Algoritmo Genético Guiado por Códigos Algébricos

Alynthor de Lima Araújo

Dissertação de Mestrado submetida à Coordenação dos Cursos de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Mestre no domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação

Francisco Marcos de Assis, Doutor
Orientador

Campina Grande, Paraíba, Brasil

A663a Araujo, Alynthor de Lima
Algoritmo genetico guiado por codigos algebricos / Alynthor de Lima Araujo. - Campina Grande, 2002.
79 f.

Dissertacao (Mestrado em Engenharia Eletrica) - Universidade Federal de Campina Grande, Centro de Ciencias e Tecnologia.

1. Algoritmo Genetico 2. Codigos Algebricos 3. Classe Lateral 4. Dissertacao I. Assis, Francisco Marcos de II. Universidade Federal de Campina Grande - Campina Grande (PB) III. Título

CDU 519.726:621.391(043)

RELAÇÃO ENTRE ALGORITMO GENÉTICO E CÓDIGOS ALGÉBRICOS

ALYNTHOR DE ARAÚJO LIMA

Dissertação Aprovada em 12.09.2002

PROF. FRANCISCO MARCOS DE ASSIS, Dr., UFCG
Orientador

PROF. BENEMAR ALENCAR DE SOUZA, D.Sc., UFCG
Componente da Banca

PROF. JOÃO MARQUES DE CARVALHO, Ph.D., UFCG
Componente da Banca

CAMPINA GRANDE - PB
Setembro – 2002

Dedicatória

Esta dissertação é dedicada aos meus pais, a minhas irmãs e a minha noiva.

Agradecimentos

- A Deus, por tudo;
- Aos meus pais, Ailton e Lindomar, pelo amor, compreensão e incentivo em toda minha vida;
- A minhas irmãs, Loudmylla e Ábia, pelo incentivo e carinho;
- À minha noiva Isabella, pelo amor e incentivo durante estes anos;
- Ao Prof. Francisco Marcos de Assis, pela orientação, incentivo, amizade e compreensão;
- Aos Professores Bruno Albert, João Marques e José Ewerton, pelos ensinamentos e amizade;
- Aos demais professores do DEE-UFCG, pelos ensinamentos e incentivo;
- Aos colegas da pós-graduação Waslon, Edmar, Luiz Felipe, Juraci, Lidiano, Gustavo, Mohit, Sérgio, Madhavan, Hallyson, Ronaldo e aos demais colegas que contribuiram direta e indiretamente para realização deste trabalho;
- Aos funcionários do DEE;
- À Coordenação do Curso de Pós-Graduação em Eng. Elétrica da UFCG;
- Ao CNPQ e à CAPES que financiaram este trabalho.

Resumo

Algoritmos evolucionários são amplamente utilizados em otimização de funções. Algoritmos Genéticos(AG) são algoritmos evolucionários baseados na seleção natural usando operadores genéticos probabilísticos. Apesar de sua eficácia em alguns problemas de otimização, os AGs apresentam também resultados não-aceitáveis em funções de difícil solução. A idéia de utilizar os códigos algébricos para guiar o AG, explorando a estrutura dos códigos para auxiliar o AG a superar problemas com funções do tipo "armadilha"é proposta neste trabalho. Esta dissertação apresenta um algoritmo que combina o AG com os códigos algébricos e também propõe a utilização do conceito de classes laterais.

Abstract

Evolutionary algorithms have been widely used in function optimization. Genetic Algorithms(GA) are evolutionary algorithms based in natural selection using probabilistic genetic-like operators. Despite of being efficient in some optimization problems, GAs have not performed so well with some "trap" functions. The idea of algebraic code introduction in GAs is to guide GA exploiting code structure for helping a GA to overcome troubles with trap functions is proposed in this work. This work presents an algorithm which combine GA with algebraic codes and also propose the code coset concept utilization.

