos do requerimento de qualidade ε
$\leq_{AP}$	AP-redução (approximation preserving)
I	Conjunto de instâncias para um problema de otimização
S_x	Conjunto de soluções viáveis para a instância $x \in I$
$m_x(y)$	Medida da solução viável y para a instância $x \in I$
y*(x)	Solução ótima para a instância $x \in I$
m*(x)	Medida de qualquer solução ótima para a instância $x \in I$
E(x,y)	Erro relativo da solução viável y para a instância $x \in I$
R(x,y)	Razão de qualidade da solução viável y para a instância $x \in I$
$\sqsubseteq_\varepsilon$	Relação de ordem parcial de ε-aproximação
$\bot_\varepsilon$	Menor elemento em relação a ordem parcial $\sqsubseteq_\varepsilon$
$\bigsqcup D$	Supremo do conjunto D
aprox(a)	Conjunto dos aproximantes finitos do algoritmo a
MAX	Problema de maximização
MIN	Problema de minimização

p_C	Problema construtivo associado ao problema de otimização *p*
p_A	Problema de avaliação associado ao problema de otimização *p*
p_D	Problema de decisão associado ao problema de otimização *p*
μ	Relação de participação para a teoria de universais
$\approx$	Relação de equivalência para a teoria de universais
α	Redução abstrata entre problemas de otimização
♦	Símbolo de término de uma demonstração
$\circ$	Composição de funções
χ_A	Função característica do conjunto A
Set	Categoria dos conjuntos
$\mathbf{C}\downarrow B$	Categoria “comma” de **C**-objetos sobre B
OPTS	Categoria dos problemas de otimização solucionáveis em tempo polinomial
OPT	Categoria dos problemas de otimização da classe NPO
$APX_{B,K}$	Categoria das aproximações para um problema de otimização

Lista de Figuras

Lista de Quadros

Resumo

Com o objetivo de desenvolver uma fundamentação teórica para o estudo formal de problemas de otimização NP-difíceis, focalizando sobre as propriedades estruturais desses problemas relacionadas à questão da aproximabilidade, este trabalho apresenta uma abordagem semântica para tratar algumas questões originalmente estudadas dentro da Teoria da Complexidade Computacional, especificamente no contexto da *Complexidade Estrutural*.

Procede-se a uma investigação de interesse essencialmente teórico, buscando obter uma formalização para a teoria dos algoritmos aproximativos em dois sentidos. Por um lado, considera-se um algoritmo aproximativo para um problema de otimização genérico como o principal objeto de estudo, estruturando-se matematicamente o conjunto de algoritmos aproximativos para tal problema como uma ordem parcial, no enfoque da Teoria dos Domínios de Scott. Por outro lado, focaliza-se sobre as reduções entre problemas de otimização, consideradas como morfismos numa abordagem dentro da Teoria das Categorias, onde problemas de otimização e problemas aproximáveis são os objetos das novas categorias introduzidas. Dentro de cada abordagem, procura-se identificar aqueles elementos *universais*, tais como elementos finitos, objetos totais, problemas completos para uma classe, apresentando ainda um sistema que modela a hierarquia de aproximação para um problema de otimização NP-difícil, com base na *teoria categorial da forma*.

Cada uma destas estruturas matemáticas fornecem fundamentação teórica em aspectos que se complementam. A primeira providencia uma estruturação *interna* para os objetos, caracterizando as classes de problemas em relação às propriedades de aproximabilidade de seus membros, no sentido da Teoria dos Domínios, enquanto que a segunda caracteriza-se por relacionar os objetos entre si, em termos de reduções preservando aproximação entre problemas, num ponto de vista *externo*, essencialmente categorial.

Palavras-Chave: Complexidade Estrutural, Problema de Otimização, Algoritmo Aproximativo, Teoria dos Domímios, Teoria das Categorias, Teoria de Universais, Teoria Categorial da Forma.

TITLE: "A THEORETICAL FOUNDATION TO STRUCTURAL COMPLEXITY OF OPTIMIZATION PROBLEMS"

Abstract

Aiming at developing a theoretical framework for the formal study of NP-hard optimization problems, with a special focus on structural properties related to approximation issues, this work presents a semantic approach to cope with several questions originally studied within structural computational complexity theory.

In particular, a special attention is given in providing a theoretical foundation for the formalization of approximation algorithms theory. On the one hand, a given approximation algorithm is considered as the main object of investigation, and the collection of all such objects is reinterpreted as a Scott domain partial order. On the other hand, the focus is now changed to reductions between optimization problems, and these are then considered as morphisms in the categorical sense, where optimization and approximation problems are viewed as the obvious objects. Within each approach, universal objects, as for instance finite elements, total objects and complete problems for a specific complexity class, are properly characterized. In addition and most importantly, a formal system modeling the approximation hierarchy of a given optimization problem is provided, based on categorical shape theory.

Both these approaches, i.e., the domain-based and the categorical one, provide complementary views of this mathematical foundation. In more detail, the first one provides an *internal* structure to the objects, characterizing the complexity class of the problems in terms of the approximation properties of their members, whereas the second one, characterizes the relationship between these objects by means of approximation-preserving reductions between optimization problems, in an *external* point of view.

Keywords: Structural Complexity, Optimization Problems, Approximative Algorithms, Domain Theory, Category Theory, Universals Theory, Categorical Shape Theory.

1 Introdução

O presente trabalho constitui-se na tese de doutorado, requisito parcial para a obtenção do título de doutor em Ciência da Computação, dentro do Programa de Pós-Graduação em Computação da UFRGS, estando inserido na área de pesquisa da Matemática da Computação e tendo como tema a fundamentação teórica da hierarquia de aproximação para problemas de otimização.

Este capítulo contextualiza o estudo de algoritmos aproximativos para problemas de otimização dentro da Teoria da Ciência da Computação, particularmente dentro da Teoria da Complexidade Computacional, apresentando o estado da arte do tema proposto e justificando a proposta de uma abordagem semântica adequada para tratar com as peculiaridades relativas à questão da aproximabilidade para problemas de otimização.

O trabalho é composto por oito capítulos, sendo este primeiro destinado à apresentação do tema da tese e dos aspectos estruturais deste volume. A primeira seção deste capítulo apresenta a motivação e as justificativas para o tema escolhido. A seção seguinte apresenta a formulação da tese e os objetivos propostos. Os trabalhos relacionados com o tema proposto são considerados na terceira seção e nas seções seguintes são apresentadas as delimitações do escopo do trabalho e a descrição da estrutura dos demais capítulos.

1.1 Tema: Motivação e Justificativa

A princípio, a motivação para este trabalho pode ser creditada ao desejo de dar continuidade à pesquisa iniciada na tese de doutorado de L. V. Toscani [TOS 88], onde é apresentada uma formalização para métodos de desenvolvimento de algoritmos, tais como "divisão e conquista" e "programação dinâmica", permitindo uma análise comparativa entre tais métodos em relação a sua generalidade e à complexidade dos algoritmos por eles gerados. Entretanto, em relação aos algoritmos aproximativos ([TOS 86]] e [TOC 86]]), considerados como a alternativa mais viável para a solução de problemas intratáveis, os trabalho citados concluem que os métodos de desenvolvimento são bem mais complexos, exigindo a formalização de mais de um nível de abstração para um melhor entendimento.

Desde o seu aparecimento, no início da década de sessenta, a Teoria da Complexidade vem firmando-se como uma das mais importantes áreas de pesquisa dentro da Ciência da Computação. Conforme [BLU 98], foi justamente com o advento dos computadores digitais, e sua promessa de resolver problemas considerados até então intratáveis, que o interesse pelos problemas classificados como solúveis (no sentido de ser conhecido algum algoritmo que os resolvessem) revigorou-se na busca por algoritmos eficientes. Entretanto, apesar do sucesso inicial obtido, tornou-se evidente que uma quantidade considerável de problemas (tal como o famoso problema do caixeiro viajante) continuavam aguardando por soluções eficientes. Desta maneira, aparecia entre os problemas solúveis, uma natural e aparentemente rica hierarquia com a dicotomia de tratabilidade/intratabilidade, refletindo a dicotomia já existente de decidibilidade/indecidibilidade. Nascia, assim, a Teoria da Complexidade Computacional, cujo formalismo baseou-se nos modelos e formalismos da teoria da computação clássica.

Muito do trabalho nessa área concentrou-se na identificação de medidas formais de complexidade, com o objetivo de classificar os problemas computacionais em classes segundo os inerentes graus de dificuldade. No artigo "An overview of Computational Complexity", Cook [COO 83] apresenta um histórico da teoria da complexidade, mencionando os mais interessantes e importantes resultados obtidos, desde o início do desenvolvimento da área de Complexidade Computacional. Sob o seu ponto de vista, a identificação da classe de problemas solúveis deterministicamente em tempo limitado por um polinômio no comprimento da entrada, hoje conhecida por classe P, foi um dos mais importantes conceitos introduzidos nesta área, sendo sua formalização determinante para o desenvolvimento da Teoria da Complexidade. Assim também se considera a classe NP de problemas que admitem algoritmos polinomiais não-determinísticos, a qual divide, com a classe P, a mais famosa questão aberta em Ciência da Computação: "Será P = NP ?"

A discussão dessa questão foi o ponto inicial para a teoria da NP-completude, que está baseada da noção de *redutibilidade* de problemas. O conceito de redutibilidade foi introduzido por Cook, no início da década de 70 [COO 71], e desempenha um papel fundamental em Complexidade Computacional, já que possibilita uma maneira de comparar o grau de dificuldade para se resolverem dois problemas e permite que seja formalizado o conceito do "mais difícil" elemento de uma classe de problemas, em relação às respectivas complexidades de seus membros.

Segundo Papadimitriou [PAP 94], a importância de uma classe de complexidade vem tanto da importância do fenômeno computacional que ela captura, quanto dos problemas que ela contém, especialmente aqueles que são completos. Desta maneira, pode-se justificar o interesse crescente da NP-completude nas últimas três décadas. Se, por um lado, o estudo de problemas que são NP-completos ou possivelmente mais fáceis, para uma determinada classe, estimula o desenvolvimento da área de análise e projeto de algoritmos, por outro lado a preocupação com problemas que são NP-difíceis (no sentido que são pelo menos tão difíceis quanto os problemas NP-completos), têm estimulado o desenvolvimento das áreas de heurísticas e algoritmos aproximativos.

Ao identificar classes de problemas com dificuldade cada vez maior, ficam definidas *hierarquias de complexidade*. Nesse sentido, focaliza-se sobre certos aspectos da Teoria da Complexidade que podem ser chamados de "qualitativos" ou "estruturais". A Complexidade Estrutural é fortemente influenciada pela teoria da recursão. Conforme Garey e Jonhson [GAR 79], a assim chamada *hierarquia polinomial*, definida por Meyer e Stockmeyer [MEY 72] em analogia à hierarquia aritmética proposta anteriormente por Kleene [KLE 36], fornece uma forma muito mais detalhada de classificação de problemas de decisão NP-difíceis e, principalmente, estende a classificação para outros tipos de problemas, além daqueles de decisão.

Considerando NPO a classe de problemas de otimização, cujo problema de decisão associado está na classe NP, depara-se naturalmente com uma maior dificuldade na identificação das classes de complexidade. Enquanto um problema de decisão é classificado como NP-completo, seu correspondente problema de otimização pode apresentar características muito distintas em relação ao grau de aproximabilidade, podendo ser classificado como altamente aproximável até ser provado impossível aproximar, sob a conjetura P≠NP[CRE 91].

Estas peculiaridades relativas à questão da aproximabilidade de problemas de otimização são o tema de interesse deste trabalho, onde se procura transcender à diversidade

característica de soluções individualizadas, muito comuns nas áreas de heurísticas e de algoritmos aproximativos, buscando-se uma fundamentação teórica adequada, que privilegie as propriedades qualitativas desses problemas.

A relevância do tema é muito bem expressada no artigo de A. Schulz, D. Shmoys e D. Williamson [SCH 97], onde eles afirmam que a crescente competição global, rápidas mudanças de mercado e consumidores cada vez mais conscienciosos, têm alterado o modo através do qual as corporações fazem negócios. Em vista disso, para se tornarem mais eficientes, muitas indústrias têm procurado modelar alguns aspectos operacionais através de problemas de otimização, que resultam ser gigantescos. Infelizmente, para a maioria destes modelos, não são conhecidos algoritmos que encontrem soluções ótimas num tempo de computação razoável e, frequentemente, as indústrias precisam confiar em soluções de qualidade não garantida, que são construídas de uma maneira "ad hoc". No entanto, para alguns desses modelos, felizmente, existem bons algoritmos aproximativos, que se caracterizam por obter soluções próximas da solução ótima e de forma rápida. Justifica-se, assim, porque esta área tem sido uma das mais florescentes dentro da Matemática Discreta e Teoria da Ciência da Computação.

Sabe-se, no entanto, que a aplicabilidade de definições formais e resultados teóricos são freqüentemente questionados. Para responder a este tipo de questionamento veja-se, por exemplo, o impacto que o crescente progresso na área de Micro-Eletrônica tem trazido para a investigação teórica em Ciência da Computação. De acordo com Lengauer [LEN 90], novas tecnologias de fabricação que fornecem meios de produção de circuitos extremamente complexos, tornam o Desenho de Circuitos uma das áreas mais críticas na Ciência da Computação atualmente. A complexidade crescente neste meio impõe cada vez mais a necessidade de uma sistematização e de um perfeito enquadramento em termos de modelagem e formalização matemáticas.

De fato, considerando por exemplo as tecnologias usualmente designadas por VLSI (do inglês, *Very Large Scale Integration*, ou Integração em Escala Muito Grande), onde bem mais de 10^5 transístores são colocados numa única pastilha do computador (os *chips* - elementos que estão cada vez mais pequenos), tenta-se produzir circuitos tão densos e complexos que o seu desenho constitui um problema completo para a classe NP [LEN 90]. Na maioria das abordagens relacionadas ao Desenho de Circuitos são utilizadas técnicas e processos de resolução pertinentes às áreas de Teoria dos Grafos e Otimização Combinatória, cuja grande parte dos problemas significativos são NP-completos.

Ao se analisar a bibliografia relacionada ao tema "otimização", observa-se que, nos primeiros tempos da história da computação, grande parte da pesquisa restringia-se ao desenvolvimento de técnicas heurísticas para resolver problemas de otimização de uma forma aproximada, sendo que a noção precisa de aproximação com desempenho garantido só foi introduzida no início da década de setenta, com o trabalho seminal de D. Johnson [JOH 74].

Já em anos mais recentes, uma das visões prevalecentes tem sido a focalização sobre a estrutura das classes de problemas de otimização, com a definição de vários e diferentes tipos de reduções preservando aproximação e, principalmente, com a identificação de problemas completos, considerados "universais" dentro de cada classe. Conforme Blum et alli [BLU 98], não se conhecem algoritmos de tempo polinomial para estes problemas, e sua "universalidade" consiste no fato de que a existência de um tal algoritmo para um deles providenciaria algoritmos polinomiais para todos os problemas da respectiva classe.

A Teoria dos Algoritmos Aproximativos preocupa-se com a localização de problemas nestas classes, provando limites superiores de aproximação – isto é, a existência de um algoritmo aproximativo ou esquema aproximativo – e limites inferiores, mostrando que um problema não é aproximável satisfazendo uma determinada exatidão.

O interesse da atividade de pesquisa direcionada para entender a rica hierarquia entre os problemas aproximáveis e seus graus de aproximabilidade é atestada pela grande quantidade de publicações, citando-se, por exemplo, [PAP 91], [BOV 94], [BAL 95], [CRE 96], [TRE 97] e [AUS 99]. Uma leitura relevante é o artigo devido a V. Kann [KAN 97], onde são apresentadas e comentadas as mais importantes referências sobre o assunto.

O estado da arte nesta área pode ser descrito pela obtenção de importantes resultados relacionados com a prova da não-aproximabilidade de problemas, os quais vêm sendo obtidos, principalmente, através de técnicas envolvendo a noção de redutibilidade preservando aproximação, como também da utilização de sistemas de provas checáveis probabilisticamente ([ARO 94], [HAS 94], [SHM 95]).

A relevância da pesquisa na área, vem do fato de existirem muitos problemas em aberto, noções não totalmente compreendidas, conceitos estabelecidos em cima de simples conjeturas, atraindo o interesse de muitos pesquisadores motivados pela consciência de estarem participando da construção de uma das mais importantes e promissoras áreas de conhecimento da atualidade. Seguramente, uma das questões mais intrigantes e que permanece em aberto na Teoria da Complexidade, apesar das constantes tentativas dos cientistas da área, é a questão “P = NP ?”. É pouco provável que P = NP, já que NP contém diversos problemas notoriamente difíceis, para os quais, apesar do considerável esforço, não foram ainda encontrados algoritmos polinomiais.

Uma interessante discussão sobre esta questão, sob o ponto de vista de S. Cook [COO 00], aparece em publicação eletrônica do *Clay Mathematics Institute* [www.claymath.org/prize_problems], onde é apresentada uma descrição matemática do problema “P versus NP”, considerado um dos mais importantes problemas de investigação para o terceiro milênio. Segundo Cook, a importância desta questão deve-se parcialmente ao sucesso da teoria da NP-completude, como também do interesse da criptografia, já que a existência de um algoritmo eficiente para um problema NP-completo resultaria, por um lado, na solução de muitos problemas práticos encontrados na indústria, mas por outro lado, destruiria a segurança de transações financeiras e outras transações usadas amplamente através da internet.

A significância do problema referido acima é ainda reforçado no artigo sobre Complexidade Computacional, devido a Goldreich [GOL 2001], que integra o livro “*Mathematics Unlimited – 2001 and beyond*”, cujo objetivo é traçar um esboço do futuro da Matemática para o novo milênio, incluindo os pensamentos, opiniões e previsões dos mais importantes cientistas em Matemática (e áreas afim) da atualidade.

Motivados em parte por este interesse, procede-se neste trabalho a uma abordagem semântica para a Teoria dos Algoritmos Aproximativos, investindo-se naqueles aspectos da Teoria da Complexidade que podem ser chamados de “qualitativos” ou “estruturais”.

Uma síntese das principais idéias desenvolvidas neste trabalho é apresentada a seguir.

1.2 Tese e Objetivos

Duas principais questões nortearam o desenvolvimento deste trabalho:

- Como podem ser caracterizados qualitativamente aqueles algoritmos que, garantidamente, fornecem soluções aproximadas para problemas de otimização NP-difíceis ?

- Como se modelam as relações entre problemas que apresentam diferenciados graus de aproximabilidade, com o objetivo de entender a rica hierarquia entre problemas aproximáveis ?

Em relação à primeira questão, buscou-se na Teoria dos Domínios de Scott ([SCO 82], [ABR 87] e [STO 94]) o modelo semântico fornecendo uma fundamentação matemática segura e unificada para tratar a questão da aproximabilidade de problemas de otimização e, principalmente, privilegiando as propriedades qualitativas desses problemas.

Quanto à segunda questão, o reconhecimento de que a noção de redutibilidade entre problemas constitui-se num processo de especialização da Teoria das Categorias ([MAC 71], [BAR 90], [ADÁ 90] e [ASP 91]), conduziu a investigação das relações entre as classes de aproximação para uma abordagem de natureza categorial, focalizando sobre as reduções preservando aproximação entre problemas de otimização.

Com este objetivo em mente, procede-se a uma abordagem semântica para a Teoria dos Algoritmos Aproximativos em dois sentidos. Por um lado, considera-se um algoritmo aproximativo para um problema de otimização genérico como o principal objeto de estudo, estruturando-se matematicamente o conjunto de algoritmos aproximativos para tal problema como uma ordem parcial dentro da Teoria dos Domínios ([LEA 97a], [LEA 97b]). Por outro lado, focaliza-se sobre as reduções entre problemas de otimização, consideradas como morfismos numa abordagem dentro da Teoria das Categorias, onde problemas de otimização e problemas aproximáveis são os objetos das novas categorias introduzidas.

Cada uma destas estruturas matemáticas (Teoria dos Domínios e Teoria das Categorias) fornecem fundamentação teórica em aspectos que se complementam. A primeira providencia uma estruturação *interna* para os objetos, caracterizando as classes de problemas em relação às propriedades de aproximabilidade dos seus membros, no contexto da Teoria dos Domínios, enquanto que a segunda caracteriza-se por relacionar os objetos entre si em termos das reduções preservando aproximação entre problemas, num ponto de vista *externo*, essencialmente categorial. Dentro de cada abordagem, procura-se identificar aqueles elementos particulares, ditos “universais”, tais como elementos finitos, objetos totais, objetos iniciais e terminais, além de outras construções significativas. Desta maneira, espera-se aprofundar o entendimento de muitos processos dentro da Teoria dos Algoritmos Aproximativos que permanecem insuficientemente explicados.

O principal objetivo deste trabalho é desenvolver uma teoria formal para algoritmos aproximativos, considerados como uma alternativa viável para problemas intratáveis, de tal forma que integre as concepções da Complexidade Estrutural aos fundamentos de estruturas semânticas adequadas.

Dentro de uma proposta de investigação de interesse essencialmente teórico, pretende-se:

- Desenvolver uma fundamentação matemática segura e unificada para tratar a questão da aproximabilidade de problemas de otimização, privilegiando as propriedades qualitativas desses problemas.
- Apresentar novas perspectivas para velhos conceitos, através da busca por modelos adequados, num processo de *especialização* – no sentido do reconhecimento de que uma estrutura particular é uma instância de um fenômeno mais geral.
- Proceder à formalização teórica da Teoria dos Algoritmos Aproximativos, focalizando sobre aspectos relacionados à Complexidade Estrutural.
- Apresentar uma interpretação da hierarquia das classes de aproximação, a partir dos modelos semânticos que fundamentam a Teoria dos Algoritmos Aproximativos.

1.3 Trabalhos Relacionados

A solução de problemas e desenvolvimento de algoritmos é uma área que há muito vem sendo pesquisada no PPGC/UFRGS. O primeiro trabalho, devido a Toscani [TOS 88], deu ênfase à formalização e comparação de métodos de desenvolvimento de algoritmos e análise da complexidade dos algoritmos gerados (algoritmos recursivos e não recursivos), tanto para ambientes seqüenciais quanto paralelos. Esse trabalho teve continuidade com a definição de metodologias para o cálculo de complexidade no pior caso [TOS 90] e caso médio [ROS 97], sendo também estendidas para o cálculo da complexidade exata de algoritmos recursivos [LOR 00]. Foram ainda desenvolvidas ferramentas computacionais para implementação de tais metodologias [BAR 01].

O conhecimento de que, quando se tratam problemas NP-difíceis, outras estratégias de solução são necessárias e, em muitos casos, uma solução relativamente próxima da solução exata pode ser satisfatória, direcionou a maior parte da pesquisa para a área de Algoritmos Aproximativos e Heurísticas.

Entre as heurísticas estudadas, estão os Algoritmos Genéticos [AGU 98], Simulated Annealing [IZQ 98] e GRASP [MOR 2000]], sendo focalizados, principalmente, os aspectos formais da complexidade computacional, numa abordagem axiomática. Também foram desenvolvidas novas estratégias de solução de problemas clássicos (problemas de sequenciamento), baseados em busca tabu, com especial atenção à questão da complexidade [ROG 99].

No entanto, acredita-se que ainda existem relativamente poucos trabalhos unindo a Teoria da Complexidade e outras áreas da Ciência da Computação. Em seu trabalho de tese, D. Gurr [GUR 91] aplicou técnicas de semântica denotacional para estudar o comportamento dos requerimentos de recursos computacionais de um programa, tais como a complexidade de tempo e de espaço, construindo uma classe de categorias que modelam programas com as respectivas complexidades. Na direção de dar uma interpretação semântica para a Teoria dos Algoritmos Aproximativos, M. Huth [HUT 98] vem desenvolvendo uma caracterização de problemas de decisão com base na Teoria dos Domínios, onde problemas de decisão seriam objetos totais aproximados por algoritmos que consideram a resposta “não sei”, além de “sim”

e “não”. Resultados de não-aproximabilidade têm sido obtidos através de técnicas algébricas derivadas de provas interativas e checagem de programas [ARO 94], além daquelas fornecidas por reduções preservando aproximação entre problemas de otimização [TRE 97].

No contexto de sistemas complexos, a obtenção de um sistema de aproximação baseado em Teoria Categorial da Forma (*Categorical Shape Theory*), tem sido o objeto de interesse da pesquisa de Rattray ([RAT 94], [RAT 96]], [RAT 98]). A idéia básica é que, num sistema complexo, as aproximações são os únicos objetos que codificam a informação que o sistema pode analisar. A identificação e o reconhecimento de um objeto de interesse, visto como uma informação desejada, requer a identificação de um modelo (ou arquétipo) que melhor aproxime esse objeto. A definição de categorias de aproximação para cada objeto de interesse modela a estrutura hierárquica de estados do sistema.

Aproximação é considerada uma noção ubíqua, naturalmente relacionada com a idéia de computação e tendo significados distintos para pessoas distintas. Aqui, o interesse está direcionado ao estudo matemático de aproximação para problemas de otimização NP-difíceis.

Pelo exposto, pretende-se com o trabalho ora proposto, diminuir a lacuna existente, abrindo-se, quem sabe, um novo campo de investigação.

Na seção seguinte são apresentadas as delimitações para o escopo do trabalho, definindo-se objetivamente a área de investigação dentro da Complexidade Computacional.

1.4 Delimitações do Trabalho

Este trabalho visa apresentar uma abordagem da Teoria da Complexidade Computacional, que, segundo Balcázar et al. [BAL 95], em anos recentes vem sendo chamada de *Complexidade Estrutural*.

Conforme os autores, o principal objeto de interesse da Complexidade Computacional refere-se ao estudo dos requerimentos de recursos computacionais dos algoritmos que resolvem problemas computacionais. Por um lado, abordagens que privilegiam as medidas de tais recursos, fornecem resultados *quantitativos*. É o caso, por exemplo, da complexidade “concreta”, que pode ser descrita como um conjunto sistemático de técnicas para avaliação de recursos, principalmente tempo e espaço, requeridos por algoritmos “concretos”. A complexidade pessimista (pior caso) e a complexidade média são as medidas de complexidade “concreta” mais usadas.

Ainda, relacionado à complexidade “concreta”, mas requerendo um nível mais alto de abstração, está a questão de encontrar limites superiores e inferiores para os recursos requeridos na resolução de algum problema particular. Em caso de sucesso, esta abordagem permite provar a existência de algum algoritmo ótimo para o problema, associando seus requerimentos de recurso com o limite inferior. No entanto, provar tal limite inferior requer a abstração de um algoritmo particular, encontrando uma característica comum a todos os algoritmos que resolvem o problema.

Naturalmente, se modelos abstratos de algoritmos podem ser usados, então suas relações devem ser estabelecidas, comparando seu poder computacional em termos do crescimento dos recursos requeridos para simular um modelo através de outro. Esta é uma

área de pesquisa na qual mais e mais sofisticados tipos de simulações entre diferentes modelos vêm sendo projetados.

Na direção desta última abordagem, problemas são entes abstratos no contexto da Complexidade Estrutural, não sendo considerados problemas particulares. O alto nível de abstração é uma característica da abordagem estrutural da complexidade. Observa-se que medir recursos é uma tarefa fundamental quando se trata de complexidade. Entretanto, o interesse da Complexidade Estrutural é transcender de resultados de caráter quantitativos, questionando sobre as propriedades inerentes dos problemas a serem resolvidos num sentido que pode ser chamado de *qualitativo.* Uma forma de realizar esta abordagem, seria procurando por estruturas matemáticas bem conhecidas, presentes nos problemas ou em classes de problemas.

A dificuldade em delimitar o escopo de um trabalho de investigação teórico é tanto maior quanto maior é a abstração do tema proposto. Por esta razão, neste trabalho definimos como meta inicial:

1. Estudo de problemas de otimização pertencentes à classe NPO;
2. Complexidade de tempo, no pior caso;
3. Algoritmos aproximativos que fornecem uma solução garantidamente próxima da solução ótima, para todas as instâncias do problema;
4. Definição abstrata para redução entre problemas de otimização e redução-preservando-aproximação.

Assim, para um tratamento abstrato, problemas de otimização para os quais não se conhecem algoritmos não-determinísticos de tempo polinomial, no pior caso, estão fora do escopo de investigação. Da mesma forma, os algoritmos devem ter qualidade de aproximação garantida, não sendo consideradas algoritmos do tipo probabilístico ou heurísticos de qualquer natureza. Quanto às reduções definidas entre problemas de otimização, propõe-se o uso do conceito abstrato de redução entre problemas de otimização, em razão da existência de uma grande diversidade nas definições encontradas na literatura, especialmente em relação às reduções-preservando-aproximação [TER 97].

Finalmente, devido à orientação geral do trabalho, resolveu-se dar mais ênfase aos conceitos e resultados importantes relativos aos aspectos de fundamentação teórica, omitindo as respectivas demonstrações e ilustrações com casos e exemplos, os quais foram substituídos por breves indicações da idéia geral e referências bibliográficas.

1.5 Estrutura do Trabalho

Os demais capítulos deste trabalho observam a seguinte estrutura:

- O Capítulo 2 apresenta os fundamentos teóricos da Complexidade Computacional de forma concisa seguindo a literatura clássica e introduzindo uma breve discussão histórica de modo a caracterizar o estado da arte na área. O objetivo deste capítulo consiste na apresentação dos resultados e conceitos necessários para o desenvolvimento dos capítulos seguintes, bem como a nomenclatura e a notação utilizadas;

- O Capítulo 3 apresenta um estudo específico sobre a complexidade de problemas de otimização. A focalização sobre as questões relativas à aproximabilidade de problemas de otimização e a correspondente estruturação em classes de aproximação, visam a efetiva compreensão dos aspectos semânticos abordados no restante do trabalho. Noções tais como a medida de qualidade de um algoritmo e redução entre problemas desempenham papéis centrais nas estruturas matemáticas propostas;

- O Capítulo 4 apresenta a Teoria dos Domínios de Scott como o modelo matemático que estrutura internamente o conjunto dos algoritmos aproximativos para um dado problema de otimização. O capítulo inicia com um estudo dos principais aspectos desenvolvidos na Teoria dos Domínios, a seguir define o conjunto dos algoritmos aproximativos para um problema de otimização como uma estrutura ordenada, tanto em termos de um requerimento de exatidão quanto em termos da complexidade computacional. Finalmente são analisadas as propriedades desta estrutura de forma a caracterizá-la dentro da Teoria dos Domínios;

- O Capítulo 5 apresenta uma abordagem categorial para problemas de otimização. A noção de redução de um problema para outro aparece, naturalmente no sentido conceitual de morfismo entre dois objetos. São definidas duas novas categorias de problemas de otimização: a categoria OPTS, cujos objetos são problemas de otimização resolvíveis polinomialmente e a categoria OPT, tendo por objetos os problemas de otimização da classe NPO. A introdução de uma teoria de universais permite formalizar a noção de um problema completo para uma classe, visto como um objeto universal concreto. Finalmente, é apresentada uma definição funcional para problemas de otimização, possibilitando uma representação gráfica e motivando para algumas investigações de abordagem categorial;

- O Capítulo 6 apresenta inicialmente uma discussão com respeito ao modelo categorial da teoria da forma (*categorical shape theory*). Em seguida apresenta as conexões entre as categorias dos problemas de otimização OPTS e OPT e a teoria categorial da forma. Baseado nessa teoria, é construída uma categoria de aproximações para cada problema de otimização;

- O Capítulo 7 apresenta o estudo da hierarquia polinomial estendida para problemas que estão fora da classe NP. Nesse contexto, são colocadas questões relativas ao caso específico das hierarquias de aproximação para problemas de otimização;

- O Capítulo 8 apresenta as conclusões do trabalho, juntamente com indicações de desenvolvimentos futuros.