Lista de Figuras

Lista de Tabelas

Conteúdo

Capítulo 1

Introdução

Nos últimos anos a busca pela resolução de problemas de otimização se intensificou devido ao grande número de casos existentes, sendo desenvolvidos vários métodos para solucionar problemas de otimização, tais como "simulated annealing", algoritmos evolucionários, etc.

Dentre todos algoritmos evolucionários se destaca o AG que baseado na seleção natural usando operadores genéticos probabilísticos. Pesquisas com AG indicam sua eficácia na solução de diversos problemas de otimização, entretanto há problemas em que os AGs exibem deficiências tais como convergência prematura ou simplesmente problemas em que não convergem para soluções aceitáveis, o que ocorre nas denominadas funções-armadilha.

Neste trabalho são apresentados dois algoritmos de otimização que combinam o algoritmo genético com a teoria dos códigos algébricos visando corrigir os problemas apresentados no algoritmo genético padrão em determinadas funções. Resultados obtidos [1] [2] [3] comprovam a vantagem do estudo e a utilização dos códigos na estrutura do algoritmo genético em certas funções multimodais.

Na transmissão e armazenamento digital de dados, há sempre a possibilidade de ocorrência de erros. O uso de códigos corretores de erros é uma constante nos meios de comunicação, códigos como BCH, Goppa, RS, Hamming são utilizados em aplicações que variam de CD player à transmissão para satélite. Os códigos algébricos baseiam-se no sistema algébrico desenvolvido por *Galois*, os Corpos Finitos.

A estrutura dos códigos algébricos é uma estrutura no espaço das sequências de comprimento determinado, por exemplo, códigos binários definem uma estrutura no espaço $\mathcal{S}^n = \{0,1\}^n$. A motivação desta dissertação é explorar a estrutura dos códigos para auxiliar o AG melhorando sua eficiência..

A dissertação está organizada da seguinte forma: No Capítulo 2 é apresentado o AG. Algumas características e propriedades são levantadas, assim como o uso de determinados operadores. Também é apresentada a teoria dos "esquemas"que descreve o funcionamento do AG, sob o ponto de vista da convergência.

No Capítulo 3 são mostrados dois algoritmos baseados nos AGs propostos no trabalho. O primeiro baseia-se na junção do AG com os códigos algébricos. O segundo introduz o conceito de classes laterais no anterior.

No Capítulo 4 são comparados os algoritmos propostos com o AG Padrão em um sistema de comunicações, bem como em funções de difícil solução.

No Capítulo 5, as conclusões e sugestões para trabalhos futuros são apresentados.

No Apêndice A é apresentado um resumo dos corpos finitos no qual são também mostrados os conceitos de corpos, grupos, anéis e bases.

No Apêndice B é apresentada a introdução aos códigos algébricos no qual são mostradas suas propriedades, seus tipos de códigos e sua introdução para controle de erros.

Capítulo 2

Algoritmo Genético

Neste capítulo o AG é introduzido: como são constituídos, como diferem dos outros métodos de busca e a teoria dos "esquemas" que ajuda a compreender seu funcionamento.

2.1 Algoritmos de Busca

Uma grande parte dos trabalhos científicos pode ser formulada como problemas de busca de otimização. O objetivo dos algoritmos de otimização é encontrar uma combinação de valores que proporcione o desempenho ótimo ou sub-ótimo de um sistema considerado. O conjunto de todas as combinações possíveis para constitui o chamado *espaço de busca.* Sem perda de generalidade, considera-se somente métodos de maximização.

Dada uma função $f : \Re^n \rightarrow \Re$ e um espaço de $S \subseteq \Re^n$, o método pode ser formulado assim:

$$encontrar\ x^* | f(x^*) \geq f(x), \forall x \in S \tag{2.1}$$

A função $f(\cdot)$ pode ser derivável ou não, uni- ou multidimensional, e o espaço de busca S pode ser contínuo ou discreto, finito ou infinito, côncavo ou convexo. O problema é chamado unimodal quando há somente um ponto máximo x^* no espaço de busca, ou multimodal em caso contrário.

Numa classificação grosseira, há essencialmente três correntes de métodos: probabilísticos, numéricos e enumerativos. Há ainda um grande número de métodos híbridos.

Consideremos um problema de maximização de uma função $f : \Re \rightarrow \Re$ dada por

$$f(x) = \cos(20 \cdot x) - \frac{|x|}{2} + \frac{x^3}{4} \tag{2.2}$$

no intervalo $-2 \leq x \leq 2$. Trata-se de uma função não-linear com 13 picos no intervalo de interesse, como mostra na Figura 2.1, sendo um correspondente ao máximo global, ou ponto-ótimo, $x^* \approx 1,88929$.

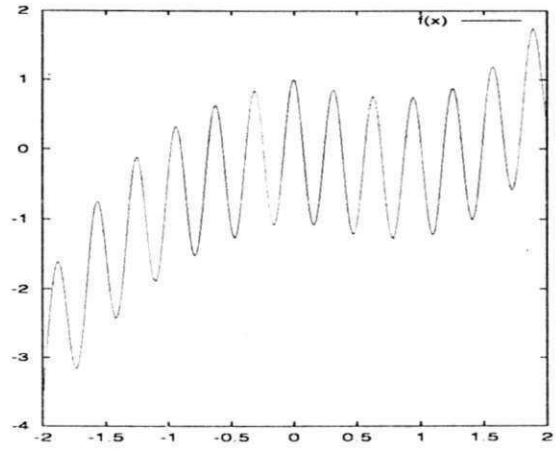

Figura 2.1: Gráfico de $f(x)$

Substituindo x^* na função, obtem-se,

$$f(x^*) \approx 1,73752 \geq f(x), \forall x, -2 \leq x \leq 2. \tag{2.3}$$

Os pontos extremos de $f(\cdot)$ podem ser determinados igualando-se a primeira derivada de $f(\cdot)$ a zero e resolvendo a equação transcendental resultante analiticamente. Se a expressão analítica de $f(\cdot)$ for desconhecida, porém, métodos numéricos são impotentes. Por se tratar de uma função multimodal, é evidente que métodos de gradiente puros não devem ser capazes de encontrar o ótimo global na maior parte das tentativas.

Otimização de funções é um bom exemplo para examinar a robustez de qualquer procedimento de busca. Uma estratégia de busca possível é a busca exaustiva que é

interessante em funções no qual o espaço de busca é pequeno pois pelo contrário tomariam um grande esforço computacional. Há também métodos baseados no cálculo diferencial. Deriva-se a função e iguala a zero, encontrando assim seus máximos ou mínimos absolutos. O grande problema destes métodos é a sua ineficiência com funções multimodais. Outro método usado é o *Simulated Annealing*, método no qual se usa processos aleatórios para guiar sua busca por estados de mínima energia. Os problemas de otimização de funções podem ser atacados por diversas técnicas que John Holland [4] desenvolveu o algoritmo genético na Universidade de Michigan. Neste capítulo encontram-se definições, características e exemplos dos AGs. Na Seção 2.2 pode-se encontrar uma apresentação do AG, bem como conceitos de seus principais operadores. A Seção 2.3 apresenta alguns fundamentos matemáticos do AG.

2.2 Algoritmo Genético

Os algoritmos genéticos emulam a teoria da evolução biológica para resolver problemas de otimização. Um AG compreende um conjunto de elementos - a *população* e um conjunto de operadores, inspirados na biologia, definidos sobre a população. De acordo com as teorias evolucionárias, os elementos da população melhor adaptados têm maior probabilidade de transmitir sua herança genética para as novas gerações.

O que caracteriza singularmente um AG é que se trata de um algoritmo "cego" no sentido em que a sua ligação com o problema se dá apenas por meio da função objetivo e por fim ele é um algoritmo de regras de transição probabilísticas, não determinísticas.