O volume encerra-se com a apresentação de uma seção de anexos, contendo a descrição dos problemas de otimização citados como exemplos e um roteiro dos principais artigos publicados durante a realização do presente trabalho. Por último, são apresentadas as referências bibliográficas utilizadas para o desenvolvimento do trabalho.

1.6 Comentários Finais

Este capítulo apresentou o tema sobre o qual é formulada a tese proposta, salientando a motivação e a justificativa para sua escolha, bem como aspectos gerais da estruturação e do desenvolvimento deste trabalho. Estes envolvem a descrição dos objetivos propostos, das delimitações do trabalho, dos trabalhos relacionados e da estrutura deste documento.

O próximo capítulo consiste dos fundamentos teóricos sobre o qual o presente trabalho está ancorado, servindo de partida para a discussão técnica dos aspectos considerados relevantes para o desenvolvimento desta tese.

2 Fundamentos Teóricos da Complexidade Computacional

O propósito deste capítulo é apresentar de forma concisa os resultados classicamente conhecidos como mais significativos dentro da Teoria da Ciência da Computação, os quais formam um conjunto mínimo de informações necessárias para os desenvolvimentos apresentados nos capítulos seguintes. Tal apresentação será relevante para a efetiva compreensão dos aspectos semânticos abordados, além de caracterizar o estado da arte nesta área de pesquisa.

2.1 Breve Histórico da Complexidade Computacional

Na década de trinta, antes do advento dos computadores, matemáticos dedicavam-se ativamente ao estudo e formalização da noção de algoritmo, que era intuitivamente interpretado como uma seqüência de instruções simples claramente especificado para ser seguido na resolução de um problema (ou computação de uma função). Vários modelos formais de computação foram criados e investigados [COT 90]. A ênfase nos trabalhos iniciais nesta área, que ficou conhecida como *Teoria da Computabilidade*, foi a descrição e caracterização daqueles problemas que poderiam ser resolvidos algoritmicamente e a exibição de algum problema que não o pudesse, além de, obviamente, conceituar formalmente a noção de algoritmo. Um dos resultados mais importantes foi a prova da insolubilidade do "Problema da Parada" (*halting problem*). Como se sabe, o "Problema da Parada" consiste em determinar se um dado algoritmo arbitrário (ou programa computacional) entrará num *loop* infinito enquanto trabalha sobre uma dada entrada. Deve-se ao matemático britânico Alan Turing [TUR36] a prova de que não existe um programa computacional que resolva este problema.

Mas, embora a Teoria da Computabilidade tenha implicações óbvias e fundamentais para a Ciência da Computação, o conhecimento de que um problema pode teoricamente ser resolvido num computador não é suficiente para dizer se é possível resolvê-lo na prática. Conforme exemplifica Baase [BAA 78], teoricamente um programa de jogo de xadrez poderia ser escrito perfeitamente, o que de forma alguma seria uma tarefa muito difícil, já que existe somente um número finito de maneiras de dispor as peças no tabuleiro e, sob certas regras, uma partida deve terminar depois de um número finito de movimentos. O programa consideraria cada um dos possíveis movimentos do computador, cada uma das possíveis respostas do adversário, cada uma das possíveis respostas para estes movimentos e assim por diante, até que cada sequência de movimentos possíveis atingisse um final. Então, já que o computador conhece os resultados ótimos de cada movimento, basta que escolha o melhor deles e o problema está resolvido. No entanto, tal programa não pode realisticamente ser executado pois o número de seqüências de movimentos razoáveis (estimado em cerca de 10^{19}) é tão grande que um programa que o simulasse levaria vários milhares de anos para executá-lo.

Assim como o exemplo descrito acima, existem numerosos problemas com aplicações práticas que podem ser resolvidos, isto é, para os quais podem ser escritos programas, mas os

requerimentos de tempo e espaço de memória são muito grandes para que estes programas sejam considerados de uso prático.

Claramente os requerimentos de tempo e espaço são de grande importância prática, de tal forma que tornaram-se o objeto de estudo de outra importante área da Ciência da Computação conhecida por *Teoria da Complexidade Computacional*. Conforme estabelece Papadimitriou [PAP 94], a *complexidade* é uma intrincada e ao mesmo tempo requintada interação entre *computação* (classes de complexidade) e *aplicações* (problemas), fundamentando as razões porque alguns problemas são tão difíceis de resolver através de computadores.

Na concepção de Bovet [BOV 94], a Teoria da Complexidade pode ser considerada como um refinamento da Teoria da Computabilidade. O nascimento da área conhecida por Complexidade Computacional teve seu início na década de sessenta, quando os primeiros usuários de computadores eletrônicos começaram a dedicar especial atenção às performances de seus programas.

Como na Teoria da Computabilidade, onde o conceito de um modelo de computação levou aos conceitos de algoritmo e problema algoritmicamente resolvível, de modo similar, na Complexidade Computacional o conceito de recurso usado por uma computação levou aos conceitos de algoritmo eficiente e de problema computacionalmente admissível.

Em seu excelente artigo “An overview of Computational Complexity”, S. Cook [COO 83] apresenta um histórico do desenvolvimento da teoria da complexidade, onde menciona os mais interessantes e importantes resultados, desde o início da área da Complexidade Computacional. Sob o seu ponto de vista, a identificação da classe de problemas resolvíveis em tempo limitado por um polinômio no comprimento da entrada, em 1965, foi um dos mais importantes conceitos introduzidos nesta área, sendo sua formalização determinante para o desenvolvimento da teoria da complexidade. A distinção entre algoritmo de tempo polinomial e de tempo exponencial já era considerada por Von Neumann, por volta da década de cinquenta, entretanto a classe de problemas polinomialmente limitados só foi estudada formalmente por Cobham cerca de dez anos depois, mostrando que a classe estava bem definida e era independente do modelo computacional escolhido. A idéia que a complexidade de tempo polinomial grosseiramente corresponde a *tratabilidade* de problemas foi primeiramente expressa por J. Edmonds, que chamou algoritmos de tempo polinomial de “bons algoritmos”. A notação atual P, identificando a classe de conjuntos de cadeias reconhecíveis em tempo polinomial por um modelo computacional, só foi introduzida por Karp em 1972.

Segundo Cook, o que torna a teoria da complexidade uma área à parte da análise de algoritmos, é a satisfação de provar limites inferiores sobre a complexidade de problemas específicos, ao mesmo tempo que importantes questões abertas na teoria continuam sendo um real desafio aos pesquisadores.

Nas seções seguintes são apresentados os elementos conceituais básicos sobre os quais está construída a Teoria da Complexidade Computacional.

2.2 Problema Computacional

Intuitivamente, computar significa: ser capaz de especificar uma sequência finita de ações que conduzem a um resultado. Mas o que se entende por *ação* ?

Conforme Bovet [BOV 94], uma resposta precisa para esta questão só pode ser dada depois de ter sido definido um modelo de computação, isto é, um sistema formal que permite expressar sem ambiguidades as ações a serem executadas. A introdução do conceito de um modelo de computação na Teoria da Computabilidade permitiu que fossem formalizadas matematicamente as noções de algoritmo e problema algoritmicamente solúvel. Muitos modelos de computação distintos, porém basicamente equivalentes, têm sido desenvolvidos por lógicos matemáticos. Observa-se que todos os esforços na tentativa de produzir um modelo computacional mais poderoso não tem obtido sucesso, nem tampouco foi possível provar que tal modelo não existe. Acrescente-se a isto a famosa conjetura conhecida por Tese de Church, que tem resistido aos ataques dos logicistas por cerca de 50 anos e estabelece que todo problema solúvel pode ser resolvido usando qualquer modelo de computação conhecido.

Pretende-se na presente seção que noções tais como problema, solução, algoritmo e complexidade recebam formulações matemáticas precisas. Neste contexto, questões relativas a estes conceitos podem então ser formuladas e examinadas de maneira rigorosa.

2.2.1 Problema

Informalmente um algoritmo pode ser definido como um método para resolver um problema detalhando etapa por etapa. Mas o que é um *problema* ? Em Teoria da Computabilidade, na concepção de Papadimitriou [PAP 94], problemas não são apenas coisas a resolver, mas também objetos matemáticos que têm interesse próprio.

Numa primeira abordagem, apresenta-se o conceito de um problema abstrato, formalizando de forma rigorosa todas as suas etapas, desde a formulação até a solução de um problema genérico.

Sob um outro enfoque, define-se um problema computacional em termos de linguagens reconhecidas por um modelo computacional dado por uma máquina de Turing, caracterizando-se problemas de decisão e problemas de otimização.

2.2.2 Problema Abstrato

Dentro de uma teoria geral de problemas, Veloso [VEL 84] define um problema abstrato com base nas sugestões dadas por Polya na obra intitulada ”How to solve it”, consistindo basicamente na formulação de três questões:

- Quais são os dados do problema ?
- Quais são os possíveis resultados do problema ?
- O que constitui uma solução satisfatória para o problema ?

Um *problema* é definido então como uma terna **p** = (D,R,q), onde D é o domínio de dados, R o domínio de resultados e q uma relação de D em R que define o problema. A

solução de um problema **p** = (D,R,q) é uma função $\alpha : D \rightarrow R$ tal que para todo $d \in D$, tem-se que o par $(d, \alpha(d)) \in q$.

Nem sempre é fácil encontrar um método de desenvolvimento algorítmico adequado para um dado problema. Muitas vezes é necessário modificar o problema, reduzindo-o a outros problemas com ele relacionados ou decompondo-o em partes menores. Estas duas técnicas de resolução de um problema são discutidas e apresentadas formalmente por Veloso [VEL 84]. Observa-se que a redução de um problema a outro não é um método de solução, mas sim de transformação do problema, sendo largamente utilizada na prática.

A idéia básica na redução de um problema a outro, é usar uma solução deste para obter uma solução para o outro. Isto pode ser feito transformando-se dados do primeiro problema em dados do segundo e transformando-se, de volta, resultados deste em resultados para aquele. Formalmente, uma *redução* foi definida da seguinte maneira: dados dois problemas **p** = (D,R,q) e **p'** = (D',R',q') , uma redução Γ de **p** em **p'** é um par (t,v) de funções tais que $t: D \rightarrow D'$ é dita função de tradução e $v: R \rightarrow R'$ é dita função de recuperação. Diz-se que Γ é uma boa redução se e somente se para qualquer solução $\alpha' : D' \rightarrow R'$ do problema **p'**, a função $\alpha : D \rightarrow R$ definida por $\alpha(d) = v(\alpha'(t(d)))$ é uma solução do problema **p**. A FIGURA abaixo ilustra o processo de redução de um problema.

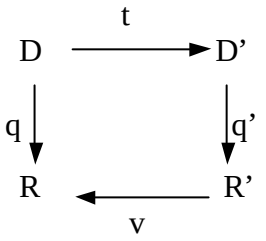

FIGURA 2.1 - Representação gráfica da redução de um problema

O *método da decomposição* para resolver um problema consiste, basicamente, em decompor repetidamente as instâncias do problema em instâncias menores, até que sejam simples o suficiente para possibilitar a obtenção direta de resultados. Tais resultados serão, então, sucessivamente combinados, de modo a se obter um resultado para a instância originaldo problema. Este processo é representado pela figura abaixo.

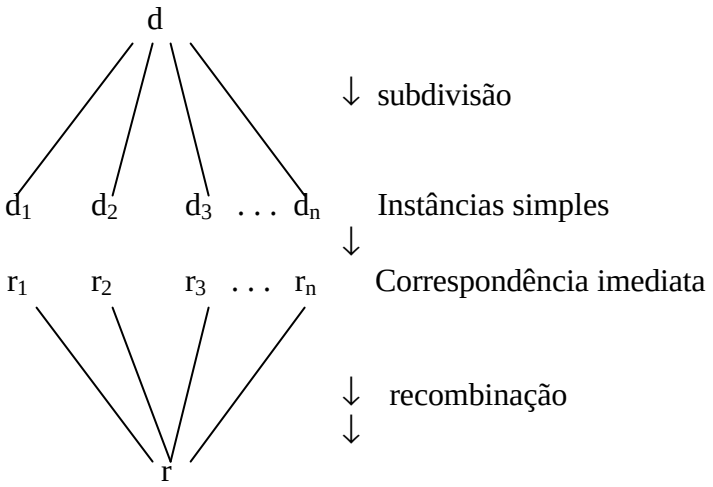

FIGURA 2.2 - Esquema da decomposição de um problema

Formalmente, a decomposição de um problema **p** = (D,R,q) é definida em termos de um decompositor n-ário Δ_n para **p** consistindo em:

- funções unárias $\pi_i : D \rightarrow D$, i=1,...,n, chamadas funções de subdivisão;
- uma função (n+1)-ária $\mu : DxR^n \rightarrow R$, dita função de recombinação;
- uma função unária $\nu : D \rightarrow R$, chamada correspondência imediata;
- uma relação unária η em D, chamada teste de simplicidade, e
- uma função $\sigma : D \rightarrow R$, $\sigma(d) = \begin{cases} \upsilon(d), & \text{se } d \in \eta \\ \mu(d, \sigma(\pi_1(d)), \ldots, \sigma(\pi_n(d))), & \text{cc} \end{cases}$

  chamada função de atribuição.

A função de atribuição σ sugere que a função ν de correspondência imediata pode ser encarada como uma "solução parcial", no sentido que ela realmente resolve o problema **p**, mas apenas para os dados que passam no teste de simplicidade η. Desta maneira, pode-se dizer que a tarefa do decompositor é estender esta "solução parcial" a uma "solução total", por meio de subdivisões e recombinações, conforme é ilustrado na Figura 1.2 acima.

2.2.3 Problema Computacional

Sob um ponto de vista mais computacional, Garey e Johnson [GAR 79] descrevem informalmente um problema como uma questão genérica a ser respondida, geralmente possuindo vários parâmetros, ou variáveis livres, cujos valores não são especificados. Um problema é então descrito através de:

(1) uma descrição genérica de todos os seus parâmetros, e

(2) uma declaração de quais propriedades a resposta ou solução, deve satisfazer.

Uma *instância* de um problema é obtida através da especificação de valores particulares para todos os parâmetros do problema. A descrição da instância de um problema, fornecida como entrada (*input*) para o computador, pode ser vista como uma simples cadeia (*string*) finita de símbolos escolhidos de um alfabeto finito. Observa-se que, embora existam muitas maneiras diferentes de descrever as instâncias de um problema, considera-se uma em particular e associa-se a cada problema um esquema de codificação fixo, o qual mapeia cada instância do problema na cadeia que a descreve.

O comprimento de entrada para uma instância I de um problema p é definida pelo nº de símbolos na descrição de I obtida do esquema de codificação do problema, sendo usado como uma medida formal do tamanho da instância. Naturalmente este parâmetro é importante, pois é em relação a ele que são definidas as medidas de complexidade, sendo por esse motivo inaceitáveis as codificações de instâncias que sejam desnecessariamente longas, como por exemplo o uso do sistema unário para codificação de numeros inteiros.

Existem certas classes gerais de problemas algorítmicos, conforme é apresentado por Szwarcfiter em [SZW 84]: os *Problemas de Decisão*, os *de Localização* e os *de Otimização*. Num problema de decisão, o objetivo consiste em decidir a resposta *sim* ou *não* a uma questão. Num problema de localização, o objetivo é localizar uma certa estrutura que satisfaça

um conjunto de propriedades dadas. Se as propriedades que esta estrutura deve satisfazer ainda envolverem critérios de otimização, então o problema é dito de otimização.

Observa-se que é possível associar problemas de decisão, otimização e localização, de tal forma que um problema possa ser derivado do outro. Exemplificando: se um problema de otimização pede por uma estrutura de um certo tipo que tem custo mínimo entre todas as outras, pode-se associa-lo com aquele problema de decisão que inclui um limite numérico B como um parâmetro adicional e que pergunta se existe uma estrutura do tipo requerido que tenha um custo não maior que B. De forma análoga, um problema de decisão pode ser derivado de problemas de otimização simplesmente trocando os termos 'não maior que' por 'no mínimo' (ou 'no máximo').
O ponto chave a ser observado nesta correspondência é que os problemas de otimização e localização apresentam ambos dificuldade não maior do que o problema de decisão associado.

2.2.4 Problemas de Decisão

Por uma questão de conveniência a teoria da complexidade restringe-se a problemas de decisão, já que o estudo de problemas NP-completos é aplicado somente para este tipo de problema. Como foi visto, tais problemas têm somente duas possíveis soluções: a resposta 'sim' ou a resposta 'não'. Em termos abstratos, um problema de decisão Π consiste de um conjunto de instâncias D_Π e um subconjunto $Y_\Pi \subseteq D_\Pi$ de instâncias com resposta 'sim'.

Conforme Garey e Johnson [GAR 79], a razão para esta restrição a problemas de decisão é justificada porque estes problemas têm uma contrapartida formal muito natural, que é um objeto adequado para o estudo dentro de uma teoria de computação matematicamente precisa. Esta contrapartida é chamada uma *linguagem* e é definida a seguir.

Para qualquer conjunto finito Σ de símbolos, denota-se por Σ^* o conjunto de todas as palavras finitas de símbolos de Σ, inclusive a palavra vazia. Um subconjunto L de Σ^* é chamada uma *linguagem* sobre o alfabeto Σ.

Dentro deste contexto, um problema pode ser definido formalmente como uma relação total sobre palavras do alfabeto $\Sigma = \{0,1\}$, isto é, sobre sequências finitas de 0's e 1's pertencentes ao conjunto Σ^* de todas as palavras sobre o alfabeto Σ. Esta definição é apresentada de forma mais precisa por Johnson [JOH 90]: "um problema é um conjunto X de pares ordenados (I, A) de palavras em Σ^*, onde I é uma instância do problema e A é uma solução para esta instância."

É claro que qualquer problema que se pretenda resolver por meios computacionais pode ser descrito utilizando este formalismo, pois uma vez que a memória de um computador pode ser entendida como uma série de palavras do alfabeto Σ, naturalmente pode-se representar instâncias e soluções do problema como palavras deste mesmo alfabeto.

A correspondência entre problemas de decisão e linguagens é colocada em termos do esquema de codificação usado para especificar as instâncias de um problema, sempre que se pretende computar com elas. Portanto o problema Π e seu respectivo esquema de codificação particiona Σ^* em três classes de palavras: aquelas que não são codificações de instâncias do problema Π; aquelas que codificam instâncias de Π para as quais a resposta é 'não' e aquelas que codificam instâncias para as quais a resposta é 'sim'. Esta terceira classe de palavras é a linguagem que é associada ao problema de decisão Π.

Simbolicamente, a linguagem L associada ao problema de decisão Π é representada pelo conjunto:

$L(\Pi) = \{x \in \Sigma^* | \Sigma$ é o alfabeto de codificação e x é a codificação de uma instância $I \in Y_\Pi\}$

A representação de problemas de decisão como linguagens, possibilita a identificação de 'resolver' um problema de decisão com 'reconhecer' a linguagem correspondente.

2.2.5 Problemas de Otimização

Um problema de otimização combinatorial Π é definido por Garey & Johnson [GAR 79], como uma terna $\Pi = (D_\Pi, S_\Pi, m_\Pi)$, onde:

(i) D_Π é o conjunto das instâncias;

(ii) S_Π é a função que associa a cada instância $I \in D_\Pi$, um conjunto finito $S_\Pi(I)$ de candidatas a solução para I; e

(iii) m_Π é uma função que atribui a cada instância $I \in D_\Pi$ e a cada candidata $\sigma \in S_\Pi(I)$, um número racional positivo $m_\Pi(I, \sigma)$ chamado valor solução de σ.

Se Π é um problema de minimização [maximização], uma *solução ótima* para uma instância $I \in D_\Pi$ é uma candidata a solução $\sigma^* \in S_\Pi(I)$ tal que, para todo $\sigma \in S_\Pi(I)$, $m_\Pi(I, \sigma^*) \leq m_\Pi(I, \sigma)$ [$m_\Pi(I, \sigma^*) \geq m_\Pi(I, \sigma)$]. O valor $m_\Pi(I, \sigma^*)$ de uma solução ótima para I é denotado por $OPT_\Pi(I)$.

Os elementos básicos de um problema de otimização são os mesmos de um problema de decisão, acrescidos da função m cujo valor mede quão boa é uma solução admissível. O problema consiste em encontrar uma solução com uma medida máxima (no caso de um problema de maximização), ou uma medida mínima (no caso de um problema de minimização). Se tal medida pode assumir somente dois valores, por exemplo 0 e 1, então o problema de otimização torna-se essencialmente um problema de decisão. Em geral, entretanto, essa medida pode assumir valores arbitrários. Para o propósito deste trabalho, serão considerados somente problemas de otimização cujas medidas assumem valores inteiros positivos.

2.3 Algoritmos

Segundo Horowitz e Sahni [HOR 78], a palavra *algoritmo* adquiriu um significado especial na Ciência da Computação, sendo referido como um método preciso usado por um computador de modo a se obter a solução de um problema.

Um algoritmo é, em geral, uma descrição passo a passo de como um problema é solucionável. A descrição deve ser finita e os passos devem ser bem definidos, sem ambigüidades, além de executáveis computacionalmente. Diz-se que um algoritmo resolve um problema **p** se este algoritmo recebe qualquer instância I de **p** e sempre produz uma solução para esta instância I. Observa-se que, para qualquer instância I do problema **p** definida como uma entrada, o algoritmo deverá ser executável em tempo finito e, além disso, produzir como saída uma solução correta para o problema **p**.

O estudo de algoritmos inclui importantes áreas de pesquisa, tais como: projeto, validação, análise, codificação e verificação.

O ato de criar um algoritmo é uma arte que dificilmente poderá ser totalmente automatizada. Técnicas como programação linear, divisão e conquista e programação dinâmica são algumas das várias técnicas que têm sido criadas para projetar bons algoritmos.

A verificação algorítmica consiste em mostrar que o algoritmo computa a resposta correta para todas as possíveis entradas, enquanto que uma prova completa de correção e verificação do programa correspondente requer que cada declaração da linguagem de programação seja definida precisamente e que todas as operações básicas estejam comprovadamente corretas.

A área conhecida por Análise de Algoritmos é considerada um desafio aos projetistas de algoritmos, porque, em geral, requer grande habilidade matemática. A análise de um algoritmo refere-se ao processo de determinar os recursos computacionais dispendidos pelo algoritmo. Questões relativas à complexidade de um algoritmo em termos do tempo de computação e espaço de memória, são determinantes para o julgamento da eficiência do algoritmo. Com base neste estudo, é possível comparar algoritmos qualitativamente, além de analisar a performance de um algoritmo no melhor caso, no pior caso ou no caso médio.

Conforme ilustrado através de vários exemplos por Toscani e Veloso [TOC 2001], a criação de máquinas cada vez mais velozes, devido ao crescente avanço tecnológico, pode, aparentemente, ofuscar a importância da complexidade de tempo de um algoritmo. Entretanto, acontece exatamente o inverso. Enquanto as máquinas tornam-se mais rápidas, podendo resolver problemas de maior porte, é justamente através da complexidade do algoritmo que será determinado o tamanho máximo de problema resolvível.

Antes de seguir com a discussão de algoritmos e suas complexidades, que serão o objeto de interesse da próxima seção, precisa-se especificar um modelo de computação para executar um algoritmo e definir o que significa uma etapa básica de computação. Segundo Aho et alli [AHO 74], talvez a motivação mais importante para modelos formais de computação é o desejo de descobrir a dificuldade computacional inerente a vários problemas. Para mostrar que não existe um algoritmo que execute uma dada tarefa dentro de um tempo estipulado, é necessário definir precisamente o que constitui um algoritmo. Máquinas de Turing são um exemplo de tal definição.

Na presente seção, procede-se à formalização do conceito de algoritmo usando o modelo de computação formal de uma máquina de Turing, de forma a caracterizar determinismo, não-determinismo e paralelismo para um algoritmo. Basicamente, a referência utilizada nesta parte é [GAR 79].

2.3.1 Algoritmos Determinísticos

O modelo computacional conhecido por *máquina de Turing*, foi proposto pelo matemático britânico Alan Turing, em 1936, consistindo de uma 'máquina' que efetua um procedimento mecânico de prova.

Para formalizar a noção de um *algoritmo determinístico*, considere-se o modelo de Máquina de Turing Determinística (MTD) apresentado esquematicamente na Figura 2.3, consistindo de um controle de estados finitos, uma cabeça de leitura e escrita e uma fita com

uma seqüência infinita de celas para os dois lados, indicadas por ...-2,-1,0,1,2,3,... .

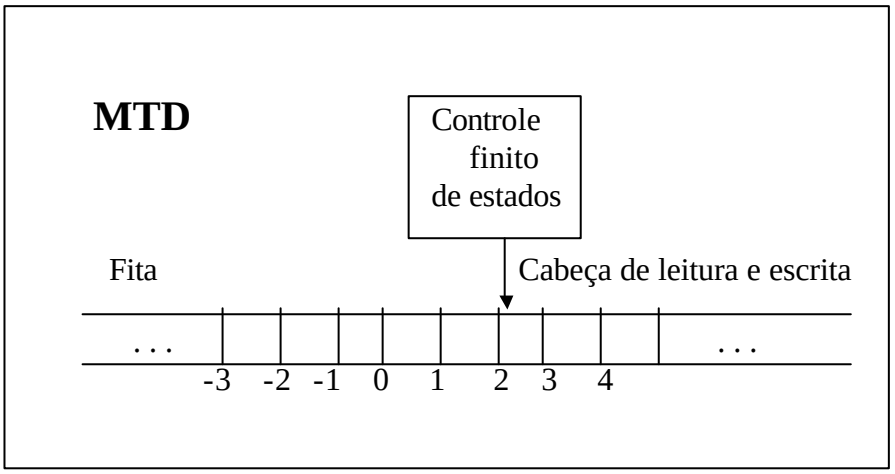

FIGURA 2.3 - Representação de uma Máquina de Turing Determinística (MTD)

Um *programa* para uma MTD é uma quádrupla $M = (Q, \Gamma, \Sigma, \delta)$, tal que:

(i) Γ é o *alfabeto* finito da fita; $\Sigma \subseteq \Gamma$ é o *alfabeto das entradas*; $\text{ð} \in \Gamma\text{-}\Sigma$ é o símbolo distinguido *branco*;

(ii) Q é o conjunto finito de *estados*, incluindo um *estado inicial* q_0 e dois *estados finais* q_S (sim) e q_N (não);

(iii) $\delta : (Q\text{-}\{q_S,q_N\}) \times \Gamma \rightarrow Q \times \Gamma \times \{-1,+1\}$

Diz-se que uma MTD (ou um programa) $M = (Q, \Gamma, \Sigma, \delta)$, com alfabeto de entrada Σ, *aceita* a entrada $x \in \Sigma^*$, se e somente se M para no estado q_S(sim), quando aplicado à entrada x.

A *linguagem reconhecida* por M é dada por:

$$L_M = \{ x \in \Sigma^* \mid M \text{ aceita } x\}$$

É importante observar que nesta definição de linguagem reconhecida pela MTD não é exigido que o programa pare, para toda entrada $x \in \Sigma^*$, somente para aquelas que pertencem a L_M. Se $x \in \Sigma^*\text{-}\ L_M$, então a computação de M sobre x pode parar no estado q_N ou pode continuar para sempre, sem parar. Entretanto para uma MTD corresponder à noção de *algoritmo*, é preciso que ela pare, para *todas* as possíveis entradas sobre seu alfabeto Σ.

A correspondência entre 'reconhecer' linguagens e 'resolver' problemas de decisão pode agora ser colocada diretamente:

Diz-se que uma MTD $M = (Q, \Gamma, \Sigma, \delta)$ resolve o problema de decisão Π, se M pára, para todas as entradas sobre seu alfabeto e $L_M = L(\Pi)$.

2.3.2 Algoritmos Não-Determinísticos

Uma máquina de Turing MTD é dita *determinística* pelo fato que cada computação pode ser vista como uma sequência (finita ou infinita) de estados, cujo primeiro elemento é o estado inicial que tem uma única possibilidade para levar ao estado seguinte e assim sucessivamente.

Já uma máquina de Turing é chamada *não-determinística* (MTND) por não apresentar tal limitação. Na sequência de computação, a partir do estado inicial pode existir mais de uma possibilidade para o estado seguinte. Uma computação com característica não-determinística pode ser vista como uma árvore, chamada *árvore de computação*, cujos nodos correspondem aos estados e as arestas entre cada nodo corresponde às transições entre estados ocorrida numa simples etapa. Cada um dos (finito ou infinito) caminhos da árvore iniciando na raiz é dito ser um *caminho de computação*.

A figura abaixo mostra um exemplo de árvore de computação para uma MTND.

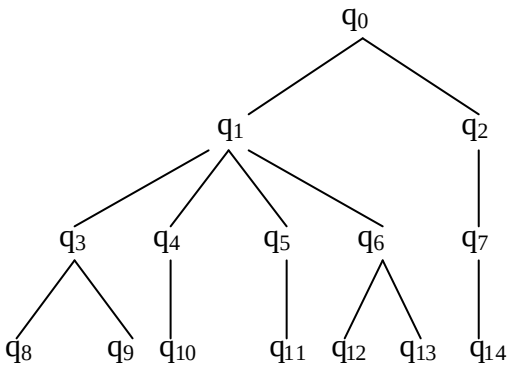

FIGURA 2.4 - Árvore de computação de uma MTND

De acordo com Garey & Johnson[GAR 79], algoritmos não-determinísticos são compostos de dois estágios separados: um primeiro estágio de conjetura (ou estimativa) e um segundo estágio de checagem (ou reconhecimento). Dada uma instância I de um problema, o primeiro estágio apenas conjetura alguma proposta de solução S. Para o estágio de checagem, ambos I e S são dados como entrada, e a computação prossegue de maneira determinística normal, parando com a resposta sim ou eventualmente com a resposta não, ou ainda sem nunca parar.

Um algoritmo não-determinístico resolve um problema de decisão $\prod$, se as seguintes propriedades são verificadas para toda instância $I \in D_{\prod}$:

(1) Se $I \in Y_{\prod}$, então existe alguma proposta S tal que, quando estimada pela entrada I, conduz o estágio de checagem à resposta 'sim', para I e S.

(2) Se $I \notin Y_{\prod}$, então não existe proposta S tal que, quando estimada pela entrada I, conduz o estágio de checagem à resposta 'não', para I e S, e o algoritmo para.

Diz-se que um algoritmo não-determinístico que resolve um problema de decisão Π opera em *tempo polinomial*, se existe um polinômio p tal que, para toda instância $I \in Y_\Pi$, existe alguma proposta de solução S que conduz o estágio de checagem determinístico à resposta 'sim' para I e S dentro de um tempo p(|I|). Observe que esta condição tem o efeito de impor um limite polinomial sobre o tamanho da solução proposta S, já que somente uma quantidade de tempo polinomialmente limitada pode ser dispendida ao examinar tal estimativa.

Um algoritmo não-determinístico é definido formalmente como um programa para uma máquina de Turing não-determinística. O modelo de MTND conforme descrito em [GAR 79], tem exatamente a mesma estrutura de uma MTD, exceto que é aumentada com um módulo de conjetura tendo seu próprio cabeçote somente de escrita, como é ilustrado na Figura 2.5. O módulo de conjetura garante o registro da estimativa e é usado unicamente para este propósito.