2.2.1 Representação Cromossômica

O primeiro passo para aplicação do Algoritmo Genético a um problema qualquer é representar cada solução eventual x no espaço de busca como uma sequência de símbolos s gerados a partir de um dado alfabeto finito A. No caso mais simples, usa-se o alfabeto binário $A = \{0,1\}$ mas, no caso geral, tanto o método de representação quanto o alfabeto genético dependem de cada problema. Usaremos algumas expressões que são bastante empregadas pelos teóricos e praticantes de AGs. Cada sequência s correspon-

de a um *cromossomo* e cada elemento de s é equivalente a um *gene*. Como cada gene pode assumir qualquer valor do alfabeto A, cada elemento de A é equivalente a um *alelo*, ou seja, um valor possível para um dado gene. A posição de um gene(Figura 2.2) num cromossomo, seu índice dentro da sequência, corresponde a um *locus gênico*. Além disso, na maior parte dos AGs atribui-se que cada indivíduo seja constituído de um único cromossomo, razão pela qual é comum se usarem os termos indivíduo e cromossomo indistintamente em trabalhos científicos e livros-textos. A maioria dos AGs propostos na literatura operam com uma população com número fixo de indivíduos e cromossomos de tamanho constante.

Tendo definido a representação cromossômica para o problema, gera-se um conjunto de soluções eventuais, chamadas de *soluções-candidatas*. Um conjunto de soluções codificadas de acordo com a representação selecionada corresponde a uma população de indivíduos, $P(0)$. AGs são algoritmos iterativos e a cada iteração a população é modificada. Cada iteração de um AG é denominada uma *geração*, embora nem todos os indivíduos de uma população sejam necessariamente "*filhos*" de indivíduos da população na iteração anterior. A população inicial de N indivíduos é gerada aleatoriamente. De acordo com a teoria da seleção natural, a população inicial deve cobrir a maior área possível do espaço de busca, pois os indivíduos devem ter diferentes graus de adaptação ao ambiente em que vivem.

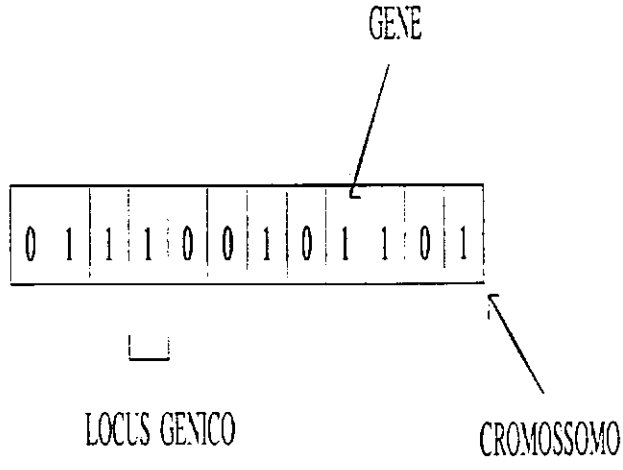

Figura 2.2: Cromossomo

Os AGs necessitam da informação do valor de uma função objetivo para cada membro da população. A função objetivo dá, para cada indivíduo, uma medida de quão bem adaptado ao ambiente ele está, ou seja, quanto maior for a sua função objetivo, maior serão as chances do indivíduo sobreviver no ambiente e reproduzir-se, passando parte de seu material genético a gerações posteriores, como também avalia se vão haver indivíduos extintos. A avaliação de cada indivíduo resulta num valor denominado "adequabilidade". Em populações naturais, a adequabilidade é determinada pela abilidade do indivíduo de sobreviver a predadores, pestes e outros obstáculos à vida adulta e sua subsequente reprodução. A função que avalia a adequabilidade é, talvez, a parte mais importante de um AG. Ela tem o potencial de afetar severamente o desempenho geral do algoritmo em termos da qualidade dos resultados e tempo de execução.

2.2.2 Operadores Genéticos

Definida a função objetivo, define-se um conjunto de operações genéticas simples que atuam sobre população inicial e geram populações sucessivas.

Um AG simples que rende bons resultados em muitos problemas práticos é composto de três operadores principais:

- Seleção e Reprodução
- Recombinação(Crossover)
- Mutação

2.2.3 Seleção

Seleção é um processo no qual cromossomos individuais são copiados de acordo com os valores de suas funções objetivos. Copiar cromossomos de acordo com os valores de suas adequabilidades significa dizer que os cromossomos com valores maiores têm uma maior probabilidade de contribuir com um ou mais descendentes na próxima geração de indivíduos. Estes cromossomos copiados são denominados de cópias. O mecanismo de seleção em AGs emula os processos de reprodução assexuada e seleção natural, uma sobrevivência darwiniana do mais adequado dentre todos os indivíduos.

Este operador pode ser aplicado de maneiras distintas, pois existem 4 principais esquemas de seleção que são mais comumente usados [5]:

- Seleção por posto
- Seleção por torneio
- Seleção por genitor
- Reprodução proporcional

Seleção por posto

Baker [6] introduziu a noção de seleção por posto ao algoritmo genético prático no qual ordena a população do melhor ao pior indivíduo, designa-se o número de cópias que cada indivíduo deverá receber de acordo com uma função atribuída não-crescente, e então usa a seleção proporcional de acordo com a dada atribuição.

Seleção por torneio

Neste caso, escolhe-se indivíduos aleatoriamente de uma população(com ou sem reposição), seleciona-se o melhor indivíduo deste grupo para processamento genético futuro, e repete até que a nova população esteja completa.

Seleção por genitor

Neste caso, é mostrado de forma bem sucinta o método usado para executar este tipo de seleção [7]. Seleção por genitor trabalha indivíduo por indivíduo, escolhendo um descendente por nascimento de acordo com um ranking linear, e escolhe-se o pior indivíduo atual para reposição. Este método extingue os indivíduos menos adaptados ao ambiente de acordo com um "ranking" e os substitui por descendentes dos indivíduos melhor adaptados.

A adequabilidade deve ser um valor não-negativo. Com isto, os indivíduos com baixa adequabilidade relativa terão alta probabilidade de serem extintos, ao passo que indivíduos bem adequados terão grandes chances de sobreviverem. Posteriormente, determina-se o número de indivíduos esperado de acordo com a $adeq_{rel}$, conforme Eq. 2.5

$$n_{ind} = N \cdot adeq_{rel} \tag{2.5}$$

Exemplo 1 *Considerando uma função* $f(x) = x^2$, *para* $0 \leq x \leq 31$, *temos uma problema de maximização, no qual* $\max(f(x)) = (31)^2 = 961$.

- Com o tamanho da população definida, neste caso $N = 4$, são gerados todos os indivíduos aleatoriamente como mostrado na Tabela 2.1 na coluna 2;
- Determinam-se suas adequabilidades de acordo com a função objetivo $f(x) = x^2$ que estão representadas na coluna 4 e com $\frac{f_i}{\sum f_i}$, as adequabilidades relativas são determinadas;
- Com estes dados, o número de indivíduos esperado na nova população é achado por $n_{ind} = N \cdot adeq_{rel}$ e o número de indivíduos real é encontrado por arredondamento simples para o inteiro mais próximo. Pode-se perceber que o indivíduos 3 foi extinto.

Neste processo algumas vezes os indivíduos menos adequados ao ambiente são deixados para poderem trocar sua herança genética com outros com melhor adequabilidade para eventualmente formarem melhores indivíduos. Sendo assim pode-se evitar a falsa convergência da população para indivíduos supostamente melhor adaptados.

2.2.4 Recombinação(*Crossover*)

Após a seleção e reprodução aparece o mecanismo da recombinação(*crossover*) que retrata um processo sexuado- ou seja, envolve mais de um indivíduo - no qual ocorre a troca de fragmentos entre pares de cromossomos. Este processo se procede em dois passos. Primeiro os indivíduos da população selecionada são divididos aleatoriamente

Reprodução proporcional

Este operador pode ser implementado na forma de algoritmo em diversas maneiras. Uma delas é criar uma roleta polarizada no qual cada cromossomo da população tem um espaço na roleta com um tamanho na proporção da sua adequabilidade como mostrado na Figura 2.3. No nosso AG utilizamos este esquema de seleção por se tratar de um método eficiente e mais simples.