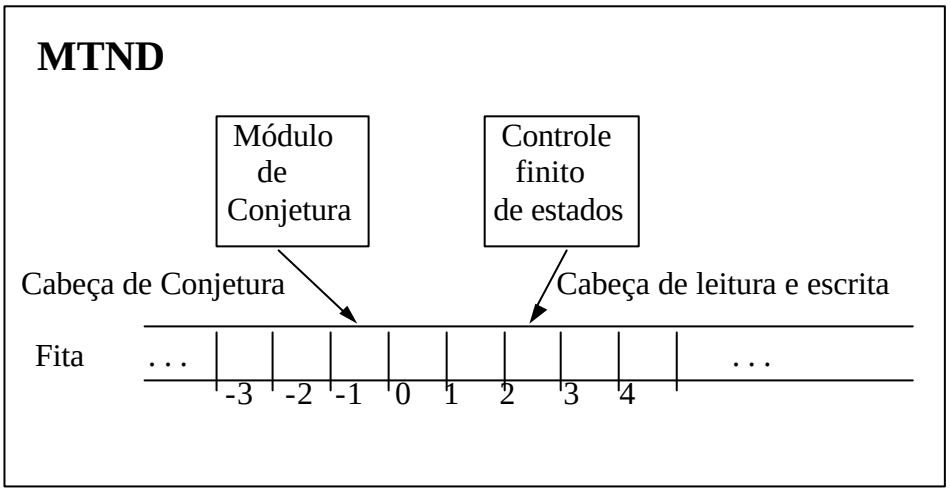

FIGURA 2.5 - Representação de uma Máquina de Turing Não-Determinística (MTND)

Observa-se que se M é uma MTND qualquer, haverá um número infinito de possíveis computações para uma dada entrada x, sendo uma computação para cada possível conjetura sobre o alfabeto Γ da fita. Diz-se que tal máquina M aceita x, se pelo menos uma destas é uma computação aceita, isto é, se uma entre todas as possíveis computações de M pára se atingir um estado de aceitação.

A linguagem reconhecida pela MTND M é dada por:

$$L_M = \{ x \in \Sigma^* \mid M \text{ aceita } x\}$$

Para obter a correpondência entre a noção de um algoritmo não-determinístico e o conceito formal de uma MTND, resta ainda a mencionar um ponto especial. Observa-se que num algoritmo não-determinístico foi considerada como conjetura uma proposta de solução S que de algum modo dependia da instância dada, enquanto que no módulo de conjetura de uma MTND a entrada foi inteiramente ignorada. Entretanto, já que cada palavra sobre o alfabeto

da fita é uma possível conjetura, pode-se sempre projetar a MTND de modo que o estágio de checagem comece reconhecendo se tal palavra proposta corresponde a uma conjetura apropriada para uma dada entrada. Caso contrário, o programa pode imediatamente entrar no estado de parada q_N.

2.3.3 Algoritmos Paralelos

Algoritmos paralelos são apresentados na literatura como uma alternativa para resolver problemas computacionais em muito grande escala, além de estarem ligados à concepção de algoritmos extremamente rápidos.

Apesar deste trabalho estar direcionado para o estudo de problemas considerados intratáveis, como será visto na próxima seção, entende-se que uma breve introdução à Teoria da Computação Paralela faz-se necessária, em face das exigências dos avanços tecnológicos e prováveis tendências da Teoria da Computação.

Em seu artigo "Parallel Combinatorial Computing", Karp [KAR 91] apresenta um esquema da história recente da computação paralela. Afirma que, embora os primeiros sucessos na paralelização tenham origem no campo da Computação Científica, onde uma grande quantidade de problemas numéricos em ciência e engenharia requeriam uma vasta quantidade de recursos computacionais perfeitamente executáveis em paralelo, as aplicações de computação paralela aumentaram maciçamente em diversidade e a Computação Científica não é mais o seu campo dominante. Em seu lugar, aplicações em áreas tais como otimização combinatória, prova de teoremas, manipulação algébrica simbólica e processamente de linguagem natural, entre outras, se tornaram proeminentes. No entanto, os atributos para estes problemas lógicos/combinatoriais são muito diferentes das propriedades regulares características de muitos algoritmos numéricos, envolvendo a manipulação de estruturas discretas tais como grafos, cadeias e expressões simbólicas. Portanto, é natural que as arquiteturas paralelas apropriadas para esta classe de aplicações sejam muito diferentes das arquiteturas paralelas que são adequadas para aplicações numéricas. Por exemplo, máquinas vetoriais e 'systolic arrays', que são perfeitamente adequados para muitos algoritmos numéricos altamente regulares, podem não resolver efetivamente problemas lógicos ou combinatoriais. O paradigma de paralelização de dados para a decomposição de um problema em interações independentes, de modo que as partes sejam executáveis concorrentemente, pode ser menos apropriado no domínio combinatorial do que no domínio numérico.

Com o considerável aumento das aplicações em computação paralela em muito grande escala, torna-se necessário reexaminar continuamente a escolha de arquiteturas de máquinas, linguagem de programação e ambiente de programação, tanto quanto as questões de como algoritmos paralelos devem ser representados e avaliados.

Entretanto, conforme argumentam Lakshmivarahan & Dhall [LAK 90], na prática o grande crescimento no 'raw computational power', possível através destas máquinas, não é traduzido diretamente num crescimento comparável na performance dos algoritmos. Existe um número de razões para esta disparidade entre força computacional possível e performance alcançada. Os autores observam que para se obter a máxima performance, precisa-se manter tantos processadores ativos quanto possível. Mas isto é criticamente dependente do algoritmo usado. Pode ser que o próprio problema seja tal que não admita um algoritmo paralelo eficiente. Ou que o tipo de paralelismo que o algoritmo escolhido exiba pode não ser interpretado apropriadamente dentro da arquitetura da máquina usada. Isto significa que nem

todo algoritmo paralelo é eficientemente mapeado sobre uma máquina paralela realística, ou então os processadores estão ativos o tempo todo, mas realizam uma grande quantidade de computações redundantes. Além disso, quando dois ou mais processadores cooperam para a solução de um problema, existirá, definitivamnete, uma necessidade de comunicação e coordenação entre eles. Entender a natureza e a extensão de como cada um destes fatores afeta a performance de algoritmos paralelos e arquiteturas paralelas, constitui os objetivos fundamentais da Teoria da Computação Paralela.

Em relação à computação paralela científica, a existência de máquinas cada vez mais poderosas trouxe consigo a necessidade de reciclagem dos princípios que norteiam o desenvolvimento de algoritmos. Infelizmente, observa-se que o desenvolvimento de 'hardware' e de 'software' paralelos não ocorreu de forma similar. Sob o ponto de vista de Cunha [CUN 97], pesquisadores têm manifestado preocupação em relação ao incrível aumento de velocidade de processamento computacional versus a exatidão e controle de erros nos cálculos.

Divério [DIV 95] propõe, em sua tese de doutorado, o desenvolvimento de uma aritmética de alta exatidão e desempenho compatível para supercomputadores vetoriais. Argumenta também sobre a existência de uma grande lacuna entre o avanço tecnológico no desenvolvimento de computadores cada vez mais rápidos e poderosos, e a qualidade de realização dos cálculos, constatando que resultados obtidos de forma extremamente rápida através de supercomputadores vetoriais e/ou paralelos, nem sempre são confiáveis.

Segundo Cook [COO 85], estas são algumas razões que motivaram muitos cientistas de Computação Téorica, incluindo ele próprio, a dedicarem-se nos últimos anos ao desenvolvimento de uma fundamentação para o estudo sistemático de computadores e algoritmos paralelos. Neste sentido, é essencial a escolha de um modelo abstrato de computação paralela. Embora tal modelo possa ter relevância limitada nesse estudo, uma máquina abstrata permite que os pesquisadores trabalhem independentemente da tecnologia existente, procurando capturar melhor as características essenciais da computação paralela. Além disso, os modelos abstratos providenciam uma forte influência nos tipos de arquiteturas e algoritmos paralelos que deverão prevalecer no início do próximo século.

Um trabalho consistente sobre computação paralela e distribuída é apresentado na obra editada por Zomaya [ZOM 96], onde um estudo comparativo entre algoritmos sequenciais e paralelos é estabelecido em termos de modelos abstratos e recursos computacionais, culminando com uma classificação de problemas resolvíveis por algoritmos paralelos. Observa que para o estudo formal de algoritmos paralelos é necessário um refinamento dos modelos de máquina de Turing, devendo-se optar por modelos mais práticos, tais como máquinas paralelas de acesso randômico (PRAM), que são mais adequadas para a descrição e programação de algoritmos paralelos.

2.4 Complexidade

O advento dos computadores permitiu a automatização dos métodos de resolução de um problema. No entanto, ao se pretender utilizar uma máquina para tratar problemas reais, é natural que exista uma preocupação computacional em relação aos processos desenvolvidos. Essa preocupação, por sua vez, implica na procura da construção de algoritmos não somente

eficazes, garantindo a obtenção da solução desejada, mas que resolvam de um modo "eficiente" o problema proposto. De fato, devido aos limites físicos dos recursos disponíveis, não basta identificar um problema e garantir que exista pelo menos uma solução para ele. É ainda necessário saber se ele pode ser *efetivamente* resolvido em tempo e espaço de memória finitos.

O termo "complexidade" refere-se em geral, aos requerimentos de recursos necessários para que um algoritmo possa resolver um problema computacional. Segundo Cook [COO 83], várias medidas de complexidade de um algoritmo surgiram historicamente, mas sem dúvida uma das mais importantes é a medida de tempo. Sua justificativa é dada em razão de que uma boa parte da pesquisa em Ciência da Computação consiste em projetar e analisar algoritmos quanto à eficiência, sendo que, sob o ponto de vista computacional, algoritmos importantes geralmente são aqueles que fornecem a solução de um problema com a rapidez desejada. Este pensamento é também compartilhado por Garey & Johnson [GAR 79], referindo-se a um algoritmo mais eficiente como aquele que é o mais rápido.

Já que requerimentos de tempo são frequentemente considerados como fatores dominantes para determinar se um particular algoritmo é eficiente o bastante para ser útil na prática, para os propósitos do presente trabalho será considerada apenas a complexidade de tempo de um algoritmo, usando o termo *complexidade*, simplesmente.

2.4.1 Complexidade de tempo

Os requerimentos de tempo de um algoritmo são convenientemente expressados em termos do "tamanho de uma instância" do problema, porque se espera que a dificuldade do problema varie rigorosamente com esta medida. Conforme estabelecido por Garey & Johnson [GAR 79], para que a complexidade de tempo de um algoritmo seja definida de um modo preciso, o primeiro passo é definir o tamanho de uma instância levando em conta um esquema de codificação fixo que associe instâncias do problema com as cadeias de símbolos que os descrevem.

Define-se, então, formalmente o tamanho da instância I de um problema Π, como o comprimento da entrada dessa instância, dado pelo número de símbolos na descrição de I, obtida de um esquema de codificação do problema Π.

A *complexidade de tempo* (ou simplesmente *complexidade*) para um algoritmo é uma função definida sobre os tamanhos das instâncias do problema, que considera o requerimento de tempo exigido por entradas de um mesmo tamanho, avaliando a quantidade de tempo necessária para o algoritmo resolver problemas desse tamanho.

2.4.2 Principais medidas de complexidade

Se para um dado tamanho de instância a complexidade é tomada como a complexidade máxima para qualquer entrada desse tamanho, a complexidade é chamada *complexidade no pior caso*.

Entretanto, em muitos casos a complexidade de um algoritmo não depende só do tamanho da entrada, mas de certas propriedades da entrada. Veja por exemplo o problema de ordenação de uma lista. Ordenar uma lista em que quase todos os seus elementos estão ordenados, não requer o mesmo esforço necessário para ordenar uma lista de mesmo tamanho,

mas com os elementos em grande desordem. Neste caso, convém estudar a *complexidade do caso médio*, que é a média ponderada da complexidade de cada entrada possível, considerada a probabilidade de sua ocorrência.

Embora a complexidade no caso médio possa ser mais significativa que a complexidade no pior caso, como no estudo de algoritmo de procura heurística em Inteligência Artificial, em geral a complexidade de um algoritmo no caso médio é mais difícil de determinar, pois é preciso que se faça algumas hipóteses sobre a distribuição das entradas, e hipóteses realísticas freqüentemente não são matematicamente tratáveis. Estudos recentes sobre a complexidade do caso médio podem ser encontrados no trabalho de Rosa [ROS 97], onde é proposta uma metodologia para o cálculo da complexidade média de algoritmos.

As complexidades no pior caso e no caso médio são consideradas as principais medidas de complexidade de tempo de um algoritmo.

Para que se tenha um tratamento uniforme no estudo de complexidade, no restante deste trabalho será considerada apenas a medida de complexidade no pior caso.

2.4.3 Complexidade assintótica

Para analisar a complexidade associada à resolução de um problema específico, é preciso determinar a quantidade de recursos que um algoritmo requer para resolver uma instância qualquer do problema. Segundo Bovet [BOV 94], é natural pensar que quanto maior as instâncias, mais unidades de recurso o algoritmo requer, isto é, o limite sobre os recursos deve ser uma função crescente no tamanho da instância.

Com o objetivo de desenvolver uma teoria tão geral quanto possível, independente do modelo de computação escolhido, e capaz de capturar a complexidade inerente do problema, estuda-se a razão de crescimento do requerimento de recurso mais que a sua própria avaliação. Na prática, sempre que as instâncias não excedem um tamanho fixo, um algoritmo com uma rápida razão de crescimento poderia ser preferível a outro com menor razão de crescimento. Entretanto, se o interesse é estudar como a complexidade cresce quando o tamanho das instâncias tornam-se maiores, pode-se formalizar tal comportamento através da notação de *ordem assintótica*, definida da seguinte maneira:

Definição 2.1: *Ordem Assintótica*

Seja $g : \mathbf{N} \rightarrow \mathbf{N}$ uma função total. A classe de ordem assintótica de g, representada por $\boldsymbol{0}[g(n)]$, consiste de todas as funções $f : \mathbf{N} \rightarrow \mathbf{N}$ para as quais existem constantes c e n_0 tal que $f(n) \leq c\ g(n)$, para qualquer $n \in \mathbf{N}$, $n > n_0$.

A avaliação da eficiência de algoritmos para problemas com instâncias muito grandes é melhor analisada através da *complexidade assintótica*, que leva em conta o comportamento assintótico da complexidade de tempo. Este fato pode ser melhor explicado da seguinte maneira: suponha que se deseja determinar a complexidade de tempo T(n), de algum algoritmo, onde n é o tamanho de uma instância qualquer do problema. Sendo T(n) definida de tal forma que seja independente da máquina, uma análise *a priori* é suficiente para determiná-la. Através desta análise pode-se encontrar uma função g(n) tal que $T(n) = \boldsymbol{0}[g(n)]$. Isto significa que se o algoritmo rodar num mesmo computador, sobre os mesmos tipos de dados, mas para valores crescentes de n, as complexidades de tempo resultantes serão menores que alguma constante multiplicada por g(n), a partir de um certo n.

Horowitz [HOR 78] apresenta as complexidades assintóticas mais comuns visualizadas através do gráfico apresentado na FIGURA 2.6 abaixo.

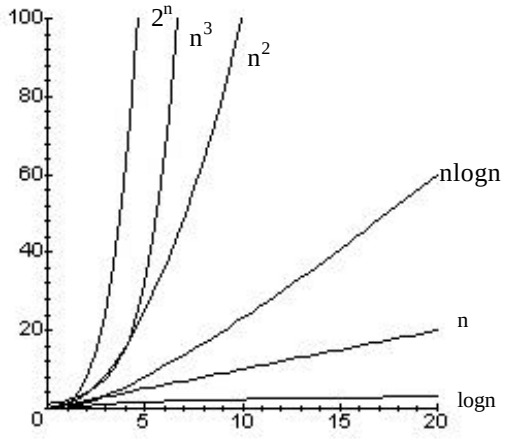

FIGURA 2.6 - Crescimento assintótico de funções de complexidade de tempo.

Observa-se que para instâncias de tamanho muito grande tem-se a seguinte relação:

$$\boldsymbol{0}(1) < \boldsymbol{0}(\log n) < \boldsymbol{0}(n) < \boldsymbol{0}(n \log n) < \boldsymbol{0}(n^2) < \boldsymbol{0}(n^3) < \boldsymbol{0}(2^n)$$

$\boldsymbol{0}(1)$ significa que o número de execuções de operações básicas é fixo e portanto o tempo total é limitado por uma constante. Com exceção de $\boldsymbol{0}(2^n)$, todas as ordens assintóticas acima tem uma importante propriedade comum: elas são limitadas por um polinômio.

Conforme Garey e Johnson [GAR 79], um algoritmo cuja complexidade assintótica é limitada por um polinômio, é chamado de *algoritmo de tempo polinomial*, caso contrário, é chamado de *algoritmo de tempo exponencial*, mesmo que sua complexidade não esteja na forma de uma função exponencial.

2.4.4 A complexidade como um funcional

A complexidade de um algoritmo foi definida como uma função sobre o conjunto de tamanhos das instâncias do problema. Retomando esta definição, apresenta-se nesta seção um refinamento do conceito de complexidade apresentada originalmente por Toscani & Veloso em [TOS 90], tendo sido aperfeiçoada em [TOC 2001]. Embora não seja usada no restante do trabalho, a concepção da complexidade como um funcional tem o propósito de obter uma maior precisão matemática na formalização dos conceitos.

Dado um problema abstrato $\mathbf{p} = (D, R, q)$, com solução α, seja $\mathbf{A}$ o conjunto dos algoritmos que computam α. Em muitos casos, para analisar um algoritmo pode-se isolar uma operação fundamental particular para o problema sob estudo, ignorar o contador ('bookkeeping') e contar o número de operações básicas escolhidas. Para calcular a complexidade de um algoritmo $\mathbf{a} \in \mathbf{A}$, é necessário identificar as operações

fundamentais do algoritmo e definir a função que dá o tamanho da entrada. No caso de haver mais de uma operação fundamental, é necessário definir o peso de cada uma [BAA 78].

Seja E o conjunto de todas as seqüências de execuções de operações fundamentais. Considere o seguinte diagrama :

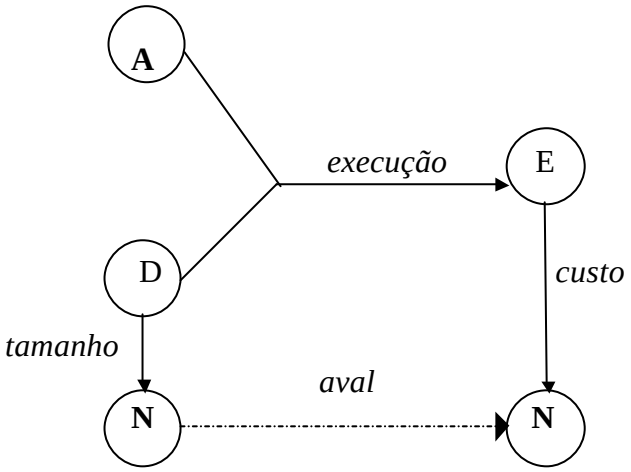

FIGURA 2.7 - Diagrama para definição da Complexidade

As funções representadas no diagrama acima são definidas por:

- *execução* : **A** x D $\rightarrow$ E ; *execução*(**a**, d) := seqüência de execuções de operações fundamentais efetuadas na execução do algoritmo **a**, com entrada d.

- *custo* : E $\rightarrow$ **N** ; *custo*(s) := comprimento da seqüência s, definido conforme o peso estabelecido para as operações fundamentais.

- *tamanho* : D $\rightarrow$ **N** ; *tamanho*(d) := tamanho da entrada d.

Escolhidas as funções fundamentais e definida a função *tamanho*, o conjunto **A** é particionado, ficando na mesma classe os algoritmos com mesmas funções fundamentais e mesma função *tamanho*. A complexidade é então definida dentro de cada classe.

Considere C como uma dessas classes e defina a complexidade de um elemento de C como uma função $aval: \mathbf{N} \rightarrow \mathbf{N}$.

A complexidade é portanto um funcional do tipo $C \rightarrow (\mathbf{N} \rightarrow \mathbf{N})$, isto é:

- *complexidade* : $C \rightarrow (\mathbf{N} \rightarrow \mathbf{N})$
  complexidade(**a**) := *aval*

onde $\mathbf{a} \in \mathbf{A}$ *e* $aval: \mathbf{N} \rightarrow \mathbf{N}$.

Uma boa parte da pesquisa dentro da Teoria de Algoritmos têm sido dedicada ao estudo da complexidade. Usualmente, um algoritmo é construído e depois analisado quanto à complexidade. Com o desenvolvimento de uma metodologia para o cálculo de complexidade, este processo pode ser revertido, de tal forma que a complexidade passe a ser um fator integrante do projeto de construção do algoritmo. Além disso, o conhecimento da complexidade de um algoritmo é determinante para a obtenção de uma solução do problema considerado, no sentido de perseguir a solução exata ou procurar uma solução aproximada.

Na próxima seção serão apresentadas as principais classes de complexidade e examinadas as questões relacionadas com a estrutura de tais classes. Especial atenção é dada à classe de problemas NP-completo, introduzida por Cook em 1971, pela sua relevância e por traduzir as principais idéias presentes atualmente na área de Complexidade Computacional.

A introdução de classes de problemas que têm complexidade similar com respeito a um modelo de computação e medida de complexidade específicos, além do estudo das propriedades intrínsecas de tais classes, são os principais objetivos da Teoria da Complexidade.

2.5 Classes de Complexidade

Fica evidente na ampla literatura sobre complexidade que muitos problemas computacionais podem ser resolvidos por algoritmos de tempo polinomial, o que significa que para qualquer entrada de tamanho n, a complexidade de tempo no pior caso é $\boldsymbol{0}(n^k)$ para alguma constante k. Outros problemas existem, chamados indecidíveis, que não podem ser resolvidos por qualquer computador, não interessa quanto tempo é providenciado. Existem também problemas que podem ser resolvidos, mas não em tempo $\boldsymbol{0}(n^k)$ para qualquer constante k.

A maioria dos pesquisadores concorda que os primeiros problemas podem, ainda que grosseiramente, ser considerados como *tratáveis*, enquanto que os outros são certamente *intratáveis* sob o ponto de vista computacional.

Há ainda a considerar problemas cujo 'status' é desconhecido. Nenhum algoritmo de tempo polinomial é conhecido, nem tampouco foi provada a não existência de um algoritmo de tempo polinomial que resolva tais problemas.

A Teoria da Complexidade está estruturada matematicamente de modo a classificar problemas computacionais relativamente a sua dificuldade intrínseca.

Nesta seção são definidas as principais classes de complexidade para problemas de decisão, relativamente à performance de seus algoritmos com respeito a complexidade de tempo, sendo introduzidas também algumas questões relacionadas à tratabilidade de problemas. Todas as definições foram retiradas de Garey e Johnson [GAR 79].

2.5.1 A Classe P

A classe P é definida como o conjunto de todos os problemas de decisão resolvíveis por um algoritmo determinístico em tempo polinomial.

A importância de tal classe deriva do fato de ela conter todos os problemas "simples", isto é aqueles que são computacionalmente tratáveis. Observa-se, no entanto, que um problema tratável não é necessariamente "eficiente". De fato, devido a classificação de algoritmo em relação à complexidade de tempo apresentada anteriormente não ser mais do que uma definição de um limite superior para a ordem de complexidade, é possível que hajam exceções, no sentido de que existam alguns algoritmos classificados como exponenciais que apenas para alguma instância apresentam esse tipo de complexidade, enquanto que outros, classificados como polinomiais, tenham um grau tão alto que se tornam ineficientes, na prática. Mas, muito embora nem todo problema pertencente a classe P possa ser considerado eficiente, o fato de não estar em P, caracteriza o problema certamente como intratável, com raríssimas exceções.

Outra razão que justifica a ampla aceitação da definição da classe P em termos de complexidade polinomial, é a equivalência entre modelos de computação realísticos com respeito ao tempo polinomial. Em outras palavras, é sempre possível transformar um algoritmo de tempo polinomial obtido em determinado modelo computacional num outro algoritmo também de tempo polinomial aplicável num modelo de computação básico. Portanto a classe P é invariante para uma grande coleção de modelos de computação frequentemente usados.

Como foi estabelecido anteriormente, problemas de decisão são formalmente representados como linguagens, de tal forma que a solução de um problema de decisão equivale ao reconhecimento da linguagem correspondente por uma máquina de Turing. A classe P é também definida formalmente através de uma máquina de Turing determinística (MTD). A contrapartida formal de um algoritmo de tempo polinomial é um programa MTD em tempo polinomial, cuja definição é apresentada a seguir:

Seja M um programa MTD que pára para todas as entradas $x \in \Sigma^*$ e $T_M : \mathbf{N} \rightarrow \mathbf{N}$ sua complexidade de tempo, definida por

$T_M(n) = \max\{m \in \mathbf{N} \mid \exists x \in \Sigma^*$, com $|x| = n$ e o tempo de computação de M sobre x é $m\}$

O programa M é chamado de *programa MTD de tempo polinomial* se existe um polinômio p tal que, para todo $n \in \mathbf{N}$,

$$T_M(n) \leq p(n).$$

A classe P é então a classe de linguagens definida formalmente por:

$P = \{ L \mid$ existe um programa MTD de tempo polinomial M para o qual $L = L_M\}$

Seguindo a notação de Johnson [JOH 90], esta classe está contida na classe mais geral FP onde se agrupam todos os algoritmos polinomiais sem restrição ao grau do polinomio envolvido na classificação. Descritas em termos de problemas, a classe FP contém todos os problemas resolvíveis em tempo polinomial, enquanto que a sua restrição P contém todos os problemas de decisão resolvíveis em tempo polinomial.

Observa-se ainda, que se existe um método com base num modelo determinístico para resolver um dado problema de decisão Π, com um gasto polinomial de tempo, ou seja uma máquina determinística reconhece a linguagem $L_\Pi = \{ x \in \Sigma^* \mid x \in Y_\Pi \}$ correspondente a resposta 'sim' para o problema em tempo polinomial, então também o problema complementar que corresponde a resposta 'não', pode

ser reconhecida com o mesmo limite polinomial de tempo. De fato, uma máquina determinística pára qualquer que seja a entrada, e, portanto pode-se fazer o reconhecimento simplesmente trocando os estados finais correspondendo às respostas ‘sim’ e ‘não’.

Se a classe de problemas complementares for representada por Co-P, verifica-se facilmente que P = Co-P.

A figura seguinte ilustra parcialmente a relação entre as classes mencionadas.

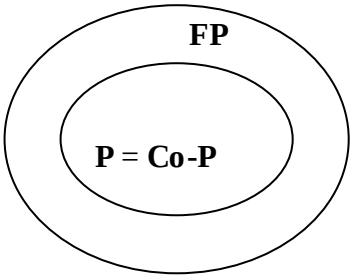

FIGURA 2.8 - Relação entre as classes de problemas de tempo polinomial

Outro ponto que deve ser ressaltado, diz respeito à classificação, dentro da classe P, daqueles problemas resolvíveis de modo muito rápido, traduzido num tempo polinomial sobre log n, através de computadores paralelos com um número razoável de processadores. Cook [COO 85] apresenta um estudo sobre a classe NC, cujo nome vem de “Nick’s Class”, em honra de Nicholas Pippenger, o primeiro pesquisador a estudar esta classe. Informalmente, a classe NC consiste de todos os problemas resolvíveis com quantidade de ‘hardware’ polinomialmente limitado, em tempo poli-logaritmico.

Sob o ponto de vista de Zomaya [ZOM 96], a Teoria da Complexidade estabelece um mapa de classificação espectral para os limites de recursos necessários para resolver diferentes classes de problemas, usando as medidas básicas de um modelo básico de computação. Neste mapa de classificação espectral, a classe mais distante do espectro seria a classe dos problemas intratáveis, cujas soluções requerem recursos de tempo e espaço computacional de crescimento exponencial; a zona intermediária corresponde à classe de problemas resolvíveis pelo consumo de uma quantidade viável de recursos de tempo e espaço que cresce como uma função polinomial; e a classe mais próxima do espectro vem a ser justamente a classe de problemas que podem ser resolvidos ultrarápido, usando um grande número de processadores, consumindo recursos de tempo e espaço limitados sublinear ou logarítmico.

2.5.2 A Classe NP

A classe NP é definida como o conjunto de todos os problemas de decisão resolvíveis por um algoritmo não-determinístico em tempo polinomial.

O uso do termo “resolver” nesta definição informal deve ser tomado com alguma restrição. Um algoritmo não-determinístico é dito de tempo polinomial quando é possível checar se uma dada entrada satisfaz o problema em tempo polinomial. Neste sentido a noção capturada é aquela da verificação de uma solução em tempo polinomial, mais que um método realístico de resolver o problema. O não-determinismo permite várias computações sobre uma

dada entrada, uma para cada conjetura. A classe NP pode ser vista informalmente, como a classe dos problemas de decisão para os quais a verificação de que uma solução estimada para uma dada entrada satisfaz todos os requerimentos do problema, pode ser checada rapidamente.

Naturalmente, a definição formal da classe NP é dada em termos de uma máquina de Turing não-determinística (MTND).

Seja M um programa MTND. O tempo requerido por M para aceitar a cadeia $x \in L_M$ é definido como o tempo mínimo, sobre todas as computações de M que aceitam x, contado como o número de passos que ocorre nos estágios de conjetura e checagem, até que seja atingido o estado de parada q_s.

A função complexidade de tempo $T_M : \mathbf{N} \rightarrow \mathbf{N}$ é definida por:

$T_M(n) = \max\{1; \min\{m \mid \exists x \in L_M$, com $|x| = n$ tal que M aceita x no tempo $m\}\}$

Observa-se que esta função depende somente do número de etapas que ocorre nas computações aceitas. Por convenção, define-se $T_M(n)$ igual a 1 sempre que nenhuma entrada de comprimento n é aceita por M.

O programa M é chamado de programa MTND de tempo polinomial se existe um polinômio p tal que $T_M(n) \leq p(n)$, para $n \geq 1$.

Finalmente, a classe NP é formalmente definida por:

NP = { L | existe um programa MTND de tempo polinomial M tal que $L_M = L$ }

Considerando os problemas complementares à classe NP como aqueles que exigem uma justificativa à resposta ‘não’ com estágio de reconhecimento correspondendo a um algoritmo polinomial no tamanho da entrada do problema, define-se a classe Co-NP. É interessante observar que, contrariamente ao que ocorre com a classe P que coincide com a sua classe complementar Co-P, a classe Co-NP contém problemas que se desconhece a pertinência ou não à classe NP.

As seguintes questões continuam em aberto:

(i) ‘P =NP ?’

(ii) ‘NP = Co-NP’?

(iii) ‘P = NP $\cap$ Co-NP ?’

A Figura 2.9 apresentada a seguir, ilustra as conjeturas a cerca destas questões:

(i) P $\neq$NP

(ii) NP $\neq$ Co-NP’

(iii) P $\neq$ NP $\cap$ Co-NP

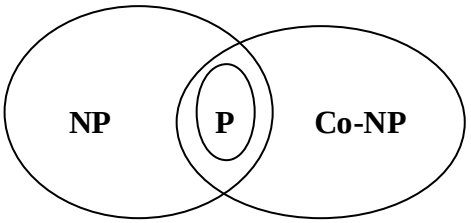

FIGURA 2.9 - Conjeturas sobre as classes P, NP e Co-NP

2.5.3 Relação entre as classes P e NP

Uma questão fundamental na Teoria da Complexidade Computacional, desde o início da década de setenta, é a relação entre as classes P e NP. Será P = NP ?

Observa-se primeiramente que $P \subseteq NP$, o que significa que todo problema de decisão resolvível por um algoritmo determinístico de tempo polinomial é também resolvível por um algoritmo não-determinístico de tempo polinomial. De fato, se $\Pi \in P$ e α é qualquer algoritmo determinístico de tempo polinomial para Π, pode-se obter um algoritmo não-determinístico em tempo polinomial para Π, simplesmente usando α como um algoritmo de reconhecimento para uma justificativa à resposta 'sim' de Π . Portanto, $\Pi \in P$ implica $\Pi \in NP$.

Entretanto, continua em aberto a questão se P está contido propriamente em NP, ou se P é igual a NP. Até hoje ninguém conseguiu provar se existe um problema em NP que não está em P.

Sob o ponto de vista de Emde Boas [EMD 90], a questão 'P = NP ?' indaga sobre a potencialidade do não-determinismo: existem problemas que uma máquina não-determinística de tempo polinomial pode resolver e que não podem ser resolvidos em tempo polinomial por uma máquina determinística razoável ? Neste contexto, Garey e Johnson [GAR 79] consideram que algoritmos não-determinísticos de tempo polinomial parecem ser mais potentes que aqueles determinísticos de tempo polinomial, além de não se conhecer nenhum método geral para converter aqueles nestes. O melhor resultado geral que pode ser estabelecido até o presente é que, se $\Pi \in NP$, então existe um polinômio p tal que Π pode ser resolvido por um algoritmo determinístico tendo tempo de complexidade $\mathbf{0}(2^{p(n)})$. Portanto, além de parecer mais razoável, existem fortes razões para se acreditar que a inclusão é própria, isto é, P não é igual a NP.

A partir da conjetura $P \neq NP$, a distinção entre os conjuntos P e NP-P é muito importante e significativa: todos os problemas em P podem ser resolvidos com algoritmo de tempo polinomial, enquanto que todos os problemas em NP-P são intratáveis. Portanto, dado um problema de decisão $\Pi \in NP$, se é verdade que $P \neq NP$, quais das duas possibilidades se mantém para Π ?

Sem dúvida, este é um dos assuntos que têm preocupado muitos, senão todos, os pesquisadores desta área. Em muitos casos, é possível responder a pergunta através da idéia de redução de um problema em outro. Segundo Papadimitriou [PAP 94], a redutibilidade dá

uma noção precisa do que significa um problema ser pelo menos tão difícil quanto outro. Daí porque a noção de redutibilidade entre problemas tornou-se tão importante.

2.5.4 Transformação Polinomial

A seguir será apresentada a definição formal de um tipo de redução chamada de *transformação polinomial*, conforme Garey e Johnson [GAR 79].

Definição 2.1: *Transformação polinomial*

Uma transformação polinomial de uma linguagem $L_1 \subseteq \Sigma^*$ para uma linguagem $L_2 \subseteq \Sigma^*$, é uma função $f : \Sigma^* \rightarrow \Sigma^*$ que satisfaz as duas condições seguintes:

(i) Existe um programa para MTD de complexidade polinomial que computa f;

(ii) Para todo $x \in \Sigma^*$, $x \in L_1$ se e somente se $f(x) \in L_2$.

Se existe uma transformação polinomial de L_1 para L_2, escreve-se simbolicamente

$$L_1 \propto L_2,$$

e lê-se 'L_1 é polinomialmente transformável em L_2'.

Pode-se também estender a definição de transformação polinomial a nível de problemas de decisão: se Π_1 e Π_2 são problemas de decisão, escreve-se

$$\Pi_1 \propto \Pi_2,$$

sempre que existir uma transformação polinomial de $L(\Pi_1)$ para $L(\Pi_2)$.

Propriedades: Uma transformação polinomial satisfaz as seguintes propriedades:

- A classe P é fechada em relação à transformação polinomial: se $L_1 \propto L_2$, então $L_2 \in \mathbf{P}$ implica que $L_1 \in P$.
- A relação $\propto$ é transitiva: se $L_1 \propto L_2$ e $L_2 \propto L_3$, então $L_1 \propto L_3$.
- Duas linguagens L_1 e L_2 são identificadas como polinomialmente equivalentes, sempre que $L_1 \propto L_2$ e $L_2 \propto L_1$.

As duas últimas propriedades caracterizam a relação $\propto$ como uma relação de equivalência. Além disso, a relação $\propto$ impõe uma ordenação parcial sobre essas classes de equivalência. Observa-se então, que a classe P é identificada como a menor classe de equivalência sob esta ordenação parcial e, portanto, pode ser vista como a classe que contém as linguagens (problemas de decisão) consideradas mais 'fáceis' sob o ponto de vista computacional.

Por outro lado, especial interesse despertam aquelas classes que são maximais nesta relação, no sentido de conterem as linguagens (problemas de decisão) mais 'difíceis'.

Num contexto mais geral, Papadimitriou [PAP 94] define a noção de *completude* para capturar a essência da dificuldade de uma classe de complexidade.

Uma linguagem L pertencente a uma classe de complexidade C é dita C-completo se, qualquer linguagem L' em C pode ser reduzida a L.

Neste sentido, a classe NP-completa, que é o objeto de interesse na seção seguinte, é distinguida por conter as linguagens mais difíceis de NP.

2.5.5 A Classe NP-Completo

Formalmente, uma linguagem L é definida ser NP-completa se são válidas as condições:

i. $L \in NP$; e

ii. para toda outra linguagem $L' \in NP$ tem-se que $L' \propto L$.

Informalmente, um problema de decisão Π é NP-completo se $\Pi \in NP$ e, para todo problema de decisão $\Pi' \in NP$, tem-se que $\Pi' \propto \Pi$. Portanto, os problemas NP-completos são identificados como os problemas mais difíceis em NP.

Assim, se um problema NP-completo pode ser resolvido em tempo polinomial, então todos os problemas em NP podem também ser resolvidos dessa forma. A partir desta caracterização, pode-se concluir que se um problema de decisão Π é NP-completo então , sob a conjetura $P \neq NP$, tem-se que $\Pi \in NP-P$.

O primeiro problema a ser identificado como NP-completo foi o da Satisfabilidade, apresentado abaixo. Este resultado, conhecido como Teorema de Cook, foi demonstrado em 1971 e se constitui num dos resultados fundamentais dentro da Teoria da Complexidade Computacional.

Satisfabilidade - **SAT**

Instância: Conjunto de literais $U = \{u_1, \bar{u}_1, \ldots, u_n, \bar{u}_n\}$ e conjunto de cláusulas $C = \{c_1, \ldots, c_m\}$, onde cada c_i, $1 \leq i \leq m$ é constituído por literais de U, na forma conjuntiva normal.

Resposta: "Sim", se e só se existe uma atribuição de valores lógicos a cada um dos literais em U tal que todas as cláusulas de C são satisfeitas (isto é, são verdadeiras);
"Não", caso contrário.

QUADRO 2.1 - SAT – O problema da satisfabilidade

Uma demonstração do Teorema de Cook é apresentada integralmente por exemplo em Garey & Johnson [GAR 79]. Em linhas gerais a demonstração deste teorema segue o seguinte raciocínio:

É imediato que existem 2^n atribuições possíveis de valores lógicos aos literais e, até o presente, não foi possível encontrar nenhum algoritmo que consiga determinar em tempo não exponencial se, pelo menos uma atribuição satisfaz todas as cláusulas. É também imediato que, em posse de uma atribuição, se pode verificar em tempo polinomial se essa atribuição satisfaz todas as cláusulas, sendo portanto **SAT** pertencente à classe NP. Resta provar que **SAT** é completo para NP para transformações polinomiais, ou seja, resta a provar que para qualquer linguagem L em NP, tem-se L $\propto$ L(**SAT**). Uma vez que, se uma linguagem L pertence a NP, ela pode ser descrita através de uma MTND polinomial no tempo que a reconhece, a restante demonstração indica como se pode construir uma MTND que além de reconhecer L, constrói a transformação polinomial de L para L(**SAT**).

Um ano mais tarde da primeira apresentação de Cook, Karp provou que outros 21 problemas eram NP-completos, demonstrando fortemente a importância do assunto. A partir daí, mais e mais problemas NP-completos têm sido estabelecidos nas mais diversas áreas, conforme é atestado pela lista de cerca de trezentos problemas apresentados por Garey e Johnson [GAR 79], e tantos outros problemas que apareceram depois disso. Informações atualizadas sobre NP-completude encontram-se na coluna de Johnson, no Journal Computing System Science.

2.5.6 A Classe NP-Intermediário

Existem problemas em NP que não estão em P nem em NP-completo ?

Antes de proceder à resposta desta questão, convém examinar as possibilidades de representação da classe NP, ilustradas na Figura 2.10 a seguir.

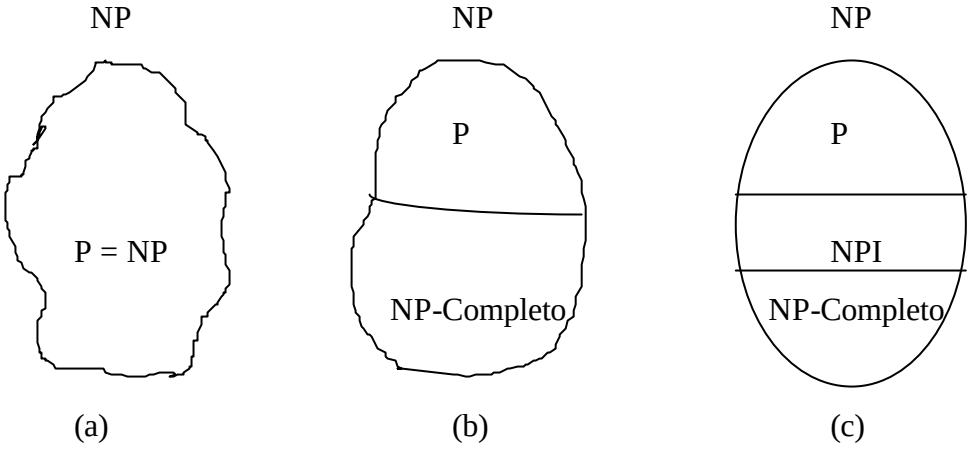

FIGURA 2.10 - As possibilidades de representação da classe NP

Na Figura 2.10(a), considera-se a possibilidade de que P = NP. Esta hipótese traduz um ponto de vista extremamente otimista, onde todo problema resolvido por um algoritmo não-determinístico em tempo polinomial admitiria também um algoritmo determinístico de tempo polinomial que o resolvesse. Neste caso, a questão colocada não teria significado algum, já que todo problema em NP seria simultaneamente computável em tempo polinomial e NP-completo.

Por outro lado, pelo que se conhece de NP-completude hoje, é provável que $P \neq NP$. Sob esta conjetura, restam as alternativas mostradas pelas outras duas figuras. Pela Figura 2.10(b), todo problema em NP deveria pertencer a classe P ou à classe NP-completo. No entanto, por um resultado devido a Ladner[LAD 75], esta alternativa foi provada ser impossível. Ladner mostrou que, se $P \neq NP$, então existe uma linguagem (problema) em NP que não está em P nem tampouco em NP-completo. O significado desta proposição é traduzido pelo fato que existem problemas cujo grau de dificuldade está entre os mais fáceis e os mais difíceis problemas em NP. Tais problemas constituem a classe chamada NP-intermediário, denotada por NPI. Qualquer problema aberto em NP, isto é, todo problema que ainda não foi provado estar em P nem em NP-completo, pode ser visto como um candidato à classe NPI.

Portanto, ao mundo de NP restaria, provavelmente, a situação representada pela Figura 2.10(c), já que a primeira alternativa é considerada pela grande maioria dos pesquisadores como improvável.

2.5.7 A Classe NP-Difícil

A classe NP-completo foi definida como aquela classe que contém os problemas mais difíceis em NP. As técnicas conhecidas para provar a NP-completude podem também serem usadas para provar a dificuldade de problemas não restritos à classe NP. Qualquer problema de decisão Π, pertencente ou não `a classe NP, que seja redutível a um problema NP-completo, terá a propriedade de não ser resolvido em tempo polinomial, a menos que P = NP. Neste caso, tal problema Π é chamado NP-difícil, já que ele apresenta dificuldade não menor que qualquer problema NP-completo, definindo desta maneira uma nova classe de problemas: a classe NP-Difícil.

Esta noção de NP-dificuldade pode ser generalizada, de forma que possa ser aplicada para outros problemas e não somente para problemas de decisão. Isto é possível através da generalização da noção de uma transformação polinomial, em termos de máquina de Turing.

A classe mais geral de problemas para os quais as definições apresentadas nesta seção se aplicam é a chamada *classe de problemas de busca*. As definições aqui apresentadas estão baseadas no trabalho desenvolvido por Garey & Johnson [GAR 79].

Um problema de busca $\Pi = (D_\Pi, S_\Pi)$ consiste de um conjunto D_Π de objetos finitos chamados instâncias e, para cada instância $I \in D_\Pi$, um conjunto $S_\Pi(I)$ de objetos finitos chamados solução para I.

Um algoritmo resolve um problema de busca Π se, dada como entrada qualquer instância $I \in D_\Pi$, o algoritmo retorna a resposta ‘não’ sempre que $S_\Pi(I)$ é vazio e, caso contrário, retorna alguma solução s pertencente a $S_\Pi(I)$. Observa-se que qualquer problema de

decisão Π pode ser formulado como um problema de busca. Basta definir $S_{\Pi}(I) = \{$'sim'$\}$ se $I \in Y_{\Pi=,}$ e $S_{\Pi}(I) = \varnothing$ se $I \notin Y_{\Pi}$. Desta maneira, um problema de decisão pode ser considerado simplesmente como um tipo especial de problema de busca.

2.6 Comentários Finais

O presente capítulo teve como objetivo elencar conceitos e resultados classicamente conhecidos dentro da Teoria da Ciência da Computação, especificamente da área de Complexidade Computacional, os quais formam um conjunto mínimo de informações necessárias para os desenvolvimentos apresentados nos capítulos seguintes.

No desenvolvimento do trabalho, muitos aspectos da teoria foram deixados de lado, não por serem menos importantes, mas por não apresentarem a relevância no direcionamento que se buscava. Por exemplo, a medida de complexidade definida pelo espaço de computação de um algoritmo não foi explorada, privilegiando-se a complexidade de tempo. O estudo da complexidade do caso médio foi preterido, em relação à complexidade do pior caso. Algoritmos paralelos, considerados sob o enfoque de uma classe de problemas altamente tratáveis, deram lugar à algoritmos com eficiência relativa, no contexto de problemas intratáveis.

No entanto, optou-se pela apresentação daqueles resultados que efetivamente pudessem contribuir para o entendimento dos capítulos seguintes, sendo que o principal aspecto que se buscou focalizar no desenvolvimento do trabalho, foi o de apresentar os conceitos dentro de uma rigorosa formalização matemática. Entende-se que um trabalho teórico bem fundamentado dá garantias para se obter resultados e conclusões com credibilidade científica. Para uma lista mais completa de resultados sugere-se a consulta à literatura considerada clássica ([AHO 74], [GAR 79], [HOR 78], [LEW 81], [PAP 94]), entre outros.

O próximo capítulo apresenta um estudo das questões relativas à complexidade de problemas de otimização, estabelecendo as bases para o entendimento da questão central deste trabalho, que é o desenvolvimento de uma teoria formal para algoritmos aproximativos, considerados como uma alternativa viável para problemas intratáveis.

3 Complexidade de Problemas de Otimização

Este capítulo apresenta os principais resultados da Teoria da Complexidade aplicados no contexto da otimização. Segundo a literatura clássica, a Teoria da Complexidade foi construída, originalmente, em termos de problemas de decisão. Em particular, quando se trata de problemas de otimização, muitos dos conceitos introduzidos no capítulo anterior precisam ser adaptados ou estendidos. Os resultados ora apresentados fazem-se necessários por seu caráter mais abrangente e, principalmente, em razão de motivar a discussão apresentada nos capítulos posteriores.

O objetivo aqui consiste em introduzir os conceitos básicos relacionados à complexidade de problemas de otimização, focalizando aqueles problemas que são computacionalmente difíceis de resolver e que por enquanto só podem ser eficientemente tratados através de algoritmos que fornecem soluções aproximadas. Nesse sentido, são abordados tópicos sobre como a eficiência de tais algoritmos e a complexidade computacional de problemas são medidos.

As definições apresentadas nesta seção seguem as notações de Bovet e Crescenzi [BOV 94] e T. Jansen [JAN 98]. Os problemas de otimização citados como exemplo fazem parte das centenas de problemas que constam no compêndio apresentado por G. Ausiello et alli [AUS 99], seguindo o mesmo padrão da famosa lista de problemas de decisão apresentada por Garey e Johnson [GAR 79] e sendo constantemente atualizados no endereço eletrônico disponibilizado em [CRE 95].

Uma breve descrição desses problemas é apresentada na seção de anexos, no final deste volume.

3.1 Problema de Otimização

Como visto anteriormente, os elementos básicos de um problema de otimização são:

- conjunto de *instâncias* ou objetos de entrada;
- conjunto de *soluções viáveis* ou objetos de saída, associados a cada instância;
- uma *função medida* definida para cada solução viável e,
- declaração do objetivo *ótimo* do problema: maximização ou minimizacão.

A definição apresentada a seguir para um problema de otimização genérico, é proposta em [AUS 99].

Definição 3.1: *Problema de otimização*

Um *problema de otimização p* é definido por uma quádrupla (I, Sol, m, OBJ), onde:

1) I é o conjunto de instâncias de *p*;

2) Sol é uma função que associa a qualquer instância $x \in I$ o conjunto S_x=Sol(x) de soluções viáveis de x;
3) m é a função medida que, para cada instância $x \in I$, associa a cada solução viável $y \in S_x$, o número inteiro positivo $m_x(y)$, denotando a medida (ou valor) da solução y;
4) OBJ $\in$ {MIN, MAX}, caracteriza se o objetivo do problema é de maximização ou minimização.

Dada uma instância $x \in I$, o objetivo do problema de otimização é encontrar uma *solução ótima*, isto é, uma solução viável $y \in S_x$, tal que

$$m_x(y) = OBJ\{ m_x(z) / z \in S_x \}$$

A medida de qualquer solução ótima y*(x), para a instância $x \in I$, é representada por m*(x).

É importante observar que um problema de otimização *p* pode ser associado, naturalmente, aos três seguintes problemas, correspondendo às diferentes maneiras de abordar sua solução:

Problema Construtivo (p_C) – Dada uma instância $x \in I$, obter uma solução ótima $y^* \in S_x$ e sua medida m*(x).

Problema de Avaliação (p_A) - Dada uma instância $x \in I$, obter o valor ótimo m*(x).

Problema de Decisão (p_D) - Dada uma instância $x \in I$ e um inteiro positivo $k \in \mathbf{Z}^+$, decidir se $m^*(x) \geq k$, se p é um problema de maximização (ou $m^*(x) \leq k$, se p é um problema de minimização).

Segundo D. Hochbaum [HOC 97], as variações na maneira de apresentar um problema pode influenciar significativamente em termos dos resultados da aproximação.

O último desses três problemas é chamado de linguagem associada ao problema p. Significa que, enquanto um problema de otimização pede para computar uma função, o problema de decisão associado é nada mais que um problema de "linguagem de pertinência". Isto é, para um dado par (x,k) a resposta pode ser "sim" ou "não".

A relevância de tal problema reside no fato de que o ponto central no desenvolvimento da Complexidade Computacional foi a introdução da teoria da NP-completude, a qual trabalha com linguagens. A bem conhecida conjetura P≠NP estabelece que, se uma linguagem é NP-completa para reduções polinomiais, então ela não pode ser reconhecida em tempo polinomial [JOH 90]. Em particular, se for possível provar que a linguagem p_D, associada ao problema de otimização *p*, é NP-completa, então tem-se um resultado de limite inferior para a complexidade do problema *p*: a solução ótima não pode ser computada deterministicamente em tempo polinomial, a menos que P=NP. Neste caso, diz-se que o problema *p* é NP-difícil.

Conforme definição de R. Motwani [MOT 92], se um problema de decisão NP-difícil *q* é polinomialmente redutível ao problema de computar a solução de um problema de otimização *p*, então o problema *p* é NP-difícil. Entretanto, a definição de NP-dificuldade

usando a noção de Turing-redutibilidade (que será definida na próxima seção), permite uma aplicabilidade mais ampla para o termo.

Para caracterizar a complexidade de problemas de otimização, de modo a obter uma classificação adequada, várias abordagens podem ser seguidas. Sob um ponto de vista mais direto, conforme [AUS 99], considera-se a medida de tempo necessária para resolver construtivamente o problema dado, estendendo-se para problemas de otimização a teoria desenvolvida para problemas de decisão.

Assim, em analogia aos problemas de decisão, são definidas a seguir, as principais classes de problemas de otimização.

3.2 Classes de Problemas de Otimização

Além de caracterizar os problemas de otimização considerados tratáveis, relativamente às suas versões construtivas, esta seção visa o estudo daqueles problemas de otimização que encontram-se na fronteira entre a tratabilidade e intratabilidade. Tais problemas são chamados de NPO, numa contrapartida natural de otimização para os problemas de decisão da classe NP.

3.2.1 Classe NPO

Um problema de otimização p = (I, Sol, m, OBJ) pertence a classe NPO se:

1) O conjunto de instâncias I é reconhecível em tempo polinomial;
2) Dada uma instância $x \in I$, a questão $y \in S_x$ (S_x = Sol(x)) é decidível em tempo polinomial e, além disso, existe um polinômio q tal que $|y| \leq q(|x|)$;
3) A função medida m é computável em tempo polinomial.

3.2.2 Classe PO

Um problema de otimização p = (I, Sol, m, OBJ) pertence a classe PO se ele está na classe NPO e se existe um algoritmo A computável em tempo polinomial tal que, para cada instância $x \in I$, retorna uma solução ótima $y^* \in S_x$, junto com o valor $m^*(x)$.

Observa-se que, para qualquer problema de otimização na classe NPO, o problema de decisão associado pertence à classe NP, enquanto que a classe PO consiste de todos os problemas de NPO cuja linguagem associada está na classe P. Além disso, a questão "PO = NPO ?" está estritamente relacionada com a questão "P = NP ?", já que uma resposta positiva para a primeira implicaria numa resposta positiva para a segunda, e vice-versa.

Praticamente, todos os problemas de otimização interessantes pertencem à classe NPO. Além de todos os problemas de otimização pertencentes à classe PO (tais como o *Problema do Caminho Mínimo*), vários outros problemas de grande relevância prática estão em NPO: os problemas *Cobertura Mínima de Vértices, Caixeiro Viajante Mínimo* e *Coloração Mínima de Grafos*, muitos outros problemas de otimização de grafos, a maioria dos problemas de empacotamento e escalonamento e, ainda, as formulações gerais de programação linear binária e inteira [AUS 99].

Entretanto, embora pertençam à NPO, não é conhecido se esses problemas pertençam à classe PO, já que nenhum algoritmo de tempo polinomial para eles é conhecido (e é provável que não exista). Para alguns deles, não somente a solução exata parece ser extremamente complexa de ser obtida, mas até mesmo conseguir boas soluções aproximadas pode ser computacionalmente difícil. Na verdade, para todos os problemas de otimização em NPO - PO, a complexidade intrínsica não é conhecida precisamente. Assim como acontece com os problemas de decisão em NP - P, nenhum algoritmo de tempo polinomial foi encontrado, nem é conhecida uma prova de intratabilidade para tais problemas.

Para identificar os mais difíceis problemas de otimização, é introduzido um tipo de redutibilidade mais geral do que a redução polinomial usada na teoria da NP-completude: a *Turing-redução*. Basicamente, neste tipo de redução, modela-se a possibilidade de, ao escrever um programa para resolver um determinado problema, usar subrotinas para resolver outro problema, tantas vezes quanto for requerido. Tal situação é formalizada em Teoria da Complexidade através do uso de máquinas de Turing com oráculo.

A definição apresentada abaixo segue [AUS 99]:

Definição 3.2: *Turing-redução*

Um problema *p* é Turing-redutível a um problema *q* se existe um algoritmo de tempo polinomial R que resolve *p*, tal que R pode acessar um algoritmo *oráculo* que resolve *q*.

A figura Fig. 3.1 mostra um esquema de aplicação da Turing-redução.

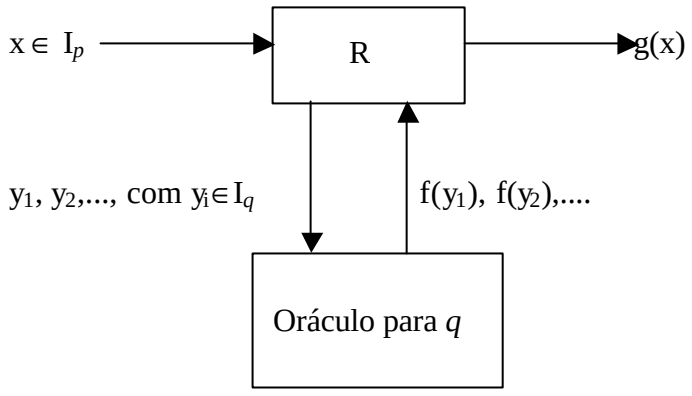

FIGURA 3.1- Esquema de uso da Turing-redução

3.2.3 Classe NP-difícil

Um problema de otimização *p* é chamado NP-difícil se, para cada problema de decisão $q \in$ NP, *q* é Turing-redutível a *p*.

Isto significa que o problema *q* pode ser resolvido em tempo polinomial por um algoritmo usando um oráculo que, para cada instância x de *p*, retorna uma solução ótima y*(x) juntamente com seu valor m*(x).

Portanto, um problema é NP-difícil se ele é pelo menos tão difícil de resolver quanto qualquer problema em NP. Conforme comentário anterior, muitos problemas interessantes são NP-difíceis. Isto acontece, por exemplo, com todos os problemas de otimização na classe NPO cujos problemas de decisão associados são NP-completos. E, assim como ocorre com estes últimos, o fato de um problema pertencente à classe NPO ser NP-difícil, o coloca no mais alto nível de complexidade dentro da classe NPO.

3.3 Algoritmos Aproximativos

Em face a um problema de otimização NP-difícil, um caminho natural é considerar algoritmos que não produzem soluções ótimas, mas soluções garantidamente próximas da ótima. Estes algoritmos que fornecem soluções que nunca diferem da solução ótima mais que um percentual especificado, apresentando uma qualidade garantida *a priori*, são chamados *algoritmos aproximativos* e foram definidos inicialmente no trabalho de D. Johnson [JOH 74], onde são apresentados os primeiros estudos formais sobre aproximabilidade de problemas de otimização.

3.3.1 Erro Relativo e Razão de Qualidade

Para expressar a qualidade de uma solução aproximada, são definidas a seguir as noções de *erro relativo* e *razão de qualidade* de uma solução.

Definição 3.3: *Erro relativo e Razão de qualidade*
Dado um problema de otimização *p*, para qualquer instância x de *p* e para qualquer solução viável y de x, o *erro relativo* de y em relação a x, é definido como

$$E(x,y) = \frac{|\, m^*(x) - m(x,y) \,|}{m^*(x)}$$

e a *razão de qualidade* de y em relação a x, como

$$R(x,y) = \max\left\{ \frac{m(x,y)}{m^*(x)} \;,\; \frac{m^*(x)}{m(x,y)} \right\}$$

O erro relativo (respectivamente, razão de qualidade) é sempre um número maior ou igual a zero (respectivamente, um) e está tão próximo de zero (respectivamente, um) quanto mais próxima da solução ótima está a solução viável y. Observa-se, ainda, que para problemas de minimização, o erro relativo pode ser qualquer número real não negativo, enquanto que para problemas de maximização, nunca será maior que um.

Estas duas medidas de qualidade de uma solução estão relacionadas através da desigualdade

$$\frac{R(x,y)-1}{R(x,y)} \leq E(x,y) \leq R(x,y)-1$$

3.3.2 Tipos de Algoritmos Aproximativos

Os algoritmos aproximativos considerados nesta seção caracterizam-se por apresentarem aproximação com qualidade garantida. A classificação de problemas computacionalmente difíceis está diretamente relacionada com o tipo de algoritmo aproximativo que eles admitem. Alguns problemas permitem eficientes e arbitrariamente bons algoritmos aproximativos, enquanto outros apresentam intrínsicas limitações à aproximabilidade.

Definição 3.4: *Algoritmo r(n)-aproximativo*

Dado um problema de otimização *p*, seja A um algoritmo que, para qualquer instância x de *p*, tal que $S_x \neq \emptyset$, retorna uma solução viável A(x) em tempo polinomial. Dada uma função arbitrária r: $Z^+ \rightarrow [0,+\infty)$, diz-se que A é um *algoritmo r(n)-aproximativo* para o problema *p*, se o erro relativo E(x, A(x)) da solução viável A(x), para qualquer instância x, verifica a seguinte desigualdade

$$E(x, A(x)) \leq r(n)$$

onde n é o tamanho da instância x.

Definição 3.5: *Algoritmo* ε*-aproximativo*

Dado um número racional $\varepsilon>0$, diz-se que A é um *algoritmo* ε*-aproximativo* para o problema *p* se, para qualquer instância x de *p*, o erro relativo E(x, A(x)) é limitado por ε, isto é

$$E(x, A(x)) \leq \varepsilon$$

Definição 3.6: *Problema aproximável*

Um problema de otimização é dito *aproximável* se ele admite um algoritmo ε-aproximativo, para algum número racional $\varepsilon>0$.

Alguns problemas aproximáveis apresentam ainda a qualidade adicional de admitirem algoritmos aproximativos de tempo polinomial com erro relativo tão pequeno quanto se queira. Naturalmente, o preço que se paga para obter soluções aproximadas de alta qualidade, será, provavelmente, um tempo de computação muito maior. Estas noções são capturadas através da definição de *esquemas aproximativos*.

3.3.3 Esquemas Aproximativos

Considerando um espectro de aproximação, problemas que admitem esquemas aproximativos encontram-se numa posição de quase fronteira com os problemas tratáveis.

Definição 3.7: *Esquema aproximativo polinomial*

Seja P um problema NPO. Um algoritmo A é dito ser um *esquema aproximativo polinomial* para P se, para qualquer número racional $\varepsilon>0$, A é um algoritmo ε-aproximativo para o problema P, com tempo polinomial no tamanho da entrada.

Definição 3.8: *Esquema aproximativo totalmente polinomial*

Se, além de ser um esquema aproximativo polinomial, o algoritmo A é de tempo polinomial também no inverso de ε, então ele é chamado de *esquema aproximativo totalmente polinomial*.

O termo "esquema" usado aqui justifica-se porque o algoritmo resulta num algoritmo aproximativo, para cada $\varepsilon>0$ fixado. Observa-se, no entanto, que a definição de um esquema aproximativo polinomial não limita a complexidade de tempo como polinomial com respeito a $1/\varepsilon$. Portanto, computações com valores de ε muito pequenos podem ser, praticamente, impossíveis. Isto significa que esquemas aproximativos totalmente polinomiais são os algoritmos ideais para resolver problemas de otimização NP-difíceis. Infelizmente, são poucos os problemas que admitem este tipo de algoritmo.

Quando se trata com um problema de otimização NP-difícil, a seguinte questão costuma ser mais importante: existem algoritmos ε-aproximativos para este problema ? E, se existem, quão pequeno pode ser ε ?

Enquanto que todos os problemas NP-completos conhecidos são redutíveis um para o outro, constata-se que os correspondentes problemas de otimização na classe NPO apresentam comportamentos muito diversificados em relação à aproximação, podendo ser completa ou parcialmente aproximáveis, ou ainda não ser aproximável para qualquer $\varepsilon>0$, a menos que P=NP.