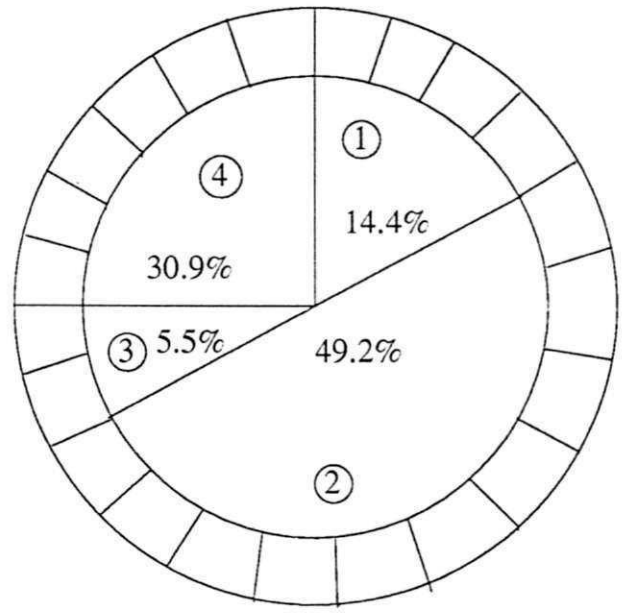

Figura 2.3: Roleta Polarizada

Logo para determinar os descendentes dos indivíduos da população roda-se a roleta polarizada de acordo com o tamanho da população. Estes descendentes são cópias exatas dos indivíduos sorteados.

Outra maneira de executar este operador é usar uma função de adequabilidade relativa que gera para uma população temporária de N indivíduos suas adequabilidades relativas respectivas, que são dadas pela Eq. 2.4:

$$adeq_{rel} = \frac{a(s)}{\sum a(s_i)} \tag{2.4}$$

no qual $a(s)$ é a função de adequabilidade.

A adequabilidade deve ser um valor não-negativo. Com isto, os indivíduos com baixa adequabilidade relativa terão alta probabilidade de serem extintos, ao passo que indivíduos bem adequados terão grandes chances de sobreviverem. Posteriormente, determina-se o número de indivíduos esperado de acordo com a $adeq_{rel}$, conforme Eq. 2.5

$$n_{ind} = N \cdot adeq_{rel} \tag{2.5}$$

Exemplo 1 *Considerando uma função* $f(x) = x^2$, *para* $0 \leq x \leq 31$, *temos uma problema de maximização, no qual* $\max(f(x)) = (31)^2 = 961$.

- Com o tamanho da população definida, neste caso $N = 4$, são gerados todos os indivíduos aleatoriamente como mostrado na Tabela 2.1 na coluna 2;
- Determinam-se suas adequabilidades de acordo com a função objetivo $f(x) = x^2$ que estão representadas na coluna 4 e com $\frac{f_i}{\sum f_i}$, as adequabilidades relativas são determinadas;
- Com estes dados, o número de indivíduos esperado na nova população é achado por $n_{ind} = N \cdot adeq_{rel}$ e o número de indivíduos real é encontrado por arredondamento simples para o inteiro mais próximo. Pode-se perceber que o indivíduos 3 foi extinto.

Neste processo algumas vezes os indivíduos menos adequados ao ambiente são deixados para poderem trocar sua herança genética com outros com melhor adequabilidade para eventualmente formarem melhores indivíduos. Sendo assim pode-se evitar a falsa convergência da população para indivíduos supostamente melhor adaptados.