Na próxima seção são introduzidas as principais classes de aproximação para problemas de otimização.

3.4 Classes de Aproximação

O comportamento diferenciado de problemas de otimização computacionalmente difíceis, em relação às suas propriedades de aproximabilidade, é capturado através da definição de *classes de aproximação*, isto é, classes de problemas de otimização que compartilham propriedades de aproximação similares.

3.4.1 Classe APX

É formada por todos os problemas em NPO tal que, para algum $\varepsilon>0$, admitem um algoritmo ε-aproximativo.

3.4.2 Classe PAS

Consiste de todos os problemas em NPO, que admitem esquema aproximativo polinomial.

3.4.3 Classe FPAS

Consiste de todos os problemas em NPO, que admitem esquema aproximativo totalmente polinomial.

O quadro apresentado abaixo mostra uma parte do estado atual do mundo NPO, exemplificando a pertinência de problemas em cada classe. Os problemas apresentados como exemplos em cada classe foram retirados do livro de Ausiello et al. [AUS 99] e são considerados paradigmas na área de Complexidade Computacional, já que conhecer as propriedades computacionais e estudar o projeto de algoritmos aproximativos para tais problemas, permite entender como resolver, de uma maneira aproximada, centenas de diferentes, embora relacionados, problemas de otimização que são encontrados nas aplicações diárias.

NPO	Caixeiro Viajante Mínimo Conjunto Independente Máximo Clique Máxima Coloração Mínima de Grafos
APX	Empacotamento Binário Mínimo Satisfabilidade Máxima Cobertura Mínima de Vértices Corte Máximo
PAS	Partição Mínima
FPAS	Mochila Máxima
PO	Caminho Mínimo

QUADRO 3.1- Exemplos de Problemas nas Classes de Aproximação

Centenas de outros problemas de otimização, cujo status de tratabilidade continua desafiando cientistas da área, fazem parte do compêndio atualizado eletronicamente no endereço dado pela referência [CRE 95].

Não custa lembrar, que a teoria está apoiada sobre a conjetura $P0 \neq NPO$.

Sob essa conjetura, estas classes formam então uma hierarquia, cujos níveis correspondem aos graus de dificuldade de aproximação.

Conforme ilustrado na Fig. 3.2, as seguintes inclusões entre as classes de problemas de otimização se verificam:

$$PO \subseteq FPAS \subseteq PAS \subseteq APX \subseteq NPO$$

Através de técnicas similares àquelas usadas para problemas de decisão, é possível mostrar que todas as inclusões são estritas, a menos que P=NP [BOV 94].

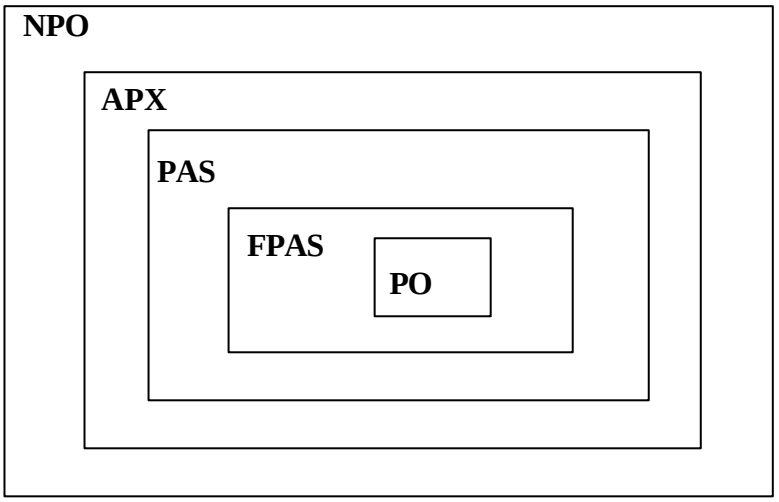

FIGURA 3.2 - Representação das classes de Problemas de Otimização

A partir desse fato, é natural o interesse por problemas de otimização tais que, por exemplo, pertençam à classe APX mas não estejam na classe PAS. Em analogia a questão P versus NP, a noção de *redutibilidade preservando aproximação* ordena problemas de otimização com respeito ao grau de dificuldade de aproximação, identificando os elementos maximais em APX, em relação a essa ordem. Tais problemas são chamados APX-completos e, conforme [AUS 99], são importantes por duas razões: por um lado, pode-se tentar resolver a questão APX versus PAS, provando que um problema APX-completo P pertence a PAS e, por outro lado, um resultado de APX-completude pode prevenir a perda de tempo na busca por um esquema de aproximação polinomial para um problema específico.

3.5 Reduções Preservando Aproximação

A definição de uma apropriada noção de redutibilidade entre problemas de otimização, permite estabelecer formalmente o significado de que um problema de otimização é mais difícil de aproximar do que outro.

Em geral, para a Teoria da Complexidade, uma redução de um problema A para um problema B especifica algum procedimento para resolver o problema A através de um algoritmo que resolve B.

Conforme Trevisan [TRE 97], no contexto da aproximação, a redução deve garantir que uma solução aproximada de B possa ser usada para obter uma solução aproximada para A. Para isto, são necessárias duas funções: uma função f que associa uma instância x de A para uma instância x’ de B, e também uma outra função g, associando de volta uma solução aproximada y’ (da instância x’) de B, para uma solução aproximada y (da instância x) de A.

Além disso, numa redução preservando aproximação, algumas relações entre as medidas de qualidade dos algoritmos devem ser satisfeitas. Dependendo de como as propriedades de aproximação desejadas são garantidas, diferentes tipos de reduções preservando aproximação têm sido definidas.

3.5.1 AP-redução

Embora muitos tipos de reduções-preservando-aproximação tenham sido definidos na literatura, a chamada AP-redução é suficientemente geral em comparação com as outras, apresentando a propriedade de estabelecer uma relação linear entre as razões de qualidade dos algoritmos. Esta propriedade é importante, pois preserva a pertinência em todas as classes de aproximação.

Definição 3.9: ***AP-redução***

Sejam A e B dois problemas em NPO. A é dito ser AP-redutível a B, denotando-se por $A \leq_{AP} B$, se existem duas funções f e g e uma constante positiva $\alpha \geq 1$, tais que:

1) Para qualquer $x \in I_A$ e para qualquer racional r >1, $f(x,r) \in I_B$
2) Para qualquer $x \in I_A$ e para qualquer racional r >1, se $Sol_A(x) \neq \emptyset$ então $Sol_B(f(x,r)) \neq \emptyset$
3) Para qualquer $x \in I_A$, para qualquer racional r >1, e para qualquer $y \in Sol_B(f(x,r))$, tem-se que $g(x,y,r) \in Sol_A(x)$.
4) f e g são computáveis por dois algoritmos A_f e A_g, respectivamente, com tempo de execução polinomial para qualquer racional r fixado.
5) Para qualquer $x \in I_A$, para qualquer racional r >1, e para qualquer $y \in Sol_B(f(x,r))$,

$$R_B(f(x,r),y) \leq r \Rightarrow R_A(x,g(x,y,r)) \leq 1+\alpha(r-1)$$

A AP-redução (AP significando “approximation preserving”) foi definida no artigo [CRE 96], sendo tema da tese de doutorado de L. Trevisan [TRE 97]. Em ambos os trabalhos são analisados também os principais tipos de reduções que apareceram na literatura. Entre estas estão: S-redução (strict reduction), L-redução (linear reduction), E-redução (error reduction), P-redução (PAS-preserving reduction) e várias outras.

A AP-redução mostrou-se mais robusta que as outras reduções, podendo ser usada para tantas classes de aproximação quantas possíveis, e ainda mantendo as propriedades de todas as outras. Um esquema ilustrando o uso desta redução é mostrado na Fig. 3.3.

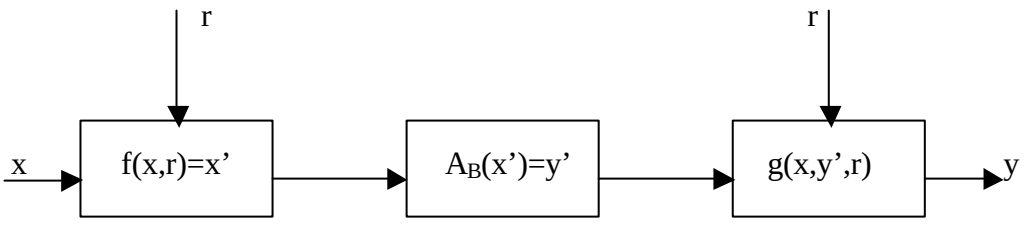

FIGURA 3.3 - Esquema de uso da AP-redução

3.5.2 Problema NPO-completo

Uma propriedade importante da AP-redução é a transitividade, que induz uma ordenação parcial entre problemas na mesma classe de aproximação. A noção de um problema completo é formalmente equivalente aquela de um elemento maximal em relação a esta ordem.

De modo análogo ao estudo da NP-completude, os problemas NPO-completos são considerados os mais difíceis na classe NPO, no sentido que nenhum problema NPO-completo pertence à classe APX, a menos que P = NP.

Definição 3.10: *Problema NPO-completo*

Um problema de minimização p é dito NPO-completo se $p \in$ NPO e, para qualquer outro problema de minimização $p' \in$ NPO, tem-se que $p' \leq_{AP} p$.

Um resultado similar ao Teorema de Cook, provando que a classe NP-completa não é vazia (isto é, contém o problema da Satisfabilidade), é obtido com o problema Satisfabilidade Mínima para a classe NPO-completa.

O problema Satisfabilidade Mínima - MINSAT é definido de modo similar ao problema Satisfabilidade, mas sendo um peso dado por um número inteiro não-negativo associado a cada variável. Neste caso deseja-se encontrar uma atribuição de valores satisfatória que minimize a soma dos pesos das variáveis verdadeiras. Uma prova de que este problema é NPO-completo é apresentada em [BOV 94].

A partir de um primeiro problema reconhecido como NPO-completo, é possível provar a NPO-completude de outros problemas de minimização. Isto deve-se ao fato que a AP-redução, assim como a redução polinomial, é *transitiva.* Observa-se, no entanto que provar NPO-completude parece ser, em geral, mais difícil que provar NP-completude, devido à redutibilidade ser mais restritiva. Muitos problemas NP-completos admitem algoritmos aproximativos, mas não podem ser NPO-completos, a menos que se prove que P = NP.

3.6 Comentários Finais

Neste capítulo apresentou-se uma breve visão do ambiente de inserção do trabalho, objetivando a compreensão dos diferentes modelos semânticos associados à questão estrutural de problemas de otimização. Em particular, a partir das noções de aproximabilidade e redutibilidade de problemas de otimização, foi possível identificar na Teoria dos Domínios de Scott e na Teoria das Categorias, os modelos que fundamentam matematicamente este trabalho.

A classificação de problemas de otimização em classes de aproximação disjuntas, sob a conjetura P $\neq$ NP, definindo uma hierarquia, cujos níveis correspondem aos graus de dificuldade de aproximação, é outra questão fundamental no entendimento de uma abordagem semântica para a complexidade estrutural de problemas de otimização.

No próximo capítulo considera-se um algoritmo aproximativo para um problema de otimização genérico como o principal objeto de estudo, estruturando-se matematicamente o conjunto de algoritmos aproximativos para tal problema como uma ordem parcial, no sentido proposto originalmente por Scott na construção da estrutura chamada *domínio*[SCO 72].

4 Algoritmos Aproximativos e a Teoria dos Domínios

O objetivo principal deste capítulo é a introdução de algumas idéias na direção de se obter uma formalização para a Teoria dos Algoritmos Aproximativos, buscando um modelo que apresente bases teóricas para desenvolver as propriedades desejadas para estes algoritmos, de tal forma que seja possível garantir uma solução satisfatoriamente próxima da solução exata, em tempo significativamente menor que o algoritmo exato.

Em vista dos resultados sobre a intratabilidade de problemas discutidos na primeira parte do presente trabalho, justifica-se o investimento na pesquisa sobre algoritmos aproximativos como uma alternativa para o tratamento de problemas considerados intratáveis.

A Teoria dos Domínios de Scott, considerada hoje como parte integrante da Teoria da Ciência da Computação, possivelmente mostra uma saída para o problema exposto. A idéia básica presente na teoria proposta originalmente por Scott [SCO 72] é a de *aproximação finita.* Assim como um círculo pode ser pensado como o limite de aproximações através de polígonos, Scott afirma que existe uma teoria geral de aproximações finitas e que muitos objetos podem ser considerados como *limites* de aproximações. Nesta teoria, a relação de aproximação seria dada num sentido qualitativo. Matematicamente, a teoria está baseada na simples idéia que a relação de aproximação seria uma ordenação parcial sobre aquele tipo de estrutura onde se possa tomar limites.

Neste sentido, propõe-se a construção de um domínio baseado na noção de *solução aproximada* para um dado problema. Tal domínio é construído através da completação por ideais, a partir da relação de ordem induzida por estas aproximações, para um determinado problema. Desta maneira, pensando no conjunto de algoritmos aproximativos para um problema como uma estrutura ordenada, acredita-se ser possível usar apropriadamente a Teoria dos Domínios para obter uma formalização da Teoria dos Algoritmos Aproximativos.

O presente capítulo inicia com um estudo dos principais aspectos desenvolvidos na Teoria dos Domínios, onde se procura resgatar as idéias seminais de Scott na construção do conceito de um domínio [LEA 97a].

A seguir define-se o conjunto dos algoritmos aproximativos para um problema de otimização genérico, como uma estrutura parcialmente ordenada, em termos dos respectivos requerimentos de qualidade dos algoritmos. Então, focaliza-se sobre as classes de aproximação APX, PAS e PTAS, investigando a aplicabilidade das noções originais da Teoria dos Domínios, tais como consistência, completude e algebricidade, para cada classe.

As idéias desenvolvidas neste capítulo foram introduzidas em [LEA 97b, LEA 98].

4.1 A Teoria dos Domínios

Os conceitos e resultados explorados nesta seção, seguem as notações de Stoltenberg-Hansen et alli [STO 94], que apresentam a Teoria dos Domínios no sentido de uma teoria matemática formalmente estabelecida. Segundo os autores, um domínio é uma estrutura que

modela as noções de aproximação e de computação. Esta é a razão porque a Teoria dos Domínios é uma parte muito usada na Teoria da Ciência da Computação.

Na presente seção, o conceito de *domínio* é construído matematicamente de maneira que seja uma estrutura cartesiana fechada, isto é, fechada em relação à construção do seu espaço de funções. Nesta construção é usada, basicamente, a teoria dos conjuntos parcialmente ordenados, enriquecendo-se as estruturas conhecidas por cpo's, até chegar no conceito de domínio propriamente dito.

É importante ressaltar que este é um dos aspectos em que se pode chegar ao conceito de domínio. Esta é uma *representação algébrica* desta estrutura. Conforme referenciado por [STO 94], duas outras representações podem ser dadas através de sistemas de informação e de espaços formais. A primeira é uma *representação lógica*, baseada numa relação de 'entailment', enquanto que a segunda é uma *representação topológica* baseada numa relação formal de inclusão entre vizinhanças.

Um breve histórico das origens e concepções que levaram à criação da Teoria dos Domínios é apresentado a seguir.

4.1.1 Origem

A Teoria dos Domínios também conhecida como Domínios de Scott, foi desenvolvida no início da década de setenta, com o trabalho de D. Scott e Y. Ershov.

Originalmente esta teoria foi criada para dar semântica à linguagem de programação, ou seja, a proposta essencial da Teoria dos Domínios é estudar classes de espaços que podem ser usados para dar semântica para definições recursivas.

Em geral uma linguagem de programação apresenta uma importante característica: *um programa pode ter outro programa como argumento.*

Conforme Abramski [ABR 87], a chave da construção de Scott foi justamente a solução da chamada *equação de domínio*:

"Que estrutura D satisfaz a equação $[D \rightarrow D] \cong D$?'

Dizendo de outro modo, Scott [SCO 82] construiu matematicamente uma estrutura isomorfa ao seu próprio espaço de funções. Na linguagem de categorias, tal estrutura é uma *categoria cartesiana fechada*.

O ponto de partida para atingir o conceito de um *domínio de Scott* é uma ordem parcial que vai sendo enriquecida com propriedades tais como completude, algebricidade e consistência, de forma a se obter uma estrutura que modele matematicamente as noções de aproximação e computabilidade.

4.1.2 Conceitos Básicos

Seja então D um conjunto, $\sqsubseteq$ uma relação de ordem parcial sobre D, e $\perp$ o menor elemento de D em relação a esta ordenação. Computacionalmente, uma ordem parcial pode ser pensada como uma relação de aproximação. Portanto $x \sqsubseteq y$ pode ser interpretado como: x é uma aproximação de y ou y contém pelo menos tanta informação quanto x. O menor elemento $\perp$, quando existe, representa ausência total de informação.

Definição 4.1: ***Conjunto Dirigido***

Um subconjunto $A \subseteq D$ é *dirigido* se $A \neq \emptyset$ e sempre que $x, y \in A$ existe $z \in A$ tal que $x \sqsubseteq z$ e $y \sqsubseteq z$.

O supremo de um conjunto dirigido A será representado por $\bigsqcup A$, se existir. Usualmente escreve-se $x \sqcup y$ para $\bigsqcup\{x, y\}$.

O exemplo mais simples de conjunto dirigido é uma *cadeia*, isto é, um subconjunto de uma ordem parcial que é totalmente ordenado. Uma cadeia pode ser representada como uma sequência

$$x_1 \sqsubseteq x_2 \sqsubseteq ... \sqsubseteq x_n \sqsubseteq ...$$

De forma análoga, um conjunto dirigido A pode ser visto como uma sequência crescente generalizada, no sentido que a informação em A é consistente e dá crescentemente melhores informações. Isto significa que para cada $x, y \in A$ existe um elemento em A que é aproximado por ambos. Em particular x e y não podem conter informações contraditórias.

Definição 4.2: ***Ordem Parcial Completa*** **(CPO)**

D é uma *ordem parcial completa* (cpo), se qualquer conjunto dirigido $A \subseteq D$ possui supremo $\bigsqcup A$ em D.

Um cpo é então *completo* no sentido que cada sequência crescente generalizada A em D converge para um elemento de D, exatamente o seu supremo $\bigsqcup A$.

Cpo's podem ser considerados como modelos mínimos para a teoria da computação. Isto significa que a exigência mínima para um modelo de computação é a convergência representada por sua completude.

No entanto, apesar de constituírem uma categoria cartesiana fechada, os cpo's não satisfazem a 'equação de domínio', pois falham em relação `a computabilidade, onde se entende que objetos infinitos devem ser dados, de alguma forma coerente, como limites de suas aproximações finitas. A noção de elemento finito de um cpo é dada a seguir.

Definição 4.3: ***Elemento Finito***

Seja x um elemento do cpo $D = (D, \sqsubseteq, \perp)$. Diz-se que $x \in D$ é *finito* sss para todo conjunto dirigido $A \subseteq D$, se $x \sqsubseteq \bigsqcup A$ então existe $a \in A$ tal que $x \sqsubseteq a$.

Os elementos finitos num cpo são considerados como os elementos concretos sobre os quais se pode computar. Representa-se por D_c o conjunto dos elementos finitos de D.

Dado $x \in D$, o conjunto dos *aproximantes finitos* de x será representado por aprox(x). Isto é: $\text{aprox}(x) = \{a \in D_c / a \sqsubseteq x\}$.

Na classe de cpo's definida a seguir a noção de aproximação finita faz sentido: *cada elemento é o limite de suas aproximações finitas.*

Definição 4.4: ***Cpo Algébrico***

Um cpo D é algébrico se ele possui uma base de elementos finitos, isto é, para cada $x \in D$, aprox (x) = $\{a \in D_c / a \sqsubseteq x\}$ é dirigido e $x = \bigsqcup$ aprox (x).

A condição de ser algébrico para um cpo, garante que pode-se estender a computação para um elemento arbitrário, visto como o limite de suas aproximações finitas, definindo-se o resultado da computação como o limite dos resultados da computação sobre cada aproximação finita. Por esta razão os cpo's algébricos parecem satisfatórios de um ponto de vista computacional, pois permitem definições de computabilidade que estabelecem ligações com a Teoria das Funções Recursivas e possibilitam visualizar a estrutura de um domínio definida a seguir, como a completação de estruturas de informações finitas. Entretanto, cpo's algébricos ainda não satisfazem a 'equação de domínio' por uma razão muito importante: não é cartesiana fechada! Existem cpo's algébricos D e E tais que o cpo $[D \rightarrow E]$ não é algébrico.

Para se obter uma categoria cartesiana fechada exige-se mais um requerimento: o cpo além de ser algébrico deve ser *consistentemente completo.*

Definição 4.5: ***Conjunto Consistente***

Um conjunto $A \subseteq D$ é consistente se existe $x \in D$ tal que para cada $y \in A$, $y \sqsubseteq x$.

Isto significa que o conjunto A tem uma cota superior em D.

Expressando em termos de aproximações, um conjunto A é consistente se existe um elemento comum x que é aproximado por cada elemento de A. Em outras palavras, o conjunto A não contém informações contraditórias.

Definição 4.6: ***Cpo Consistentemente Completo***

Um cpo é consistentemente completo se $\bigsqcup A$ existe em D, sempre que $A \subseteq D$ é um conjunto consistente.

Depois desta definição, finalmente pode-se atingir aquela estrutura cartesiana fechada D que satisfaz a 'equação de domínio', e que é muito apropriadamente chamada *Domínio de Scott* .

Definição 4.7: ***Domínio de Scott***

Um *domínio de Scott* $D = (D; \sqsubseteq, \perp)$ é um conjunto parcialmente ordenado completo (cpo), algébrico e consistentemente completo.

- *Completo* : no sentido que cada conjunto dirigido $A \subseteq D$, considerado como uma sequência crescente generalizada, converge para um elemento em D, exatamente o seu supremo, representado por $\bigsqcup A$.
- *Algébrico :* por que cada elemento $x \in D$ é o supremo do conjunto dirigido formado por suas aproximaçõs finitas.
- *Consistentemente completo :* todo subconjunto de D que tem cota superior, isto é, que não possui informações contraditórias, possui uma menor cota superior.

Definição 4.8: ***Semi-reticulado condicionalmente superior com menor elemento*** **(CUSL)**

Uma ordem parcial P = (P, $\sqsubseteq$, $\perp$) com menor elemento $\perp$ é um *semireticulado condicionalmente superior com menor elemento* (cusl) se sempre que {a, b} $\subseteq$ P possui cota superior em P, então a $\sqcup$ b existe em P.

Definição 4.9: ***Ideal***

Seja P um cusl. Então I $\subseteq$ P é um *ideal* se:

i) $\perp \in$ I

ii) Se a $\in$ I e b $\sqsubseteq$ a, então b $\in$ I

iii) Se a, b $\in$ I então a $\sqcup$ b $\in$ I

Se a $\in$ P, o conjunto [a] = {b $\in$ P / b $\sqsubseteq$ a } é um ideal, chamado de *Ideal Principal* gerado pelo elemento a. Este ideal é o menor ideal contendo a.

O conjunto $\overline{\mathbf{P}}$ = {I $\subseteq$ P/ I é um ideal} é chamado de *completação de* P *por ideais.*

O teorema apresentado a seguir estabelece que, dado qualquer cusl P, pode-se obter uma estrutura de domínio através da completação de P por ideais. A prova deste teorema é apresentada detalhadamente em [STO 94].

Teorema 4.1.1

Seja P um cusl. Então a estrutura $\overline{\mathbf{P}}$ = ($\overline{\mathbf{P}}$, $\subseteq$, [$\perp$]) é um domínio, onde a ordenação é dada pela inclusão de conjuntos, e cujos elementos compactos são os ideais principais:

$$\overline{\mathbf{P}}_c = \{[a] / a \in P\}.$$

Reciprocamente, dado qualquer cusl P, existe um domínio D tal que D_c é isomorfo a P. Portanto, a menos de isomorfismo, D é completamente determinado através do seu conjunto de elementos finitos D_c. Isto é o que afirma o Teorema da Representação a seguir.

Teorema 4.1.2 (Teorema da Representação)

Seja D um domínio. Então $\overline{D}_c \cong D$.

4.2 Ordem de e-Aproximação

Nesta seção, propõe-se a construção de um domínio baseado na noção de aproximação da solução de um dado problema. Tal domínio é construído através da completação por ideais, a partir da relação de ordem induzida por estas aproximações, para um determinado problema. Desta maneira, pensando no conjunto de algoritmos aproximativos para um problema como uma estrutura ordenada, acredita-se ser possível usar apropriadamente a Teoria dos Domínios para obter uma formalização da Teoria dos Algoritmos Aproximativos.

4.2.1 Função de Qualidade de um Algoritmo

Definição 4.10: *Função de qualidade de um algoritmo*

Seja *p* um problema de otimização e $\mathbb{A}$ o conjunto dos algoritmos aproximativos para *p*. Dado $a \in \mathbb{A}$, define-se a *função de qualidade* do algoritmo *a*, simbolizada por ε_a, a função que associa a cada instância I de *p*, a razão de qualidade do algoritmo *a*, isto é

$$\varepsilon_a(I) = \frac{|\, m^*_p(I) - a(I) \,|}{m^*_p(I)},$$

para cada instância I de *p*.

Conforme observado na seção 3.3.1, os valores de $\varepsilon_a(I)$ variam entre 0 e 1, para cada instância I.

Na prática, o que se procura é um número real e que seja maior que $\varepsilon_a(I)$, para todo I, chamado *requerimento de qualidade* do algoritmo *a*. Inicialmente esse número será representado também por ε_a e tratado como um real simplesmente.

Observa-se que, caso exista tal número, o problema *p* é dito e-aproximável, ou, de modo equivalente, que *p* pertence à classe APX.

Definição 4.11:

Dados dois algoritmos *a*, *a'* $\in \mathbb{A}$, define-se a relação $a \sqsubseteq_\varepsilon a'$, que se lê " *a* é melhor que *a'*", em termos dos respectivos requerimentos de qualidade ε_a e $\varepsilon_{a'}$, por

$$a \sqsubseteq_\varepsilon a' \qquad \text{se, e somente se} \qquad \varepsilon_a \leq \varepsilon_{a'}$$

Observa-se, no entanto, que para ε_a ser uma caracterização do algoritmo é preciso identificar algoritmos com mesmo requerimento de qualidade.

Felizmente a relação

$$a \approx_\varepsilon a' \qquad \text{se, e somente se} \qquad \varepsilon_{a'} = \varepsilon_a$$

é uma relação de equivalência.

Toma-se, então, os representantes das classes de equivalência para representar $a \in \mathbb{A}$, incluindo-se em $\mathbb{A}$ o algoritmo $\perp_\varepsilon$(bottom), que representa o menor elemento para a relação $\sqsubseteq_\varepsilon$.

A relação definida acima, em termos dos requerimentos de qualidade dos algoritmos, é uma relação de ordem total, chamada *ordem de e-aproximação*.

Teorema 4.2.1:

Se o problema *p* está na classe APX, então a estrutura $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ é uma ordem parcial, mas não é completa, a menos que P=NP.

Prova: É fácil verificar que a estrutura $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ é uma ordem parcial (na verdade uma ordem total) já que é definida em termos dos números reais - totalmente ordenados. Entretanto a completude não é satisfeita. De fato, se *p* está na classe APX, então existe um número real positivo r tal que, para todo número real positivo ε, tal que *p* é ε-aproximável, tem-se que $\varepsilon > r$. Para provar a completude, devemos mostrar que, para todo conjunto dirigido $D \subseteq \mathbb{A}$ (isto é $\forall x, y \in D, \exists z \in D$ tal que $x \sqsubseteq_\varepsilon z$ e $y \sqsubseteq_\varepsilon z$), existe um algoritmo $a \in \mathbb{A}$ tal que $a = \bigsqcup D$.

Seja $D \subseteq \mathbb{A}$ o conjunto dirigido $D=\{x \in \mathbb{A} / \varepsilon_x > r\}$. Se $a = \bigsqcup D$, então

$$\varepsilon_a = \inf\{\varepsilon_x \in \mathbb{R} / x \sqsubseteq_\varepsilon a\} = r.$$

Portanto *a* não pertence ao conjunto $\mathbb{A}$, a menos que P=NP. ♦

Assim, pelo teorema acima, no caso do problema *p* pertencer à classe APX, não está garantida a completude da estrutura $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$, já que nem todo conjunto dirigido admite supremo.

Para verificar a condição de algebricidade da estrutura, precisa-se antes definir algoritmos finitos.

Definição 4.12: *Algoritmos finitos*

O algoritmo aproximativo $x \in \mathbb{A}$ é dito um algoritmo finito se e somente se, para todo conjunto dirigido $D \subseteq \mathbb{A}$, tal que $x \sqsubseteq_\varepsilon \bigsqcup D$, existe $d \in D$ tal que $x \sqsubseteq_\varepsilon d$.

O conjunto dos algoritmos finitos de $\mathbb{A}$ é denotado por $\mathbb{A}_c$.

Dado $a \in \mathbb{A}$, denota-se o conjunto de *algoritmos aproximantes finitos* para o algoritmo *a* por $\text{aprox}(a) = \{x \in \mathbb{A}_c / x \sqsubseteq_\varepsilon a\}$.

A primeira condição a ser verificada é se aprox(*a*) é dirigido. Sejam $x, y \in \mathbb{A}_c$. Então $x \sqcup y$ existe, pois um algoritmo dessa classe seria o que executa x, depois y e escolhe o resultado mais exato. Assim, tem-se que

$$x \sqsubseteq_\varepsilon (x \sqcup y) \quad \text{e} \quad y \sqsubseteq_\varepsilon (x \sqcup y)$$

Certamente $(x \sqcup y) \sqsubseteq_\varepsilon a$. Falta verificar se $(x \sqcup y) \in \mathbb{A}_c$. Seja $D \subseteq \mathbb{A}$ conjunto dirigido tal que $(x \sqcup y) \sqsubseteq_\varepsilon \bigsqcup D$. Então $x \sqsubseteq_\varepsilon \bigsqcup D$ e $y \sqsubseteq_\varepsilon \bigsqcup D$. Como $x, y \in \mathbb{A}_c$, existem $d_1, d_2 \in D$ tal que $x \sqsubseteq_\varepsilon d_1$ e $y \sqsubseteq_\varepsilon d_2$. Então $(x \sqcup y) \sqsubseteq_\varepsilon (d_1 \sqcup d_2)$. Sendo D dirigido, então existe $d \in D$ tal que $d_1 \sqsubseteq_\varepsilon d$ e $d_2 \sqsubseteq_\varepsilon d$. Logo $(x \sqcup y) \sqsubseteq_\varepsilon D$. Portanto $(x \sqcup y) \in \mathbb{A}_c$, o que permite concluir que aprox(*a*) é um conjunto dirigido.

Não se pode garantir entretanto, que $\bigsqcup \text{aprox}(a) = a$, pela mesma razão que justifica a sua não completude. Isto implica que a estrutura $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ não é, garantidamente, algébrica.

A discussão acima é resumida na seguinte proposição:

Proposição 4.2.1: Se o problema *p* está na classe APX, então a estrutura $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ não é cpo-algébrico.

Mas, por outro lado, pela definição da classe PAS (e FPAS), tem-se o seguinte resultado positivo:

Teorema 4.2.2:

Se o problema *p* está na classe PAS, então a estrutura $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ é uma ordem parcial completa, mas não é cpo-algébrico.

Entretanto a condição de consistência é satisfeita por esta estrutura, pois dado $\{a, b\} \subseteq \mathbb{A}_c$ consistente (limitado superiormente em $\mathbb{A}$), $\bigsqcup\{a, b\}$ existe em $\mathbb{A}$ e, além disso, a $\bigsqcup b = \bigsqcup\{a, b\}$, como foi visto anteriormente.

Uma condição, portanto, suficiente para $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ ser um domínio é que $\mathbb{A}$ seja o conjunto dos algoritmos desenvolvidos por um esquema aproximativo que garanta o seguinte:

"Para todo conjunto dirigido $D \subseteq \mathbb{A}$, $\exists\, a \in \mathbb{A}$ tal que $\varepsilon_a = \inf\{\varepsilon_x \in \mathbb{R} / x \in D\}$ e para todo $a \in \mathbb{A}$, $\bigsqcup \text{aprox}(a) = a$."

As demais condições a serem satisfeitas por um domínio, decorrem naturalmente da condição acima.

Teorema 4.2.3:

Se *p* está na classe APX, então $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ é um semi-reticulado condicionalmente superior (abreviado por CUSL).

Prova: A condição de consistência verificada anteriormente confere a $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ a condição de CUSL, isto é, todo subconjunto consistente de $\mathbb{A}$ possui supremo. ♦

Este último resultado garante as condições necessárias para a obtenção de uma estrutura de Domínio de Scott para o conjunto de algoritmos aproximativos de um dado problema ε-aproximável.

Antes disso, porém, são necessárias algumas definições.

4.2.2 Ideal Aproximativo

Definição 4.13: *Ideal aproximativo*

Seja $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ um CUSL. $I \subseteq \mathbb{A}$ é um *ideal aproximativo* se:

i) $\perp \in I$;

ii) se $a \in I$ e $b \sqsubseteq_\varepsilon a$ então $b \in I$; e

iii) se $a, b \in I$ então $a \sqcup b$ existe em $\mathbb{A}$ e $a \sqcup b \in I$

Definição 4.14: *Ideal principal*

Para $a \in \mathbb{A}$, define-se $[a] = \{b \in \mathbb{A} / b \sqsubseteq_\varepsilon a\}$ o *ideal principal* gerado por *a*.

Então, [*a*] consiste das aproximações de *a*, não somente as finitas, como em aprox(*a*).

$\overline{\mathbb{A}}$ denota o conjunto de todos os ideais contidos em $\mathbb{A}$.

O próximo resultado é uma adaptação de [STO 94] para a estrutura dos algoritmos aproximativos, onde é obtida a *completação por ideais* desta estrutura. Isto significa que o conjunto dos ideais de $\mathbb{A}$, com a relação de ordem dada pela inclusão de conjuntos e $\perp = [\perp]$, é um Domínio de Scott, onde os elementos finitos de $\overline{\mathbb{A}}$ são exatamente os ideais principais.

Teorema 4.2.4: $\overline{\mathbb{A}} = (\overline{\mathbb{A}}, \subseteq, [\perp]\}$ é um Domínio de Scott e $\overline{\mathbb{A}}_c = \{[a] / a \in \mathbb{A}\}$.

A partir deste ponto, pretende-se, futuramente, retomar a definição de ordem de ε-aproximação, considerando-a como a ordem parcial definida em termos da função de qualidade do algoritmo (ver Definição 4.1). Neste caso, observa-se que poderão existir algoritmos *a* e *b* tais que, para alguma instância I, *a* será melhor que *b* e, para outra instância I', *b* será melhor que *a*.

Ainda, sob este ponto de vista "local" dos algoritmos aproximativos para um determinado problema de otimização, resta estabelecer uma noção adequada de *função contínua*, permitindo a computabilidade em tempo polinomial, senão da solução exata, pelo menos de uma solução tão boa quanto se queira para esse problema, em relação ao requerimento de qualidade desses algoritmos.

Além disso, num ponto de vista mais "global", um aspecto importante a ser abordado é o significado da *redutibilidade preservando aproximação* dentro da Teoria dos Domínios.

4.3 Comentários Finais

Nesta seção apresentou-se uma estruturação *interna* para o conjunto dos algoritmos aproximativos para um problema de otimização genérico, caracterizando as classes de aproximação no contexto da Teoria dos Domínios.

Iniciando pela definição de uma ordenação parcial em termos dos requerimentos de qualidade dos algoritmos, a seguir procedeu-se à investigação das propriedades dessa ordem parcial, em relação à aplicabilidade das noções originais da Teoria dos Domínios, tais como consistência, completude e algebricidade, para cada classe.

Em decorrência do aprofundamento no estudo e consequente avanço no entendimento do comportamento estrutural dos problemas de otimização com respeito à questão da aproximabilidade, a partir deste ponto optou-se por ampliar a investigação desenvolvida no presente trabalho, de modo a considerar também o relacionamento dos problemas entre si, sob um ponto de vista *externo*, seguindo uma abordagem categorial.

No próximo capítulo serão investigadas as relações entre as classes de aproximação através de uma abordagem dentro da Teoria das Categorias.

5 Categorias dos Problemas de Otimização

Este capítulo apresenta uma abordagem categórica para problemas de otimização, num processo de *especialização* – no sentido do reconhecimento de que uma estrutura particular é uma instância de um fenômeno mais geral, de acordo com a interpretação dada por Goldblat [GOL 84]. Uma versão preliminar desta idéia foi apresentada primeiramente em [LEA 2000]].

Após investigar a estruturação *interna* para os objetos, mediante a caracterização das classes de problemas em relação às propriedades de aproximação de seus membros, no contexto da Teoria dos Domínios, no presente capítulo pretende-se relacionar os objetos entre si, num ponto de vista *externo*, essencialmente categorial.

Aparentemente, Teoria das Categorias e Ciência da Computação constituem dois campos científicos completamente diferentes. Na realidade, não só possuem muito em comum como são enriquecidos mutuamente a partir de abordagens de um campo sobre o outro, na visão de Menezes e Haeusler [MEN 2001]].

A Teoria das Categorias pode ser considerada atualmente como uma das mais valiosas ferramentas em Computação Científica, especialmente na área de Informática Teórica. Segundo Asperti e Longo [ASP 91], um dos aspectos fundamentais que justificam o uso de Teoria das Categorias, é a premissa de que todo tipo de objeto estruturado matematicamente vem equipado com uma noção de transformação ou construção "aceitável", isto é, um morfismo que preserva a estrutura do objeto.

Neste contexto, a noção de *redução* entre problemas de otimização aparece, naturalmente, no sentido conceitual de *morfismo* entre objetos. Sob esta perspectiva, define-se neste capítulo as categorias dos problemas de otimização, cujos objetos são os problemas de otimização e cujos morfismos são as reduções entre tais problemas.

Antes, porém, é visto um pouco da história e fundamentação da Teoria das Categorias, seguida de uma breve introdução à Teoria de Universais, devida à Ellerman [ELL 88], com o objetivo de formalizar a noção de *universalidade* – um dos conceitos mais importantes para a definição das diversas construções categóricas.

As idéias apresentadas neste capítulo constituem o tema do trabalho apresentado no congresso internacional EUROCAST'2001 [LEA 2001]], com a respectiva publicação do artigo completo na série *Lecture Notes Computer Science*, sob o número 2178, referenciado neste volume por [LEL 2001]].

5.1 Teoria das Categorias

Esta seção apresenta um breve histórico do aparecimento da Teoria das Categorias, motivando seu uso em Ciência da Computação e introduz alguns conceitos básicos necessários para o entendimento do restante do capítulo, além de fixar a notação utilizada.

5.1.1 Origem

Barr & Wells [BAR 90] iniciam seu livro com uma vaga referência sobre Teoria das Categorias:

> *"There are various view on what category theory is about, and what it is good for."*

No entanto, a frase acima adquire sentido no decorrer de sua exposição. Segundo os autores, a Teoria das Categorias é um ramo relativamente novo da matemática, tendo sido criada por S. Eilenberg e S. MacLane em 1945, com origem na topologia algébrica e tendo por finalidade descrever os diversos conceitos estruturais de diferentes campos da matemática de uma forma uniforme.

Num primeiro momento, a Teoria das Categorias pode ser vista como uma generalização da álgebra de funções. Nesse contexto, a principal operação sobre funções é a de composição. Na realidade, numa visão mais abrangente, uma categoria é uma estrutura abstrata, constituída de objetos e setas entre estes objetos (chamados morfismos), com a propriedade fundamental de composicionalidade das setas.

Desde então, esta teoria tem influenciado muitas áreas, como uma forma revolucionária de entendimento e abordagem. De fato, a teoria das categorias providencia uma bagagem de conceitos (e teoremas sobre estes conceitos) que formam uma abstração de muitos conceitos concretos em diversos ramos da matemática, incluindo a Ciência da Computação.

Portanto, não será nenhuma surpresa se os conceitos de teoria das categorias formem uma abstração de muitos conceitos pertinentes à área de Complexidade Estrutural.

Mas, isto justifica a opção por uma abordagem categórica neste trabalho?

Uma possível resposta pode ser encontrada nas argumentações apresentadas por Menezes e Haeusler [MEN 2001]] em sua recente publicação da série Livros Didáticos do Instituto de Informática, na UFRGS, justificando o uso de Teoria das Categorias em Ciência da Computação. Segundo os autores, uma das características importantes que se destacam no uso de tal teoria é *a independência de implementação*, garantida a partir de um tratamento abstrato de entes primitivos (objetos e setas). A *herança de resultados e comparação da expressividade de formalismos* é outra característica distinguida na teoria, obtida através da possibilidade de construir categorias a partir de outras categorias existentes, bem como passar de um tipo de estrutura para outro (via funtores e transformações naturais). O conhecimento de Teoria das Categorias pelo cientista de computação, possibilita a aplicação dos conceitos básicos da teoria como uma ferramenta matemática unificada para a investigação científica, abordando problemas complexos e propondo com naturalidade novas tecnologias. Além disso, pode ajudar a responder muitas questões críticas dentro da Ciência da Computação, tal como a identificação do melhor modelo para concorrência. A idéia, neste caso, é definir uma hierarquia entre os diversos modelos através do uso de funtores (adjunções).

Pelo exposto, entende-se que a Teoria das Categorias é um modelo que permite uma abordagem original e unificada para muitos conceitos dentro da complexidade estrutural de problemas de otimização, dando abertura para um novo e promissor campo de investigação.

Com o objetivo de tornar o texto tão autocontido quanto possível, mas sem a pretensão de ser completo, a seguir são apresentadas algumas definições básicas da teoria. Para uma maior informação sobre Teoria das Categorias são indicadas as referências ([ASP 91], [BAR 90], [GOL 84], [LAW 97], [MAC 71], [MEN 2001]]).

5.1.2 Conceitos Básicos

São apresentados a seguir as definições usuais de categoria, funtor e categoria "comma", consideradas suficientes para o entendimento do que é proposto nas seções seguintes.

Definição 5.1: ***Categoria***

Uma categoria C é especificada através de uma coleção de objetos *ob***C**, conjuntos disjuntos **C**(A, B) para A, B $\in$ *ob***C**, e uma operação associativa $\circ$, tal que

1. (f $\circ$ g) é definida para g$\in$ **C**(A, B), f$\in$ **C**(D, E) sss B =D;
2. para cada A$\in$ *ob***C**, existe $1_A \in$ **C**(A, A) tal que ($1_A \circ$ f) = f e (g $\circ$ 1_A) = g, sempre que a composição estiver definida.

Membros de *ob***C** são chamados **C**-*objetos* e membros de **C**(A, B) são **C**-*morfismos* para A, B $\in$ *ob***C**.

Um dos universos mais gerais dos estudos matemáticos correntes é a categoria conhecida como **Set**, que possui conjuntos como objetos e funções como morfismos. Nessa categoria é dada uma descrição formal para os conceitos matemáticos fundamentais (número, função, relação) e são especificados axiomas estabelecendo as propriedades dos conjuntos. A categoria **Set** serve como modelo para a obtenção de outras estruturas mais gerais dentro de Teoria das Categorias.

Definição 5.2: ***Funtor***

Um funtor F : **C** $\rightarrow$ **D** para as categorias **C** e **D** associa *ob***C** com *ob***D** e transforma **C**(A, B) em **D**(FA, FB) de tal forma que são preservados:

1. unidades, isto é, $1_{FA} = F(1_A)$, para cada objeto de **C**;
2. composição, isto é, F(f $\circ$ g) = (Ff $\circ$ Fg), sempre que (f $\circ$ g) estiver definida.

Um funtor é um mapeamento de uma categoria para outra que preserva a estrutura categórica, isto é, preserva a propriedade de ser um objeto, a propriedade de ser um morfismo, os tipos, a composição e as identidades. Funtores são transformações entre categorias do tipo definido em matemática, e formam uma ferramenta categórica para tratar com objetos estruturados.

Definição 5.3: ***Categoria "comma"***

Seja **C** uma categoria e B qualquer objeto de **C**. A categoria "comma" **C**$\downarrow$B é a categoria de objetos sobre B cujos objetos são os **C**-morfismos com codomínio B, e cujos morfismos de um objeto f: A $\rightarrow$ B para outro objeto g: A′ $\rightarrow$ B é o **C**-morfismo k: A $\rightarrow$ A′, onde g$\circ$ k = f.

Segundo Goldblat [GOL 86], a categoria "comma" (*comma category*) pode ser considerada como um tipo particular de categoria das setas (categorias denotadas por $\mathbf{C}^{\rightarrow}$ cujos objetos são todos os **C**-morfismos, para uma dada categoria **C**), na qual as setas tem um domínio ou codomínio fixo.

5.2 Teoria de Universais

Apresenta-se, nesta seção, uma breve introdução para a Teoria de Universais, baseada nos trabalhos de Ellermann ([ELL 88], [ELL 95]).

Nessa teoria, a noção de universal concreto permite a interpretação lógico-filosófica sistemática das propriedades das setas universais da Teoria das Categorias. Enquanto que a Teoria Axiomática dos Conjuntos é caracterizada como uma teoria de *universais abstratos*, a Teoria das Categorias caracteriza-se por ser uma teoria de *universais concretos*, providenciando os conceitos para isolar a instância universal dentre todas as instâncias de uma propriedade.

A seguir são apresentados a origem e os conceitos básicos da teoria, introduzindo alguns exemplos ilustrativos.

5.2.1 Origem

A Teoria de Universais está interessada, originalmente, em explicar muitas das antigas idéias filosóficas sobre *universais*, tais como:

1. a noção platônica que todas as instâncias de uma propriedade tem a propriedade em virtude de participar no universal;
2. a noção do universal como mostrando a essência de uma propriedade sem qualquer imperfeição.

A noção de universalidade é fundamental para a Teoria das Categorias. O papel de fundamentação teórica de categorias é caracterizar o que é importante em matemática, exibindo suas propriedades de universalidade concretas, e não produzindo algumas construções alternativas das mesmas entidades.

O universal concreto para uma propriedade representa a característica essencial da propriedade, sem qualquer imperfeição, sendo o papel da Teoria das Categorias providenciar os conceitos de modo a salientar a instância considerada universal dentre todas as instâncias apresentando uma certa propriedade.

Todos os objetos em Teoria das Categorias com propriedades de setas universais tais como *limites* e *colimites* são universais concretos para propriedades universais. Portanto, os objetos universais de Teoria das Categorias podem, tipicamente, ser apresentados como limites (ou colimites) de um processo de filtragem para chegar à essência da propriedade.

5.2.2 Conceitos Básicos

Definição 5.5: *Teoria de Universais*

Uma teoria matemática é dita ser uma teoria de universais se ela contém uma relação binária μ e uma relação de equivalência $\approx$, tal que juntamente com certas propriedades F existem entidades associadas u_F satisfazendo as seguintes condições:

1. Universalidade: para qualquer x, $(x \mu u_F)$ sss F(x), e
2. Unicidade: se u_F e $u_{F'}$ são universais para o mesmo F, então $u_F \approx u_{F'}$.

Um universal u_F é dito *abstrato* se não for auto-participante, isto é, $\neg(u_F \mu u_F)$. Alternativamente, um universal u_F é *concreto* se ele for auto-participante, isto é, $(u_F \mu u_F)$.

Teoria dos Conjuntos: ***Teoria de Universais Abstratos***

A teoria dos conjuntos é qualificada, de imediato, como uma teoria de universais abstratos. De fato, o universal u_F representando uma propriedade F pode ser visto como o conjunto de todos os elementos com a propriedade:

$$u_F = \{ x \ / \ F(x) \}$$

A relação de participação μ é a relação de pertinência usualmente representada por $\in$.

A condição de *universalidade* em teoria dos conjuntos é o chamado *axioma da compreensão*:

"Existe um conjunto y tal que, para todo x, $x \in y$ se e somente se F(x)."

A teoria dos conjuntos também possui um *axioma de extensão*, estabelecendo que dois conjuntos com os mesmos elementos são idênticos:

"Para todo x, ($x \in y$ se e somente se $x \in y'$) implica $(y = y')$."

A característica "ingênua" do axioma da compreensão conduz à inconsistência de algumas propriedades, levando à contradições tais como o Paradoxo de Russel. Desta maneira, a teoria dos conjuntos não pode ser qualificada como uma teoria geral de universais.

Teoria das Categorias: ***Teoria de Universais Concretos***

Para definir os universais concretos da teoria das categorias, a relação de participação μ é dada pela relação de *fatoração única:*

"$x \mu y$" significa que "existe uma única seta $x \rightarrow y$"

Em relação à condição de universalidade,

"para qualquer x, $(x \mu y)$ se e somente se F(x)",

tem-se que, a existência da seta identidade para u garante a condição de auto participação de universal concreto, que corresponde com F(u) à aplicação da propriedade para u.

Em teoria de categorias, a relação de equivalência usada na condição de unicidade é naturalmente dada pela relação de *isomorfismo* na categoria.

Fazendo uma comparação entre as noções de *concreto* e *abstrato* que caracterizam as duas teorias, a teoria das categorias vista como uma teoria de universais concretos tem pelo menos um enfoque diverso da teoria dos conjuntos, vista como uma teoria de universais abstratos. Dada a coleção de todos os elementos com uma certa propriedade, a teoria dos conjuntos pode *postular* a existência de uma entidade mais abstrata – o conjunto de todos os elementos como sendo universal, enquanto que a teoria das categorias deve *encontrar* seus universais dentre todas as entidades com a propriedade.

5.2.3 Exemplo

Dados dois conjuntos A e B, considere a propriedade sobre conjuntos:

$$F(X) \equiv \text{“X está contido em A e em B”.}$$

A propriedade pode ser enunciada como: “a propriedade de ser um subconjunto de A e também um subconjunto de B”.

Nesse caso, a relação de participação é de inclusão de conjuntos.

Existe um conjunto, a interseção de A e B, denotada por $A \cap B$, que tem a propriedade F e é universal, no seguinte sentido: qualquer outro conjunto tem a propriedade F se, e somente se, participa do objeto universal. Assim, ficam caracterizadas as propriedades seguintes:

Universalidade: X participa em $A \cap B$ sss F(X), isto é:

X é um subconjunto de $A \cap B$ sss X está contido em A e em B

Concreteza: $F(A \cap B)$, isto é:

$A \cap B$ é um subconjunto de A e de B.

A interseção e a união de conjuntos numa ordenação de inclusão, são exemplos de limites e colimites em teoria de categorias. Todos os limites e colimites são universais concretos para certas propriedades.

5.3 Problemas de Otimização: Definições Revisitadas

Afim de relacionar as noções envolvidas num problema de otimização e estudar suas propriedades estruturais ao nível conceitual da Teoria das Categorias, na presente seção apresenta-se a definição de um problema de otimização através de uma relação funcional.

A princípio, o propósito desta definição de caráter funcional consiste em mostrar que a categoria dos problemas de otimização, introduzida adiante, pode ser vista como um tipo de categoria de *feixes* – uma subcategoria da categoria *“comma”*, conduzindo a investigação para a teoria de *topos* [GOL 84].

5.3.1 Problema de Otimização

Apresenta-se a seguir a definição de um problema de otimização tipado como função com codomínio fixo, dado pelo conjunto dos inteiros positivos Z^+. Obtém-se, desta forma,

uma representação gráfica para um problema de otimização, que se mostra conveniente para investigar suas propriedades estruturais, numa abordagem categorial.

Definição 5.6: *Problema de Otimização*

Um problema de otimização ***p*** é uma tripla ***p*** = (I, S, Opt), com Opt: $(I, S_I) \rightarrow Z^+$, onde:

1) I é o conjunto de instâncias do problema ***p***.
2) S é o conjunto de soluções viáveis do problema ***p***.
3) Opt é a composta das seguintes funções:
   - A função medida m: $(I, S_I) \rightarrow P(\mathbf{Z}^+)$ definida anteriormente;
   - A função *Obj* que determina o objetivo do problema (maximizar ou minimizar), definida por:

$$Obj: P(\mathbf{Z}^+) \rightarrow \mathbf{Z}^+ \text{ tal que } Obj(M_x) = m^*(x)$$

onde $M_x = \{ m_x(z) / z \in S_x \} \in P(\mathbf{Z}^+)$, para cada instância $x \in I$.

A figura Fig. 5.1 abaixo, mostra uma visualização das funções definidas acima:

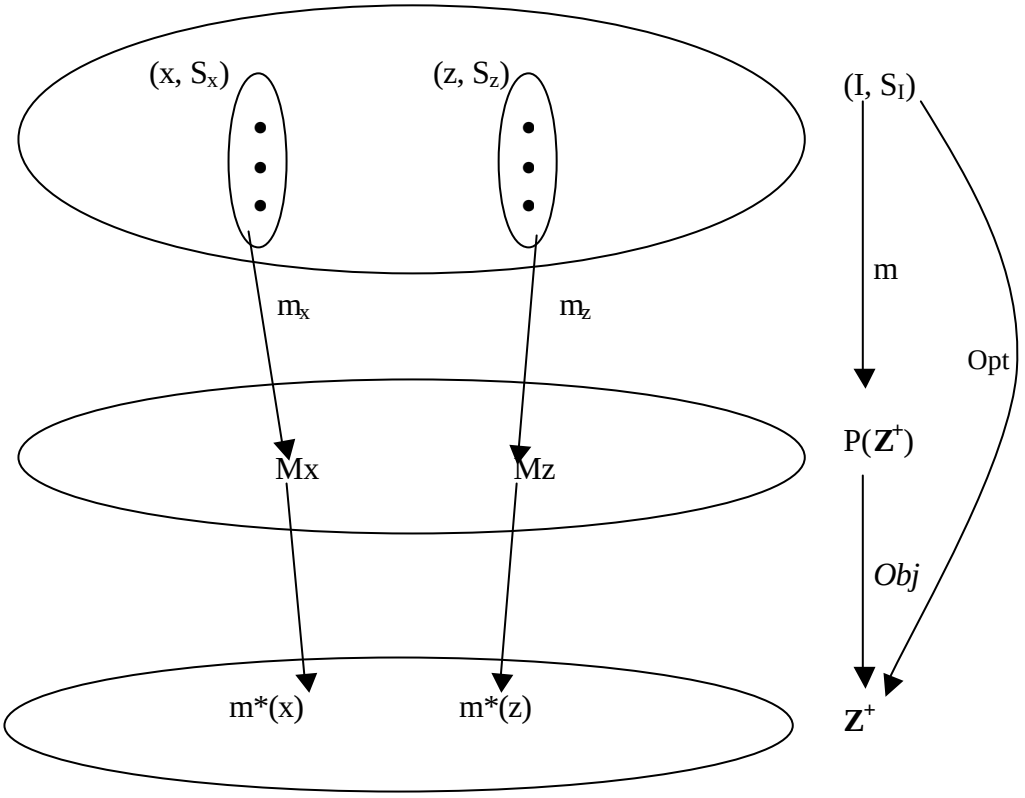

FIGURA 5.1 - Definição funcional de um problema de otimização

Observa-se que um problema de otimização ***p*** tanto pode ser representado por uma tripla ***p*** = (I, S, Opt), quanto por uma seta Opt: $(I, S_I) \rightarrow Z^+$.

Dependendo se o problema de otimização é de maximização ou minimização, a função objetivo *Obj* pode ser substituída por *Max* ou *Min*, respectivamente.

5.3.2 Redução entre Problemas de Otimização

O reconhecimento que a única estrutura que um objeto possui é devido à sua interação com outros objetos, leva ao conceito-chave na abordagem categorial para problemas de otimização: a noção de redução entre problemas.

Definição 5.7: ***Redução***

Uma redução entre os problemas de otimização ***p*** = (I, S, Opt) e ***q*** = (I', S', Opt') é um par de funções (f, g) computáveis em tempo polinomial, onde f: I $\rightarrow$ I' e g: (I', S') $\rightarrow$ (I, S) são tais que o diagrama representado na figura abaixo comuta.

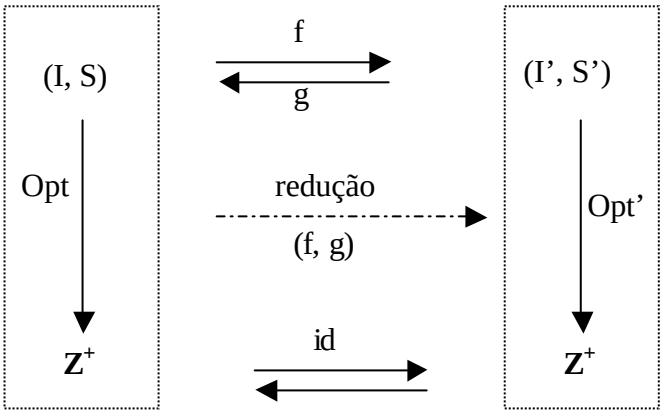

FIGURA 5.2 - Redução entre Problemas de Otimização

O significado de que o diagrama acima comuta é que, para obter a solução ótima para o problema ***p***, é possível primeiramente reduzir o problema ***p*** para o problema ***q*** e, a seguir, resolver o problema ***q***. A solução obtida será a mesma solução dada por algum procedimento que resolva o problema ***p*** diretamente.

Uma redução é definida de tal maneira que satisfaz as propriedades de *transitividade* e *reflexividade*. Dois problemas são ditos *polinomialmente equivalentes* sempre que cada um deles se reduz para o outro. Segue que uma redução define uma relação de equivalência, e portanto impõe uma ordenação parcial sobre as classes de problemas resultantes.

Como foi visto anteriormente, quando se trata de problemas de otimização NP-difíceis, soluções aproximadas com qualidade garantida podem ser obtidas mediante a definição adequada de reduções-preservando-aproximação.

5.3.3 Redução-preservando-aproximação

Uma redução-preservando-aproximação é definida como uma redução entre problemas de otimização com a adição de algumas condições garantindo alguma propriedade relacionada com aproximação.

Definição 5.8: ***Redução-preservando-aproximação***

Uma redução-preservando-aproximação entre os problemas de otimização ***p***=(I, S, Opt) e ***q***=(I', S', Opt') é uma tripla de funções (f, g, c) computáveis em tempo polinomial, onde f: I $\rightarrow$ I', g: (I', S') $\rightarrow$ (I, S) e c: $\mathbf{Z}^+ \rightarrow \mathbf{Z}^+$ são tais que o diagrama representado na Figura 5.2 comuta.

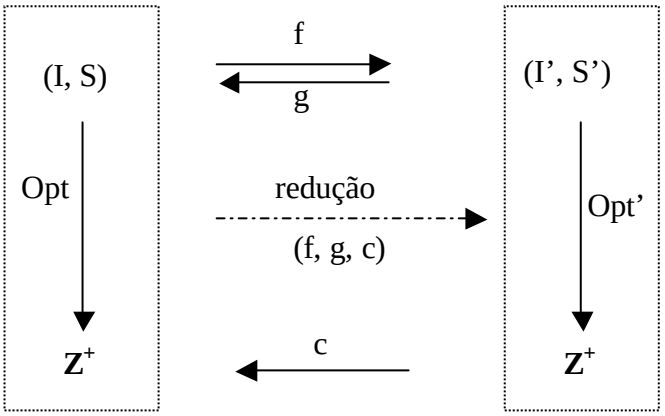

FIGURA 5.3 - Redução-preservando-aproximação

Aqui é introduzida uma função c, cujo papel é aquele de preservar a qualidade de aproximação. Dependendo da relação imposta sobre a qualidade de aproximação dos problemas, muitos tipos diferentes de redução que preservam aproximação tem sido definidos nas últimas duas décadas [TRE 97].

5.4 Categorias dos Problemas de Otimização

Com o objetivo de estruturar as classes de problemas de otimização, explora-se a noção de redutibilidade entre problemas no sentido conceptual de morfismo dentro de Teoria das Categorias. Sob este enfoque, define-se nesta seção as categorias dos problemas de otimização da classe PO e da classe NPO.

Tais categorias desempenham papel fundamental na definição de um sistema de aproximação para um problema de otimização, conforme será visto no próximo capítulo.

5.4.1 A categoria OPTS

Definição 5.9: *Categoria OPTS*

A categoria dos problemas de otimização solucionáveis em tempo polinomial, denotada por OPTS é definida como a categoria cujos objetos são os problemas de otimização pertencentes à classe PO e cujos morfismos são as reduções entre esses problemas.

A comprovação de que a categoria OPTS satisfaz as condições de uma *categoria*, vem da observação que reduções entre problemas de otimização são funções computáveis em máquinas de Turing com oráculo, com operações de composição perfeitamente estabelecidas, além de serem, comprovadamente, reflexivas e transitivas.

Teorema 5.4.1:

OPTS é uma categoria.

Prova:

Considerando reduções definidas como funções computáveis, que satisfazem as propriedades reflexiva e transitiva, tem-se que a existência de morfismos identidade é garantida através da reflexividade. A operação de composição de morfismos é dada pela composição de funções, sendo que a associatividade da composição é obtida através da propriedade transitiva das reduções.

♦

Depois de ter dado um primeiro passo na direção de uma abordagem categórica, com a definição da categoria dos problemas de otimização solucionáveis em tempo polinomial, é natural prosseguir nessa direção, visando estender para os problemas de otimização da classe NPO, especialmente aqueles considerados intratáveis.

A seguir, considerando a noção de redução-preservando-aproximação como morfismos entre problemas de otimização, é possível definir uma categoria mais ampla.

5.4.2 A Categoria OPT

Definição 5.10: *Categoria OPT*

A categoria dos problemas de otimização, denotada por OPTS é definida como a categoria cujos objetos são os problemas de otimização pertencentes à classe NPO e cujos morfismos são as reduções-preservando-aproximação entre esses problemas.

Analogamente à categoria OPTS, é facilmente verificado que OPT é realmente uma categoria.

Teorema 5.4.2:

OPT é uma categoria.

Prova:
Desde que reduções-preservando-aproximação são tambem definidas como funções computáveis satisfazendo as propriedades reflexiva e transitiva, tem-se que morfismos identidade e composição com a propriedade associativa estão garantidas. ♦

O resultado apresentado a seguir formaliza a noção do problema mais difícil para uma redução dentro da teoria de universais. No contexto de problemas de otimização, problemas NP-difíceis são provados serem objetos universais concretos para a categoria OPT.

Teorema 5.4.3:

Problemas NP-difíceis são universais concretos para a categoria OPT.

Prova:
Dada uma redução α, seja ***U*** um problema NP-difícil em relação a α. Deve-se mostrar que ***U*** é um objeto universal concreto para alguma relação de participação μ e uma relação de equivalência $\approx$, de acordo com a Definição 5.5.

Seja a seguinte relação de participação:

Para qualquer problema de otimização ***p***,

$$(\boldsymbol{p}\ \mu\ \boldsymbol{U})\ \text{ sss }\ (\boldsymbol{p}\ \alpha\ \boldsymbol{U}),$$

onde F(***p***) ≡ "***p*** se reduz através de α para ***U***".