2.2.4 Recombinação(*Crossover*)

Após a seleção e reprodução aparece o mecanismo da recombinação(*crossover*) que retrata um processo sexuado- ou seja, envolve mais de um indivíduo – no qual ocorre a troca de fragmentos entre pares de cromossomos. Este processo se procede em dois passos. Primeiro os indivíduos da população selecionada são divididos aleatoriamente

Ind.	Pop. Inicial	Valor x	$f(x)$ x^2	$adeq_{rel}$ $\frac{f_i}{\sum f_i}$	n_{ind}	Número Ind. Esperado Real
1	01101	13	169	0.14	0.58	1
2	11000	24	576	0.49	1.97	2
3	01000	8	64	0.056	0.22	0
4	10011	19	361	0.31	1.23	1
Soma			1170	1.0	4.00	4
Média			293	0.25	1.00	1.0
Máximo			576	0.49	1.97	2.0

Tabela 2.1: Adequabilidade Relativa

em duas novas populações com $\frac{N}{2}$ indivíduos(o que indica que N deve ser par) e formando pares entre os indivíduos da cada população. Segundo, determina-se se cada par de indivíduos passa pela recombinação. Trata-se de um processo aleatório que ocorre com probabilidade fixa p_c especificada pelo usuário. Em caso afirmativo, escolhe-se um ponto de corte aleatoriamente e efetua-se a troca de segmentos correspondentes. Podemos verificar o exemplo a seguir:

Exemplo 2 *Crossover entre 2 indivíduos*

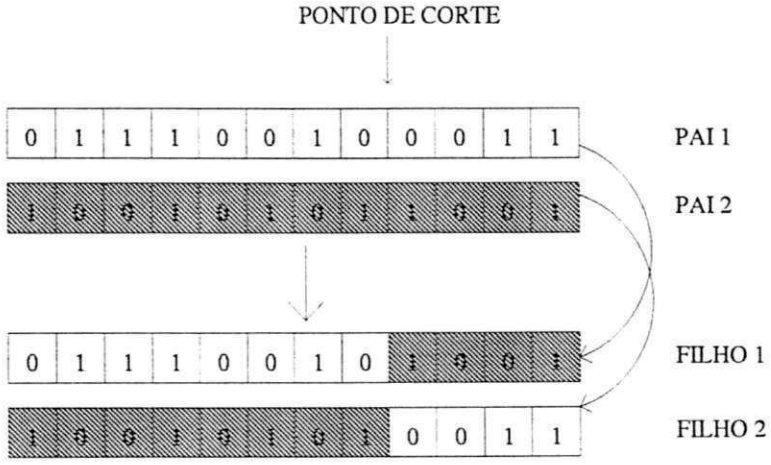

Figura 2.4: Crossover

População após Seleção	Nova População	Valor x	$f(x)$ x^2
0110—1	01100	12	144
1100—0	11001	25	625
11—000	11011	27	729
10—011	10000	16	256

Tabela 2.2: Crossover

Com estes novos cromossomos "filhos", o emparelhamento é desfeito e tem-se uma nova população, diferente da população resultante do processo de seleção, contendo N indivíduos. Pode-se ver na Tabela 2.2 um exemplo do operador "crossover", no qual os indivíduos da nova população gerada a partir da Tabela 2.1 exexutam o "crossover"de acordo com a $p_c = 0,6$. Determinados os pontos de corte, é feito o "crossover"e a nova população é determinada. Numa análise ampla percebe-se que o crossover faz uma busca "local"no espaço de busca. O valor de p_c foi encontrado de forma empírica e pode variar entre $0,6 \leq p_c \leq 0,7$.

2.2.5 Mutação

Trata-se de uma busca aleatória em que seleciona-se uma posição num cromossomo e muda-se o valor do gene correspondente aleatoriamente para um outro alelo possível, este processo é controlado pela probabilidade fixa p_m(também encontrado de forma empírica e pode variar entre $0,05 \leq p_m \leq 0,1$) , que indica a probabilidade de um gene sofrer mutação.