A relação de equivalência $\approx$ é definida como a relação de equivalência polinomial em relação à redução α.

Tem-se que ***U*** satisfaz a condição de universalidade, através da definição de problema NP-difícil, isto é, para qualquer problema ***p*** pertencente à classe NPO,

$$(\boldsymbol{p}\ \mu\ \boldsymbol{U})\ \text{ sss }\ (\boldsymbol{p}\ \alpha\ \boldsymbol{U})$$

Além disso, já que a redução induz uma relação de equivalência, tem-se que, se ***U'*** é também um problema NP-difícil, então ***U*** α ***U'*** e ***U'*** α ***U.*** Isto significa que ***U*** e ***U'*** são polinomialmente equivalentes.

Finalmente, comprova-se que ***U*** é realmente um objeto *universal concreto* para a categoria OPT, sendo a condição de ser concreto correspondente à propriedade reflexiva da redução. ♦

A próxima seção apresenta aspectos de uma investigação sobre a possível caracterização das categorias dos problemas de otimização, no sentido da teoria de *topos*.

5.5 Sob o ponto de vista da teoria de *topos*

Esta seção apresenta algumas considerações sobre a investigação da caracterização da categoria dos problemas de otimização como um tipo especial de categoria: um *topos*.

A idéia intuitiva que conduziu esta investigação deve-se à representação funcional de um problema de otimização como um objeto do tipo definido pela categoria *comma*: uma função com codomínio fixo.

A seguir é apresentada uma breve introdução à teoria de *topos*, focalizando sua origem e conceitos básicos.

5.5.1 Teoria de *topos*: Origem

Segundo Barr e Wells [BAR 85], um *topos* é, sob um certo ponto de vista, uma categoria com determinadas propriedades características da categoria **Set**. No entanto, os autores observam que um *topos* não é meramente uma teoria de conjuntos generalizada, embora muitas de suas construções possam estar relacionadas com as construções da categoria dos conjuntos. Por outro lado, um *topos* é considerado como uma abstração da categoria dos feixes sobre um espaço topológico.

A justificativa para a dupla interpretação de um *topos* dada acima está na própria origem da teoria. Conforme Goldblat [GOL 84], a teoria de topos tem sua origem em duas linhas distintas do desenvolvimento matemático: Geometria Algébrica e Teoria das Categorias. A primeira iniciou com Grothendieck e sua escola de Geometria Algébrica na busca de um tratamento axiomático para a teoria dos feixes (*sheaf theory*). O ponto de partida da segunda linha é atribuído a Lawvere, com a publicação de um artigo sobre a teoria elementar da categoria dos conjuntos, no início da década de sessenta, introduzindo o conceito de *topos elementar*, a partir do estudo de categorias com um tipo especial de morfismo, chamado “sub-objeto classificador”.

5.5.2 Conceitos Básicos

Um *topos* pode ser visto como uma categoria com algumas propriedades típicas da categoria dos conjuntos. Por exemplo, em **Set** existe uma bijeção entre os subconjuntos de um conjunto D e a coleção de todas as funções com domínio em D e contradomínio no conjunto $2 = \{0, 1\}$.

De fato, dado qualquer subconjunto $A \subseteq D$, define-se a função característica $\chi_A: P(D) \rightarrow 2^D$.

Reciprocamente, dada qualquer função $f \in 2^D$, seja $f = \chi_{A(f)}$, onde $A(f) = \{x \mid x \in D \text{ e } f(x)=1\}$.

O diagrama representado na figura abaixo apresenta a situação descrita acima, onde $1 = \{0\}$ e a função *true* é a função de inclusão de 1 em 2, tal que *true*(0)=1.

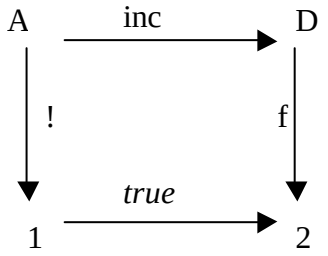

FIGURA 5.4 – Estrutura de *topos*

Nesse caso, o objeto (conjunto) 2 = {0, 1}, juntamente com a função *true*, é interpretado como um sub-objeto classificador, pois é capaz de classificar a pertinência dos elementos do conjunto A.

De modo análogo, é possível definir sub-objeto classificador para uma categoria qualquer que tenha objeto terminal.

Definição 5.11: *Sub-objeto Classificador*

Se C é uma categoria com objeto terminal 1, então um sub-objeto classificador para C é um objeto Ω junto com uma seta *true*: 1→Ω que satisfaz o seguinte axioma:

Ω-axioma: Para cada monomorfismo f: a?d existe uma única seta χ_f: d→Ω tal que o diagrama na figura abaixo representa um produto fibrado [MEN 2001]]:

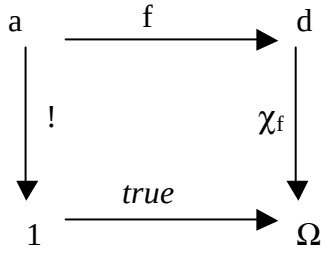

FIGURA 5.5 – Sub-objeto classificador

A seguir é introduzido o conceito de *topos elementar*, seguindo Goldblat [GOL 84].

Definição 5.12: *Topos Elementar*

Um *topos elementar* é uma categoria cartesiana fechada com sub-objeto classificador.

Exemplo: A categoria "comma" Set↓I, cujos objetos são funções com codomínio fixo I, satisfaz as condições de *topos*. Esta categoria é conhecida por ser um tipo especial de categoria de feixes (*bundles category*), definida sobre conjuntos que não são espaços vetoriais.

O exemplo acima motivou a investigação sobre as propriedades da categoria dos problemas de otimização, no sentido de representar um *topos*. A idéia inicial era a obtenção de um sub-objeto classificador que caracterizasse a exatidão da solução encontrada, em termos de sua medida.

Observou-se que problemas de decisão definidos como linguagens (conjuntos de cadeias de símbolos) tornam-se objetos da categoria (*topos*) **Se**t, cujo sub-objeto classificador é o conjunto 2 = {0, 1}. No entanto, a existência de problemas indecidíveis, mostrou que a subcategoria dos problemas de decisão (com reduções como morfismos) não caracteriza um *topos*.

Já os problemas de otimização tendo uma estrutura mais complexa, seriam representados como objetos de uma categoria mais rica – uma subcategoria da categoria "comma" Set$\downarrow \mathbf{Z}^{+}$.

A dificuldade na caracterização dos elementos desta categoria conduziram as investigações desenvolvidas neste trabalho para um direcionamento diferente.

5.6 Comentários Finais

Neste capítulo foram introduzidas as categorias OPTS e OPT dos problemas de otimização pertencentes às classes PO e NPO, respectivamente, seguindo a idéia intuitiva de uma abordagem categorial para a complexidade estrutural de problemas de otimização.

Os modelos que fornecem fundamentação matemática para esta abordagem foram apresentados de modo informal. Em especial, a introdução de uma teoria geral de universais permitiu a formalização de problemas NP-difíceis como objetos universais dentro da categoria OPT.

Dentre as diversas tentativas para o encaminhamento das investigações sobre as categorias definidas neste capítulo, são apresentadas algumas considerações sobre a caracterização da categoria dos problemas de otimização como um tipo especial de categoria: um *topos*.

O estudo das relações entre as duas novas categorias definidas nesta seção e a introdução de um modelo para a hierarquia de aproximação para um problema de otimização, são os objetos de interesse do próximo capítulo.

6 Categoria das Aproximações

O presente capítulo tem por objetivo o estudo formal das questões relativas à aproximabilidade de problemas de otimização NP-difíceis, sob o ponto de vista de uma abordagem categorial.

A idéia central deste capítulo está baseada na Teoria Categorial da Forma (*Categorical Shape Theory*), desenvolvida por Cordier e Porter [COR 90], e tem sido motivada por trabalhos prévios de Rattray ([RAT 94], [RAT 96]], [RAT 98]), no contexto de sistemas complexos. O tema da investigação que vem sendo desenvolvida por Rattray, estabelece que, num sistema complexo, as aproximações são os únicos objetos que codificam a informação que o sistema pode analisar. A identificação e o reconhecimento de um objeto de interesse, visto como uma informação desejada, requer a identificação de um modelo (ou arquétipo) que melhor aproxime esse objeto.
A definição de categorias de aproximação para cada objeto de interesse modela a estrutura hierárquica de estados do sistema.

A seção seguinte apresenta a motivação para a investigação das relações entre as categorias OPTS e OPT introduzidas no capítulo anterior, em termos da Teoria Categorial da Forma. Uma breve introdução a essa teoria é apresentada a seguir, visando o entendimento das conexões com as categorias OPTS e OPT.

Finalmente, são definidas as categorias de aproximação para cada problema de otimização da categoria OPT, a partir da definição de um funtor de comparação entre as categorias OPTS e OPT.

Uma primeira versão dessa idéia foi apresentada no V International Conference on Computing Anticipatory Systems – CASYS'01, tendo sido aceito para publicação o respectivo artigo completo no International Journal of Computing Anticipatory Systems, a ser publicado em maio de 2002 (ref. [LEA 2002]]).

6.1 Categoria OPTS x Categoria OPT

Tendo definido a categoria dos problemas de otimização solucionáveis em tempo polinomial - OPTS e a categoria dos problemas de otimização – OPT, a próxima etapa é identificar as relações entre elas. Iniciando a partir das seguintes questões básicas:

1. Como as categorias OPTS e OPT interagem uma com a outra?
2. O que significa dizer que um problema A "aproxima" um problema de otimização B?
3. O que se entende por "melhor aproximação" para um tal problema de otimização?

Observa-se que as questões acima são fundamentais para o entendimento do processo de aproximação para problemas de otimização.

De fato, considere B um problema de otimização da classe NPO, reconhecidamente NP-difícil. Nesta situação, uma alternativa é encontrar um algoritmo aproximativo que apresente uma solução com qualidade garantida para tal problema. Entretanto, sabe-se que problemas que possuem solução pertencem à classe PO. Portanto, a solução aproximada encontrada é a solução exata de um certo problema A na classe PO, relacionado de alguma forma ao problema B. Isto significa que estabeleceu-se uma comparação entre elementos das duas classes, de modo a encontrar a solução aproximada para o problema de interesse.

A situação discutida acima é ilustrada na figura abaixo.

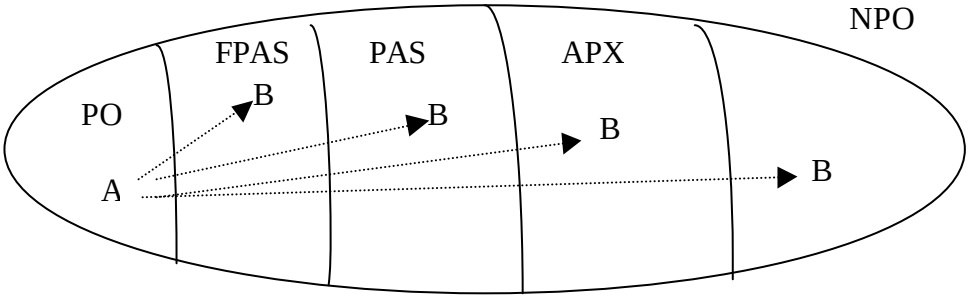

FIGURA 6.1 - Idéia intuitiva de aproximação para problema NP-difícil

Na busca pela formalização desta idéia intuitiva, o objetivo agora é providenciar mecanismos para a comparação entre tais categorias. Isto conduzirá para a Teoria Categorial da Forma, objeto de interesse da próxima seção.

6.2 Teoria Categorial da Forma (*Categorical Shape Theory*)

Frequentemente deseja-se encontrar um modelo matemático de uma estrutura, de modo a explicar suas propriedades e prever seu comportamento em diferentes circunstâncias. Relacionado à questão da aproximabilidade de problemas de otimização, é provável que a teoria categorial da forma seja tal modelo.

Esta seção apresenta uma visão geral dessa teoria, introduzindo a idéia básica originalmente desenvolvida por seus autores Cordier e Porter [COR 90].

6.2.1 Origem

A idéia básica da Teoria Categorial da Forma é que, em qualquer situação de aproximação, as aproximações são os objetos que codificam a única informação que se pode analisar. Basicamente, consiste em definir um mecanismo de comparação entre certas categorias, a partir do qual é possível construir um sistema de aproximações para objetos de interesse, estabelecendo uma hierarquia sobre as aproximações.

No contexto da teoria categorial da forma são identificados os seguintes elementos:

1. uma categoria **B** de objetos de interesse;
2. uma categoria **A** de arquétipos ou objetos-modelo;
3. uma "comparação" de objetos com arquétipos, isto é, um funtor K: **A** $\rightarrow$ **B**.

Traduzindo em poucas palavras, a idéia por trás da teoria categorial da forma é que, reconhecer e entender um objeto de interesse B via uma comparação K: **A** $\rightarrow$ **B**, requer a identificação do correspondente arquétipo A que melhor represente B.

A definição arbitrária desses elementos permite, por exemplo, explorar as propriedades de sistemas de reconhecimento, os quais devem ser independentes de qualquer característica especial que o mecanismo de comparação K possa ter. Já ao impor condições estruturais sobre K, **A** ou **B**, pode-se tentar revelar mais sobre as estruturas em situações diversas.

6.2.2 Conceitos Básicos

Definição 6.1: *Aproximação*

Dadas uma categoria **A** de arquétipos, uma categoria **B** de objetos de interesse e uma comparação K: **A** $\rightarrow$ **B** , uma aproximação para um objeto de interesse B$\in$**B** é o par (f, A), onde A$\in$**A** é um arquétipo e f: B $\rightarrow$ KA.

Um morfismo entre aproximações h: (f, A) $\rightarrow$ (g, A') é um morfismo h: A $\rightarrow$ A' dos arquétipos subjacentes, tal que K(h) $\circ$ f = g, isto é, o triângulo abaixo comuta.

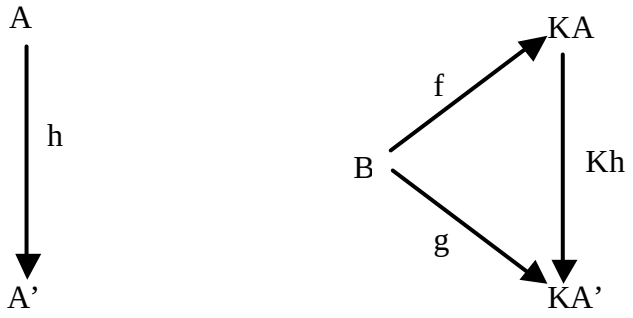

FIGURA 6.2 - Morfismo entre Aproximações

Aproximações com seus morfismos formam a categoria B$\downarrow$K, a categoria "comma" de K-objetos sob B.

A forma de cone do diagrama que representa os morfismos na categoria **B** definindo as aproximações para algum objeto de interesse B, sugere que tomar o objeto limite do diagrama resultaria num arquétipo A* satisfazendo a condição de estar "tão próximo quanto possível" do objeto B. A situação descrita é ilustrada na seguinte figura:

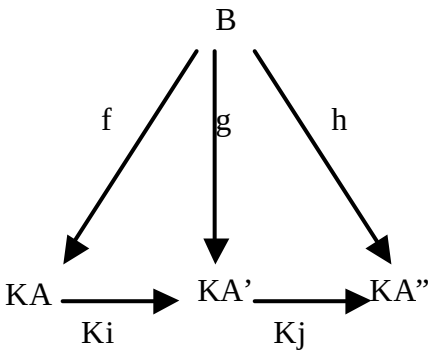

FIGURA 6.3 - Aproximações para objeto de interesse B

A noção de que um objeto é a " melhor aproximação" para outro é dada formalmente através de um objeto universal na categoria.

Definição 6.2: ***Objeto K-universal***

Seja K: **A** $\rightarrow$ **B** um funtor comparação. Um arquétipo A$\in$**A** é dito ser K-universal para um objeto de interesse B$\in$**B** , se existe uma aproximação (f, A) para B tal que, para cada aproximação (g, A') para B, com A'$\in$**A**, existe um único morfismo h: A $\rightarrow$ A' em **A**, com g = K(h) $\circ$ f.

Definição 6.3: ***Categoria K-universal***

Uma categoria de arquétipos **A** é dita ser K-universal na categoria **B** se cada objeto de interesse de **B** tem um arquétipo K-universal em **A**.

6.3 Conexões com as Categorias OPTS e OPT

No cenário da Teoria Categorial da Forma, considera-se a categoria OPT como a categoria de objetos de interesse B, a categoria OPTS como a categoria de arquétipos A, e o funtor K: OPTS $\rightarrow$ OPT um mecanismo de comparação relacionado com um método de aproximação (por exemplo, a relaxação). A idéia, neste caso, é manter a definição de K arbitrária, de modo a se alcançar a abstração que torna a teoria tão geral quanto possível.

Assim, voltando às questões propostas na primeira seção do presente capítulo, procede-se nesta seção a uma interpretação para o relacionamento entre as categorias OPTS e OPT, além de apresentar uma semântica para a aproximação de um problema de otimização.

A Figura 6.3 abaixo ilustra o relacionamento que se deseja definir entre estas categorias.

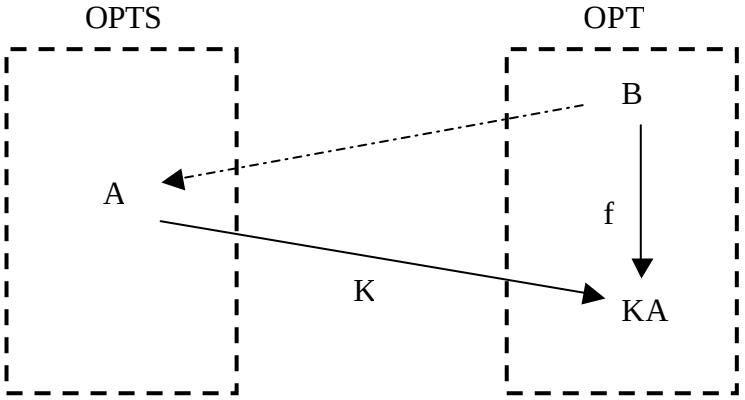

FIGURA 6.4 - Aproximação para um Problema de Otimização

Definição 6.4: *Problema Aproximável*

Dada uma comparação K: OPTS $\rightarrow$ OPT, um problema $B \in$ OPT é dito *aproximável*, se existir um problema $A \in$ OPTS e uma redução-preservando-aproximação f tal que f: B $\rightarrow$KA.

Nesse caso, diz-se que o par (f, A) é uma *aproximação* para o problema B.

Verifica-se que, através desta teoria, é possível identificar a melhor aproximação para um problema de otimização B, se existir.

De fato, a existência de problemas de otimização não permitindo qualquer tipo de aproximação torna a proposição abaixo consistente.

Proposição 6.1:

A categoria OPTS é K-universal em OPT se e somente se PO = NPO.

6.4 Categoria das Aproximações

Nesta seção apresenta-se uma interpretação semântica para a resposta à terceira questão formulada na primeira seção deste capítulo, a saber:

3. O que se entende por “melhor aproximação” para um problema de otimização?

Como foi visto anteriormente, a Teoria Categorial da Forma permite caracterizar os graus de aproximação, através da construção de um sistema de aproximação para cada objeto de interesse, usando a noção de colimite para o diagrama em forma de cone das aproximações. Antes disso, porém, é necessário definir a categoria das aproximações para um problema de otimização.

Sejam (f, A) e (g, A´) aproximações para um problema de otimização B. Um morfismo entre aproximações h: (f, A) $\rightarrow$ (g, A') é um morfismo (redução) h: A $\rightarrow$ A' entre os problemas A e A' na categoria OPTS, tal que K(h) $\circ$ f = g.

Definição 6.5: ***Categoria das Aproximações***

Dado um funtor comparação K, a categoria $APX_{B,K}$ das aproximações para um problema de otimização B é definida pela categoria B$\downarrow$K, a categoria "comma" de K-objetos sob B.

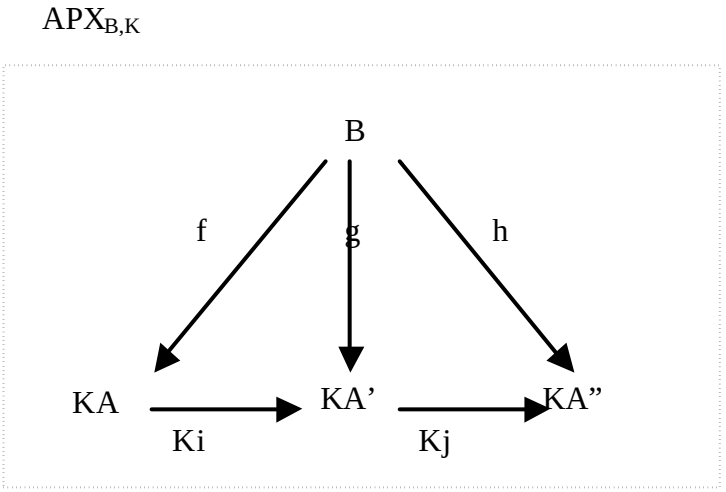

FIGURA 6.5 - Categoria das Aproximações

Respondendo à questão anteriormente proposta, formaliza-se a idéia de objetos satisfazendo propriedades universais, no contexto da aproximação.

Associado com a categoria das aproximações tem-se um funtor projeção pr_B que recupera o elemento arquétipo da aproximação. Este funtor, chamado funtor codomínio, é definido da seguinte maneira:

$$pr_B : APX_{B,K} \rightarrow OPTS$$

onde

pr_B(f, A) = A, e

pr_B(h: (f, A) $\rightarrow$(g, A')) = h: A $\rightarrow$ A'

A existência do limite do diagrama pr_B: $APX_{B,K} \rightarrow$ OPTS resultaria num arquétipo A* em OPTS que melhor aproxima o problema B, no sentido de ser um objeto K-universal, conforme definido anteriormente.

Outras importantes questões surgem. Por exemplo:

- Como comparar dois problemas de otimização em relação ao grau de aproximação?
- O que significa dizer que dois problemas de otimização possuem uma melhor aproximação comum?

A comparação de dois objetos de interesse B e B', usando somente as informações obtidas através do funtor comparação K: OPTS $\rightarrow$ OPT requer a comparação das correspondentes categorias de aproximação $APX_{B,K}$ e $APX_{B',K}$.

Se h: B $\rightarrow$ B' é um morfismo (redução) na categoria OPT, qualquer aproximação (g: B'$\rightarrow$ KA, A) dá uma aproximação (gh: B $\rightarrow$ KA, A) através da composição, isto é, o diagrama apresentado na figura abaixo comuta.

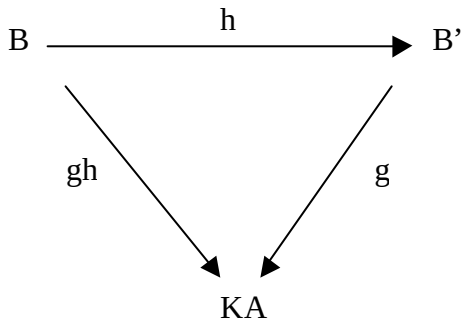

FIGURA 6.6 - Composição de Aproximações

O morfismo h: B $\rightarrow$ B' induz um funtor

$$h^*: APX_{B',K} \rightarrow APX_{B,K}$$

tal que

$$h^*(g: B' \rightarrow KA, A) = (gh: B \rightarrow KA, A)$$

Assim, constata-se que o funtor h* satistaz a condição de tornar o diagrama abaixo comutativo, preservando o esquema de aproximação.

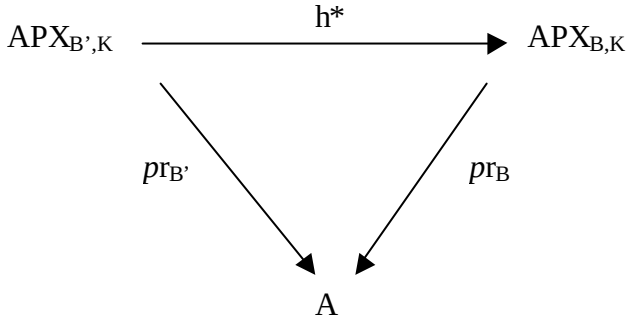

FIGURA 6.7 - Preservação das Aproximações

Por outro lado, para se dizer que dois problemas de otimização possuem uma melhor aproximação comum, é necessário entender o significado da ação do funtor comparacão K. Para isso precisa-se definir a categoria da forma (*shape category*).

Definição 6.6: Categoria da Forma (*Shape Category*)

A categoria da forma Sh(K) para o funtor comparação K: $\mathbf{A} \rightarrow \mathbf{B}$, sendo **A** e **B** as categorias de arquétipos e objetos de interesse, respectivamente, tem como objetos os objetos de **B** e para morfismos h: B $\rightarrow$ B' os funtores induzidos h*: $APX_{B',K} \rightarrow APX_{B,K}$, satisfazendo $pr_B \circ h^* = pr_{B'}$.

Diz-se que dois objetos de interesse B e B' na categoria **B** tem a mesma K-forma se eles são isomorfos na categoria Sh(K).

Objetos tendo arquétipos equivalentes, identificados com aqueles arquétipos K-universal isomorfos, terão a mesma K-forma, já que K é insuficientemente refinado para distinguir entre eles. Portanto, ter a mesma K-forma é uma generalização de ter arquétipos K-universais isomorfos.

6.5 Comentários Finais

Neste capítulo foram apresentados importantes resultados na direção de obter uma fundamentação teórica para o tema de principal interesse deste trabalho: a questão da aproximabilidade de problemas de otimização.

Baseado na Teoria Categorial da Forma, apresentou-se uma interpretação para o relacionamento entre as categorias OPTS e OPT, respondendo efetivamente às questões referentes ao significado de que um problema aproxima outro, além da definição da melhor aproximação para um problema, quando existe.

A introdução de uma categoria de aproximações, definida para cada problema de otimização, permitiu responder à questão sobre o que significa uma "melhor aproximação" para um problema de otimização

Muitas outras questões poderiam ser formuladas, com base na Teoria Categorial da Forma, justificando uma possível continuação da pesquisa aqui iniciada.

7 Hierarquia de Aproximação para Problemas de Otimização

Conforme mencionado na introdução deste trabalho, a identificação das classes P e NP e a introdução da teoria da NP-completude no início da década de setenta, foi determinante para o desenvolvimento da Teoria da Complexidade Computacional.

A classificação de problemas em termos da dificuldade computacional para obter as respectivas soluções, manteve-se, quase que exclusivamente, restrita aos problemas pertencentes à classe NP, e principalmente aos problemas de decisão.

Conforme Papadimitriou [PAP 94], embora seja reconhecido que a maioria dos problemas combinatoriais ocorrendo na prática pertencem à classe NP, existe uma variedade de problemas interessantes que parecem não estar incluídos em NP, justificando desta maneira a definição de classes de complexidade adicionais.

Neste capítulo apresenta-se uma visão mais geral para a classificação de problemas, considerando também aqueles problemas que não parecem estar incluídos na classe NP.

O objetivo deste estudo consiste em investigar as propriedades estruturais das classes de problemas de otimização, no contexto de uma classificação mais geral de problemas.

A partir daí, pretende-se obter uma fundamentação da hierarquia de complexidade das classes de aproximação a partir da fundamentação teórica dada pelos modelos semânticos estudados. O ponto de partida será a extensão da chamada *hierarquia polinomial*, construída basicamente sobre problemas de decisão, para problemas de otimização.

7.1 Hierarquias de Complexidade

Nesta seção são apresentadas as formas de classificação para problemas que trascendem o mundo de NP. Tais problemas apresentam grau de dificuldade não menor que qualquer problema em NP, ou seja, são considerados tão difíceis computacionalmente, do que qualquer problema NP-completo.

Dependendo das características dos problemas, diversas hierarquias de complexidade são definidas na literatura.

A seguir são apresentadas as principais hierarquias, conforme as referências ([BOV 94], [GAR 79], [PAP 94]).

7.1.1 Hierarquia Booleana

A Hierarquia Booleana é uma coleção de classes de linguagens definidas por:

$$BH = \cup \{BH_i \mid i \geq 1\}$$

onde

$BH_1 = NP$;

$BH_2 = \{L_A \cap L_B \mid L_A, L_B \in NP\}$;

$BH_{2i+1} = \{L_A \cup L_B \mid L_A \in BH_{2i} \wedge L_B \in NP\}$, com $i \geq 1$;

$BH_{2i+2} = \{L_A \cap L^c_B \mid L_A \in BH_{2i+1} \wedge L_B \in NP\}$, com $i \geq 1$;

A hierarquia booleana é a menor classe que inclui NP e que é fechada com respeito a união, interseção e complemento de linguagens - por isso chamada FECHO BOOLEANO DE NP.

7.1.2 Classes Relativizadas

Dado qualquer conjunto A, define-se a classe

$$P^A = \{ B \mid B \propto_T A \}$$

consistindo de todos os conjuntos que são deterministicamente Turing-redutíveis em tempo polinomial ao conjunto A.

P^A é chamada uma "relativização de P para o oráculo A".

Isto significa que a cada conjunto A está associada uma classe de conjuntos: aqueles conjuntos que podem ser resolvidos em tempo polinomial, dado um oráculo A.

De modo análogo, define-se a classe

$$NP^A = \{ B \mid B \propto_T^{NP} A \}$$

consistindo de todos os conjuntos que são não-deterministicamente Turing-redutíveis em tempo polinomial ao conjunto A.

7.1.3 Hierarquia Polinomial

O processo de definir novas classes em termos de classes já conhecidas pode ser estendido indefinidamente, levando para classes de dificuldade aparentemente cada vez maiores.

Este foi o processo concebido por Meyer e Stockmeyer(1972) para definir a chamada HIERARQUIA POLINOMIAL, baseada na hierarquia aritmética de Kleene.

A construção da hierarquia polinomial é feita pela relativização das classes P e NP pelas classes de níveis inferiores.

Hierarquia Polinomial: *nível zero*

O nível zero da hierarquia é definido por:

$$\Sigma_0 = \Pi_0 = \Delta_0 = P$$

Hierarquia Polinomial: *nível 1*

$$\Delta_1 = P^{\Sigma_0} = P^P = P$$

$$\Sigma_1 = NP^{\Sigma_0} = NP^P = NP$$

$$\Pi_1 = \text{co-}\Sigma_1 = \text{co-NP}$$

Hierarquia Polinomial: *nível k*

$$\Delta_{k+1} = P^{\Sigma_k}$$

$$\Sigma_{k+1} = NP^{\Sigma_k}$$

$$\Pi_{k+1} = \text{co-}\Sigma_{k+1}$$

7.1.4 Hierarquia de Consulta (*Query hierarchy*)

Para cada função f: $Z^+ \rightarrow Z^+$, a classe $P^{NP[f(n)]}$ é definida como o conjunto de todas as linguagens que podem ser reconhecidas por uma OTM com um oráculo para SAT que faz no máximo f(n) consultas para o oráculo, onde n é o tamanho da entrada.

A hierarquia de consulta consiste das classes definidas por

$$QH_k = P^{NP[k]} \text{ , para } k \geq 0$$

A classe $QH = \cup_{k \geq 0} QH_k$

Para problemas que não são de decisão, definem-se de modo análogo as classes $FP^{NP[f(n)]}$.

7.2 Comentários Finais

O objetivo deste capítulo foi introduzir uma classificação geral para problemas, considerando os diversos modos de definição de hierarquias de complexidade existentes na literatura.

A partir desta abordagem inicial, investiga-se uma interpretação para a hierarquia de aproximação em termos da Teoria das Categorias.

No próximo capítulo são apresentadas as conclusões deste trabalho.

8 Conclusão

Este capítulo apresenta um fechamento das principais idéias desenvolvidas na tese, destacando os pontos positivos alcançados, identificando possíveis pontos negativos e propondo questões que permanecem em aberto, direcionando para trabalhos futuros.

Na primeira seção é apresentado um sumário das principais contribuições julgadas relevantes para o enriquecimento da pesquisa científica na área de Matemática da Computação.

A segunda e última seção finaliza o relato do trabalho desenvolvido, indicando tópicos que poderiam ser abordados, a médio e longo prazo, como continuação deste trabalho.

8.1 Principais Contribuições

O principal objetivo deste trabalho consistiu no desenvolvimento de uma fundamentação teórica para o estudo formal de problemas de otimização NP-difíceis, focalizando sobre as propriedades estruturais desses problemas relacionadas à questão da aproximabilidade. Neste sentido, apresentou-se uma abordagem semântica para tratar algumas questões originalmente estudadas dentro da Teoria da Complexidade Computacional, especificamente no contexto da Complexidade Estrutural. Convém acrescentar, que embora o tratamento dessas questões tenha sido matemático, manteve-se o ponto de vista da Ciência da Computação.

Uma das principais questões que nortearam o desenvolvimento desta tese, indagava sobre a caracterização num sentido qualitativo, daqueles algoritmos que, garantidamente, fornecem soluções aproximadas para problemas de otimização NP-difíceis.

A resposta para tal questão foi dada através da estruturação do conjunto dos algoritmos aproximativos para um problema de otimização genérico como uma ordem parcial, através da Teoria dos Domínios.

No contexto da Teoria dos Domínios foram obtidos os seguintes resultados:

- Definição de uma relação de ordem parcial, em termos dos requerimentos de qualidade dos algoritmos, chamada *ordem de e-aproximação*;
- Estruturação do conjunto $\mathbb{A}$ de algoritmos aproximativos para um problema de otimização genérico como uma ordem parcial, representada por $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$;
- Se o problema *p* está na classe APX, então a estrutura $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ é uma ordem parcial, mas não é completa, a menos que P=NP;
- ♦ Se o problema *p* está na classe PAS, então a estrutura $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ é uma ordem parcial completa, mas não é cpo-algébrico;

- Se *p* está na classe APX, então $\mathbb{A} = (\mathbb{A}; \sqsubseteq_\varepsilon, \perp_\varepsilon)$ é um semi-reticulado condicionalmente superior (abreviado por CUSL);
- $\overline{\mathbb{A}} = (\overline{\mathbb{A}}, \subseteq, [\perp]\}$ é um Domínio de Scott e $\overline{\mathbb{A}}_c = \{[a] / a \in \mathbb{A}\}$, onde $\overline{\mathbb{A}}$ denota o conjunto de todos os ideais contidos em $\mathbb{A}$ e $\overline{\mathbb{A}}$ são os ideais principais.

Quanto à segunda questão colocada, indagando sobre como se modelam as relações entre problemas que apresentam diferenciados graus de aproximabilidade, buscou-se na Teoria das Categorias o modelo semântico para entender a rica hierarquia entre problemas aproximáveis. O reconhecimento de que a noção de redutibilidade entre problemas constitui-se num processo de especialização da Teoria das Categorias, conduziu a investigação das relações entre as classes de aproximação para uma abordagem de natureza categorial, focalizando sobre as reduções preservando aproximação entre problemas de otimização.

No contexto da Teoria das categorias, as principais contribuições foram:

- Apresentação de novas perspectivas para conceitos já conhecidos, num processo de *especialização* – no sentido do reconhecimento de que uma estrutura particular é uma instância de um fenômeno mais geral.

- Utilização de uma linguagem universal para tratar com as questões específicas da complexidade de problemas de otimização.

- Introdução de duas novas categorias: a categoria OPTS, cujos objetos são os problemas de otimização da classe PO, tendo reduções abstratas como morfismos e, da categoria OPT, cujos objetos são os problemas de otimização da classe NPO, com morfismos definidos por reduções-preservando-aproximação;

- Formalização de problemas NP-difíceis como objetos universais concretos para a categoria OPT.

- Definição de uma sistema de aproximação para problemas de otimização, baseado em teoria categorial da forma (*categorical shape theory*).

8.2 Prosseguimento do Trabalho

Para o prosseguimento deste trabalho sugere-se:

- No contexto da Teoria dos Domínios, pretende-se retomar a definição de ordem de ε-aproximação, considerando-a como a ordem parcial definida em termos da função de

qualidade do algoritmo (ver Definição 6.3.1). Neste caso, observa-se que poderão existir algoritmos *a* e *b* tais que, para alguma instância I, *a* será melhor que *b* e, para outra instância I', *b* será melhor que *a.;*

- Ainda, sob este ponto de vista "local" dos algoritmos aproximativos para um determinado problema de otimização, resta estabelecer uma noção adequada de *função contínua*, permitindo a computabilidade em tempo polinomial, senão da solução exata, pelo menos de uma solução tão boa quanto se queira para esse problema, em relação ao requerimento de qualidade desses algoritmos;
- Além disso, num ponto de vista mais "global", um aspecto importante a ser abordado é o significado da *redutibilidade preservando aproximação* dentro da Teoria dos Domínios;
- Retomar a interpretação das categorias OPTS e OPT em termos de uma abordagem no sentido da Teoria de TOPOS,
- Explorar outras possibilidades de aplicação da Teoria Categorial da Forma, considerando tambem o requerimento de exatidão para uma solução aproximada, na definição de um sistema de aproximação para um problema de otimização;
- Aprofundar a investigação sobre o estudo da hierarquia de aproximação para problemas de otimização, usando as ferramentas da Teoria das Categorias tais como funtores, transformações naturais e adjunções

Anexos

Serão apresentados, a seguir, os anexos descritos no texto.

O Anexo 1 corresponde à descrição dos problemas de otimização apresentados como exemplos, seguindo o modelo apresentado por Ausiello et alli [AUS 99].

O Anexo 2 consiste de uma seleção dos artigos publicados referentes ao trabalho ora apresentado.

Anexo 1

Uma lista de problemas de otimização da classe NPO pode ser encontrada no compêndio desenvolvido por Crescenzi e Kann [CRE 95], atualizada eletronicamente pelo endereço

http://www.nada.kth.se/theory/problemlist.html.

Seguindo o modelo introduzido por Garey e Johnson [GAR 79], também Ausiello et alli [AUS 99] apresentam uma lista contendo informação sobre a aproximabilidade de mais de 200 problemas de otimização, classificada por categorias de acordo com o tema principal de que trata o problema.

Todos os problemas apresentados a seguir foram retirados de [AUS 99].

Teoria de Grafos

GT1: *Cobertura Mínima de Vértices*

Instância: Grafo G = (V, E)

Solução: Uma cobertura de vértices para G, isto é, um subconjunto V' $\subseteq$ V tal que, para cada aresta (u, v) $\in$ E, pelo menos um de u e v pertence a V'.

Medida: Cardinalidade da cobertura de vértices, isto é, | V' |.

Status: APX-completo

GT5: *Coloração Mínima de Grafos*

Instância: Grafo G = (V, E)

Solução: Uma coloração de vértices de G, isto é, uma partição de V em conjuntos disjuntos V_1, V_2, ..., V_k tal que cada V_i é um conjunto independente para G.

Medida: Cardinalidade da coloração, isto é, o número de conjuntos independentes V_i.

GT22: *Clique Máximo*

Instância: Grafo G = (V, E)

Solução: Um clique em G, isto é, um subconjunto V' $\subseteq$ V tal que cada dois vértices em V' são unidos por uma aresta em E.

Medida: Cardinalidade do clique, isto é, |V'|.

GT23: ***Conjunto Independente Máximo***

Instância: Grafo G = (V, E)

Solução: Um conjunto independente de vértices, isto é, um subconjunto $V' \subseteq V$ tal que cada não existem dois vértices em V' unidos por uma aresta em E.

Medida: Cardinalidade do conjunto independente, isto é, |V'|.

Projeto de Redes

ND14: ***Corte Máximo***

Instância: Grafo G = (V, E)

Solução: Uma partição de V em conjuntos disjuntos V_1 e V_2.

Medida: Cardinalidade do corte, isto é, o número de arestas com uma exttremidade em V_1 e outra extremidade em V_2.

Status: APX-completo

Problemas de Roteamento

NT32: ***Caixeiro-viajante Mínimo***

Instância: Conjunto C de m cidades, distâncias $d(c_i, c_j) \in \mathbf{N}$, para cada par de cidades $c_i, c_j \in C$

Solução: Um roteiro de C, isto é, uma permutação $\pi : [1..m] \rightarrow [1..m]$.

Medida: Comprimento do roteiro, isto é,

$$d(\{ c_{\pi(m)}, c_{\pi(1)} \}) + \sum_{i=1}^{m-1} d(\{c_{\pi(i)}, c_{\pi(i+1)}\})$$

Status: NPO-completo

Armazenamento

SR1: ***Empacotamento binário mínimo***

Instância: Conjunto finito U de ítems, um tamanho $s(u) \in \mathbf{Z}^+$ para cada $u \in U$, e uma capacidade B inteiro binário positivo.

Solução: Uma partição de U em conjuntos disjuntos U_1, U_2, ..., U_n tal que a soma dos ítems em cada U_i é menor ou igual a B.

Medida: O número de binários usados, isto é, o número m de conjuntos disjuntos.

Status: FPAS

Programação Matemática

MP13: *Problema da Mochila Máxima*

Instância: Conjunto finito U, para cada $u \in U$ um tamanho $s(u) \in \mathbf{Z}^+$ e um valor $v(u) \in \mathbf{Z}^+$, um inteiro positivo $B \in \mathbf{Z}^+$.

Solução: Um subconjunto $U' \subseteq U$ tal que $\sum_{u \in U'} s(u) \leq B$.

Medida: Peso total dos elementos escolhidos, isto é, . $\sum_{u \in U'} v(u)$

Status: FPAS

Anexo 2

São apresentados neste anexo as referências de algumas publicações realizadas durante o desenvolvimento do presente trabalho.

Convém ressaltar que as discussões geradas em torno dos temas desenvolvidos nos trabalhos apresentados em congressos, seminários e artigos para jornais, muito contribuíram para o entendimento e aprofundamento pessoal das questões fundamentais para a Ciência da Computação em geral, em particular da área de Matemática da Computação.

[LEA 97a] LEAL, L. A. S. **Domínios de Scott: Teoria e Aplicações**. Porto Alegre. PPGC/UFRGS. TI nº 623, 1997.

[LEA 97b] LEAL, L. A. S. **A Teoria de Algoritmos frente à Intratabilidade de Problemas**. Porto Alegre. PPGC/UFRGS. EQ nº 16, 1997. 79p.

[LEA 98] LEAL, L. A. S.; TOSCANI, L. V.; CLAUDIO, D. M. Approximative Algorithms through Domain Theory. In: **Optimization'98**. Coimbra-PT. Resumos, p. 44, 1998.

[LEA 00] LEAL, L. A. S.; MENEZES, P. B.; CLAUDIO, D. M.; TOSCANI, L. V. Categorias dos Problemas de Otimização. In: **WMF2000 - Workshop on Formal Methods, 3. SBES, 14**. 04-06 Outubro, João Pessoa, Paraíba. Ed. SBC, 2000. p. 104-109.

[LEA 2001]] LEAL, L. A. S.; MENEZES, P. B.; CLAUDIO, D. M.; TOSCANI, L. V. Optimization Problems Categories. In: **Formal Methods and Tools for Computer Science, EUROCAST'2001, 8**. 19-23 Fevereiro, Las Palmas de Gran Canária, Ilhas Canárias, Espanha. IUCTC. R. Moreno-Diáz e A. Quesada-Arencibia (Eds.), 2001. p. 93-96.

[LEL 2001]] LEAL, L. A. S.; MENEZES, P. B.; CLAUDIO, D. M.; TOSCANI, L. V. Optimization problems Categories. **EUROCAST'01, LNCS 2178**. R. Moreno-Diáz, B. Buchberger, J-L. Freire (Eds.). Springer-Verlag, 2001. p. 285-299.

[LEA 2002]] LEAL, L. A. S.; CLAUDIO, D. M.; MENEZES, P. B.; TOSCANI, L. V. A Modelling the Approximation Hierarchy to Optimisation Problems Through Category Theory. **International Journal of Computing Anticipatory Systems**, 2002. (Artigo completo do CASYS'01 aceito para publicação)

Artigos submetidos para publicação:

LEAL, L. A. S.; CLAUDIO, D. M.; TOSCANI, L. V.; MENEZES, P. B. A Categorical Approach to NP-Hard Optimization Problems, 2001. 15p
Submetido à **Seleta do XXIV CNMAC.**
Publicação da Sociedade Brasileira de Matemática Aplicada e Computacional.

LEAL, L. A. S.; TOSCANI, L. V.; MENEZES, P. B.; CLAUDIO, D. M.; Structural Complexity: A Categorical View, 2001. 17p
Submetido ao **Eletronic Journal on Mathematics of Computation – EJMC**
EDUCAT – Editora, Universidade Católica de Pelotas - Brasil

Bibliografia

[ABR 87] ABRAMSKI, S.; JUNG, A. **Domain Theory**. London: Departament of Computer Science, Imperial College, 1987. 108p.

[AGU 98] AGUIAR, M. S. de. **Análise Formal da Complexidade de Algoritmos Genéticos**. 1998. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[ARO 94] ARORA, S. **Probabilistic Checking of Proofs and Hardness of Approximation Problems**. Londres: Department of Computer Science, Princepton University, 1994. 150 p. Technical Reports.

[ASP 91] ASPERTI, A.; LONGO, G. **Categories, Types and Structures**: An Introduction to Category Theory for the Working Computer Scientist. Cambridge: MIT Press, 1991. 306p.

[ADA 90] ADÁMEK, J.; HERRLICH, H.; STRECKER, G. E. **Abstract and Concrete Categories**: The Joy of Cats. New York: John Wiley & Sons, 1990. 482p.

[AHO 74] AHO, A.V. ; HOPCROFT, J.E.;ULLMAN, J.D. **The Design and Analysis of Computer Algorithms**. Readings: Addison-Wesley, 1974. 470p.

[AUS 99] AUSIELLO, G.; CRESCENZI P.; GAMBOSI G.; KANN V. ; MARCHETTI-SPACCAMELA A.; PROTASI M. **Complexity and Approximation**: Combinatorial Optimization Problems and their Approximability Properties. Berlim: Springer-Verlag, 1999. 524 p.

[BAA 78] BAASE, S. **Computer Algorithms**: Introduction do Design and Analysis. Readings, Mass: Addison-Wesley, 1978.

[BAL 95] BALCÁZAR, G.; DÍAZ, J.; GABARRÓ, J. **Structural Complexity I** 2nd ed. Berlin: Springer-Verlag, 1995. 208p.

[BAR 85] BARR, M.; WELLS, C. **Toposes, Triples and Theories**. [S.l: s.n.], 1985.

[BAR 90] BARR, M.; WELLS, C. **Category Theory for Computing Science**. New York: Prentice Hall, 1990. 432p.

[BAR 2001] BARBOSA, M. A. C. **ANAC – uma Ferramenta para Automatização da Análise da Complexidade de Algoritmos**. 2001. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre..

[BLU 98] BLUM, L. et al. **Complexity and Real Computation**. Heidelberg: Springer-Verlag, 1998. 453p.

[BOV 94] BOVET, D. P.; CRESCENZI, P. **Introduction to the Theory of Complexity**. Londres: Prentice Hall, 1994. 282p.

[COO 71] COOK, S.A The Complexity of Theorem Proving Procedures. In: ACM SYMPOSIUM ON THEORY OF COMPUTING, **Proceedings..**, [S.l.], 1971.

[COO 83] COOK, S.A. An overview of Computational Complexity**. Communications of the ACM**, New York, v.26, n.6, p. 401-407, June 1983.

[COO 85] COOK, S.A A Taxonomy of Problems with Fast Parallel Algorithms**. Information and Control**, New York, v.64, n. 1-3, p. 2-22, Jan/Feb/Mar 1985.

[COO 2000] COOK, S.A. The P versus NP Problem**. Millennium Prize Problems**.Clay Mathematics Institute, [S. l.], 2000.
Disponível em: < http://www.claymath.org/prize_problemsl>. Acesso em: 10 abr. 2001.

[COR 90] CORDIER, J-M.; PORTER, T. **Shape Theory**: Categorical Approximations Methods, Londres: Ellis Horwood, 1990. 202p.

[COT 90] CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L. **Introduction to Algorithms**. Cambridge MIT PRESS, 1990. 1028p.

[CRE 91] CRESCENZI, P. ; PANCONESI, A. Completeness in Approximation Classes. **Information and Computation**, [S. l.], v. 93, p. 241-262, 1991.

[CRE 95] CRESCENZI, P.; KANN, V. **A Compendium of NP Optimization Problems**. Roma: Università di Roma "La Sapienza", 1995.(T.R. SI/RR-95/02)
Disponível em: <http://www.nada.kth.se/theory/problemlist.html>. Acesso em: 01 mar. 2000.

[CUN 97] CUNHA, R. D. Computação Paralela. In**:** CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, CNMAC**,** 20, 1997, Gramado. **Resumos**. Rio de Janeiro: SBMAC, 1997.

[DIV 95] DIVÉRIO, T. A. **Uso Efetivo de Matemática Intervalar em Supercomputadores Vetoriais**. Tese (Doutorado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 1995.

[ELL 87] ELLERMAN, D. P. Category Theory and Concrete Universals. **Erkenntnis**. [S.l.], v.28, n. 3, p. 409-429, 1987.

[ELL 95] ELLERMAN, D. P. **Intellectual Trespassing as a Way of Life**: Essays in Philosophy, Economics, and Mathematics. Londres: Rowman & Littlefield Publishers, 1995.

[EMD 90] EMDE BOAS, P. van. Machine Models and Simulations. LEEUWEN, J. V. (Ed.) **Handbook of Theoretical Computer Science**. Amsterdam: Elsevier Science, 1990. p. 835-868, 1990.

[GAR 79] GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability**: a guide to the Theory of NP-Completeness. San Francisco: W. H. Freeman, 1979. 340 p.

[GOL 84] GOLDBLAT, R. **TOPOI - The Categorial Analysis of Logic**: Studies in Logic and the Foundations of Mathematics. New York: North Holland, v.98, 1984. 551p.

[GOD 89] GOLDBERG, D. Genetic Algorithms. In: **Search, Optimization and Machine Learning**. [S. l.]: Addison-Wesley, 1989. 412p

[GOL 2001] GOLDREICH, O. Computational Complexity. ENGQUIST, B.; SCHIMID, W. (Ed.) **Mathematics Unlimited – 2001 and Beyond**. New York: Springer-Verlag, 2001. p. 507-524

[GUR 91] GURR, D. J. **Semantic Frameworks to Complexity**. CST – 72-91. Thesis. Department of Computer Science. University of Edinburgh, 1991. 241p.

[HAS 94] HASTAD, J. **Recent Results in Hardness of Approximation**, New York: Springer-Verlag, p. 231-239, 1994. (Lecture Notes in Computer Science, 824)

[HOC 97] HOCHBAUM D. (Ed.) **Approximation Algorithms for NP-hard Problems**. [S. l.]: PWS Publishing Company, 1997. 596p.

[HOR 78] HOROWITZ, E.; SAHNI, S. **Fundamentals of Computer Algorithms**. Maryland: Computer Science, 1978. 626p. (Computer Software Engeneering Series)

[HUT 98] HUTH, M. **Approximative Polynomial Algorithms for Satisfiability**. [S.l.: s.n.], 1998. 21p.
Disponível em: <http://www.cis.ksu.edu/~huth>. Acesso em: 10 mai. 1997.

[IZQ 2000] IZQUIERDO, V. **Fundamentos do Simulated Annealing**. 2000. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[JAN 98] JANSEN, T. Introduction to the Theory of Complexity and Approximation Algorithms., New York: Springer-Verlag, p. 5-28, 1998. (Lecture Notes in Computer Science, 1367)

[JOH 74] JOHNSON, D. S. Approximation Algorithms for Combinatorial Problems**.** **Journal of Computing System Science**, [S. l.: s.n.], v.9, p. 256-279, 1974.

[JOH 90] JOHNSON, D. S. Algorithms and Complexity - A Catalogue of Complexity Classes. LEEUWEN, J. V. (Ed.). **Handbook of Theoretical Computer Science**. Amsterdam: Elsevier Science, p. 67-161, 1990.

[KAN 97] KANN, V.; PANCONESI, A. Hardness of Approximation. In: JOHN WILEY & SONS. **Annotated Bibliographies in Combinatorial Optimization**. [S. l.], 1997.

[KAR 76] KARP, R. M. The Probabilistic Analysis of Some Combinatorial Search Algorithms. In: TRAUB, J. F. (Ed.). SYMPOSIUM ON NEW DIRECTIONS AND RECENT RESULTS IN ALGORITHMS AND COMPLEXITY, 1976, Pittsburgh. **Algorithms and Complexity**. New York: Academic Press, 1976.

[KAR 91] KARP, R. M. **Parallel Combinatorial Computing**. Berkeley: Computer Science Division. University of California, 1991

[KLE 36] KLEENE, S. General Recursive Functions of Natural Numbers. **Matematische Annalen**. [S. l.: s. n.], v.112, p. 727-742, 1936.

[KRO 85] KRONSJÖ, L. **Computational Complexity of Sequential and Parallel Algorithms**. [S. l.], John Willey & Sons, 1985. 224p

[LAD 75] LADNER, R. On the Structure of Polynomial Time Reducibility. **Journal of the ACM**. [S.l.], v.22, p. 159-171, 1975.

[LAK 90] LAKSHIMIVARAHAN, S.; DHALL, S. K. **Analysis and Design of Parallel Algorithms - Arithmetic and Matrix Problems**. [S. l.], McGraw-Hill, 1990. 657p

[LAW 97] LAWVERE F. W.; SCHANUEL, S. H. **Conceptual Mathematics – A First Introduction to Categories**. Cambridge: Cambridge University, 1997. 358p

[LEA 97] LEAL, L. A. S. **Domínios de Scott: Teoria e Aplicações**. 1997. TI-623 (Programa de Pós-Graduação em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[LEL 97] LEAL, L. A. S. **A Teoria de Algoritmos frente à Intratabilidade de Problemas**. 1997. EQ-16 (Programa de Pós-Graduação em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[LEA 98] LEAL, L. A. S.; TOSCANI, L. V.; CLAUDIO, D. M. Approximative Algorithms through Domain Theory. In: OPTIMIZATION, 1998. **Program**. Coimbra: Apolio, 1998. p. 44.

[LEA 2000] LEAL, L. A. S.; MENEZES, P. B.; CLAUDIO, D. M.; TOSCANI, L. V. Categorias dos Problemas de Otimização. In: WORKSHOP ON FORMAL METHODS, WMF2000, 3. SBES, 14, 2000. **Resumos.** João Pessoa: SBC, p. 104-109.

[LEA 2001] LEAL, L. A. S.; MENEZES, P. B.; CLAUDIO, D. M.; TOSCANI, L. V. Optimization Problems Categories. In: MORENO-DIÁZ, R.; QUESADA-ARENCIBIA, A. (Ed.), FORMAL METHODS AND TOOLS FOR COMPUTER SCIENCE, EUROCAST, 8, 2001. **Extended Abstracts**, Las Palmas de Gran Canária: IUCTC, p. 93-96.

[LEL 2001] LEAL, L. A. S.; MENEZES, P. B.; CLAUDIO, D. M.; TOSCANI, L. V. **Optimization Problems Categories**. Berlim: Springer-Verlag, p. 285-299, 2001. (Lecture Notes in Computer Science, 2178).

[LEA 2002] LEAL, L. A. S.; CLAUDIO, D. M.; MENEZES, P. B.; TOSCANI, L. V. A Modelling the Approximation Hierarchy to Optimisation Problems Through Category Theory. **International Journal of Computing Anticipatory Systems**, Liège, v. 11, p. 336-349, 2002.

[LEN 90] LENGAUER, T. Algorithms and Complexity – VLSI Theory. LEEUWEN, J. V. (Ed.). **Handbook of Theoretical Computer Science**. Amsterdam: Elsevier Science, p. 835-868, 1990.

[LEW 81] LEWIS, H. R.; PAPADIMITRIOU, C. H. **Theory of Computation**. [S. l.] Prentice-Hall, 1981. 446p

[LOR 00] LORETO, A. **Cálculo da Complexidade Exata de Algoritmos do tipo Divisão e Conquista através das equações Características**. 2000. Dissertação (Mestrado em Matemática) – Instituto de Matemática, Universidade federal do Rio Grande do Sul, Porto Alegre.

[MAC 71] MaC LANE, S. **Categories for the Working Mathematician**. New York: Springer-Verlag, 1971.

[MEN 2001] MENEZES, P. B.; HAEUSLER, E. H. **Teoria das Categorias para Ciência da Computação**. Porto Alegre: Sagra-Luzzatto, 2001. 324p. (Livros Didáticos, nº 12).

[MEY 72] MEYER, A. e STOCKMEYER, L. The Equivalence Problem for Regular Expressions with Squaring Requires Exponencial Space. In: IEEE SYMPOSIUM ON SWITCHING AND AUTOMATA THEORY, **Proceedings..**, p. 125-129. [S. l.: s. n.], 1972.

[MOR 2000] MORELLI, C.; TOSCANI, L. V. An Experiment on mixing GRASP and HBSS to solve MAX-SAT instances. In: INTERNATIONAL WORKSHOP CP-AI-OR, OF AI AND OR TECHNIQUES IN CONSTRAINT PROGRAMMING FOR COMBINATORIAL OPTIMIZATION PROBLEMS: **Proccedings..** Paderborn: University of Paderborn, 2000.

[MOT 92] MOTWANI, R. **Lecture Notes on Approximation Algorithms**. Stanford: Department of Computer Science, Stanford University, v. 1, 1992. 134p.

[PAP 91] PAPADIMITRIOU, C. H.; YANNAKAHIS, D. Optimization, Approximation and Complexity Classes. **Journal of Computing System and Sciences**, [S. l.], v. 43, p. 425-440, 1991.

[PAP 94] PAPADIMITRIOU, C. H. **Computational Complexity**. New York: Addison-Wesley, 1994. 523p.

[PIE 91] PIERCE, B. **Basic Category Theory for Computer Scientists**. MA: MIT Press, 1991. 91p.

[PIN 83] PINHO, A. de A. **Algoritmos Polinomiais Exatos ou Aproximáticos para Certos Problemas em Scheduling**. 1983. Tese (Doutorado), Universidade Federal do Rio de Janeiro, Rio de Janeiro.

[RAB 76] RABIN, M. O. Probabilistic Algorithms. In: TRAUB, J. F. (Ed.). **Algorithms and Complexity**: New Directions and Recent Results. New York: Academic Press, p.21-40, 1976.

[RAT 94] RATTRAY, C. The Shape of Complex Systems. In: INTERNATIONAL WORKSHOP ON COMPUTER AIDED SYSTEMS THEORY, EUROCAST, 3, 1993. **Proccedings..** Berlin: Springer-Verlag, p. 72-82, 1994. (Lecture Notes in Computer Science, 763)**.**

[RAT 96] RATTRAY, C. Identification and Recognition Through Shape in Complex Systems. In: INTERNATIONAL WORKSHOP ON COMPUTER AIDED SYSTEMS THEORY, EUROCAST, 5, 1995. **Computer Aided Systems Theory**. Berlin: Springer-Verlag, p. 19-29, 1996. (Lecture Notes in Computer Science, 1030)**.**

[RAT 98] RATTRAY, C. Abstract Modelling Complex Systems. In: ALBRECHT, R. (Ed.). **Systems**: Theory and Practice, Advances in Computing Science, Viena: Springer-Verlag, p. 1-12, 1998.

[ROG 99] ROGGIA, I. B. **Métodos Heurísticos para a Solução do Problema de Sequenciamento Cíclico de n Tarefas em m Processadores Paralelos Idênticos**. 1999. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[ROS 97] ROSA, D. S. **Complexidade Média Algorítmica: uma Metodologia para o seu Cálculo**. 1997. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[SCH 97] SCHULZ, A. ; SHMOYS, D.; WILLIANSON, D. Approximation Algorithms. In: NATIONAL. ACADEMY OF SCIENCE. [S.l.: s. n.], **Proccedings..** v.94, p. 12734-12735, 1997.

[SCO 72] SCOTT, D. S. Lattice Theory, Data Types and Semantic. In: COURANT COMPUTER SCIENCE SYMPOSIUM, 2, 1970, New York. **Formal Semantics of Programming Languages**. Englewwood Cliff: Prentice Hall, p. 66-106, 1972.

[SCO 82] SCOTT, D. S. Domains for Denotational Semantics. In: COLLOQUIUM ON AUTOMATA LANGUAGES AND PROGRAMMING, 9, 1982. Aarhus. **Proccedings...**Berlin: Springer-Verlag, p. 231-252, 1982.

[SHM 95] SHMOYS, D. **Computing Near-Optimal Solutions to Combinatorial Optimizations Problems**., [S. l.: s. n.], 1995. 43p. (DIMACS Series in Discrete Mathematics and Theoretical Computer Science).

[STO 94] STOLTENBERG-HANSEN, V.; LINDSTRÖM, I.; GRIFFOR, E. **Mathematical Theory of Domains**. Cambridge: University of Upsala., 1994. 349p.

[SZW 84] SZWARCFITER, J. **Grafos e Algoritmos Computacionais**. Rio de janeiro: Editora Campus, 1984. 216p

[TEN 75] TENNISON, B. R. **Sheaf Theory**. Cambridge: Cambridge University Press, 1975. 164p.

[TOS 86]] TOSCANI, L. V.; SZWARCFITER, J. **Algoritmos Aproximativos**. Porto Alegre: CPGCC-UFRGS. 1986. (RP-58)

[TOC 86]] TOSCANI, L. V.; SZWARCFITER, J. Algoritmos Aproximativos: uma Alternativa para Problemas NP-Completos. In: CONFERÊNCIA LATINOAMERICANA DE **N**FORMÁTICA, CLEI, 12, 1986, Montevideo, **Anales**. Montevideo: CLEI, p. 155-166, 1986.

[TOS 88] TOSCANI, L. V. **Métodos de Desenvolvimento de Algoritmos: Especificação Formal, Análise Comparativa e de Complexidade**. 1988. Tese (Doutorado em Ciência da Computação), Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

[TOS 90] TOSCANI, L. V.; VELOSO, P. A. S. Uma Metodologia para o Cálculo da Complexidade de Algoritmos. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 4. Águas de São Pedro, 1990. **Anais**. São Paulo: SBC, 1990.

[TOS 2001] TOSCANI, L. V.; VELOSO, P. A. S. **Complexidade de Algoritmos**: análise, projeto e métodos. Porto Alegre: Sagra-Luzzatto, 2001. 202p. (Livros Didáticos, nº 13).

[TRE 97] TREVISAN, L. **Reductions and (Non-)Approximability**. 1997. Thesis (Doutorado), Università degli Studi di Roma “La Sapienza”, Roma.

[TUR 36] TURING, A. On Computable Numbers, with an Application to the Entscheidunhsproblem. In: LONDON MATHEMATICAL SOCIETY. **Proccedings..** Oxford, v. 42, n. 2, p 230-265, 1936.

[VEL 84] VELOSO, P. A. S. Aspectos de uma Teoria Geral de Problemas. **Cadernos de História e Filosofia da Ciência**, Campinas, n. 7, p. 21-42, 1984.

[ZOM 96] ZOMAYA, A. Y. **Parallel and Distributed Computing Handbook**. New York: McGraw-Hill, 1996. 1198p.