Existem muitas discordâncias entre os pesquisadores a respeito do papel da mutação na genética (seja natural ou artificial), mas segundo Goldberg [8] a mutação assume um papel secundário nos AGs. Em sistemas genéticos artificiais, o operador de mutação os protege contra uma perda irrecuperável, tal como algum material genético(1's ou 0's em posições específicas) potencialmente útil que os operadores anteriores introduzem no algoritmo. Nota-se que a freqüência da mutação que obtém bons resultados em estudos empíricos sobre o AG é na ordem de uma mutação por 1000 bits(posições)

verificadas. Taxas de mutação são similarmente baixas(ou menores) em populações naturais, nos levando a concluir que a mutação é considerada apropriadamente como um mecanismo secundário da adaptação do AG. Como no caso do crossover, em uma visão ampla a mutação é uma busca aleatória "global"através do espaço de busca. Este operador é apresentado na Figura 2.5

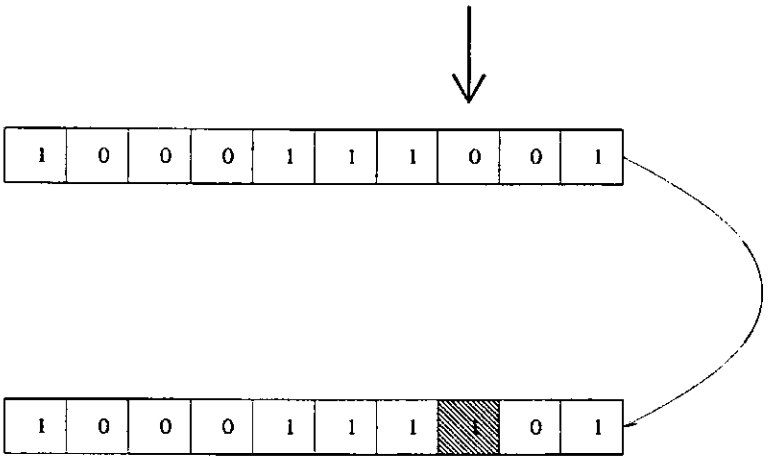

Figura 2.5: Mutação

Outros microoperadores podem ser observados na natureza, operadores genéticos que operam no nível do cromossomo. Além dos mais conhecidos e mais representativos operadores, existem outros tais como dominância, diploidia, segregação, translocação, duplicação e apagamento. Dominância e diploidia são métodos de implementar uma grande memória da população. Estes operadores são úteis em funções não-estacionárias, especialmente funções cíclicas. Segregação e translocação requerem a extensão do genótipo para incluir cromossomos múltiplos. Duplicação e apagamento também causa uma perda de noção de representação apropriada do cromossomo; eles nos forçam a considerar representações de tamanhos variáveis. Como no nosso caso especial, consideramos cromossomos de comprimento fixo, portanto estes últimos operadores são inviáveis.

2.2.6 Condições de Parada

Os indivíduos de cada geração são ordenados de acordo com suas adequabilidades. Ao longo das gerações as adequabilidades das novas populações são encontradas até encontrar uma solução ótima ou sub-ótima. É necessário estabelecer uma condição de parada que pode ser de várias maneiras:

- Estagnação, ou seja, quando não é observada nenhuma melhoria da população depois de várias gerações, o algoritmo encerra a busca;
- Número máximo de gerações;
- Tempo limite de processamento.

2.2.7 Pseudo-Código

Designando cada geração por um índice t, o algoritmo geral de um AG simples é descrito abaixo, cuja condição final é o número máximo de gerações.

Algoritmo 1 AG
Inicialização

- $t \leftarrow 0$;
- *inicializar* $P(t) \subset \{0,\ 1\}^n$;
- *avaliar* $P(t)$;

Iteração

Enquanto $Final = FALSO$ **faça**

- $t \leftarrow t + 1$;
- *selecionar* $P(t)$ *a partir de* $P(t-1)$;
- *recombinar;*
- *mutar;*

- *avaliar* $P(t)$;

Fim
Fim

2.3 Fundamentos Matemáticos do AG

Em geral é mais importante a relação entre os cromossomos do que os cromossomos individualmente. Desde que similaridades importantes entre cromossomos com adequabilidade alta podem ajudar a guiar a busca, pergunta-se: como um cromossomo pode ser similar aos seus equivalentes?

Um "esquema" [4] é um modelo de similaridade descrevendo um subconjunto de cromossomos com similaridades em certas posições. Sem perda de generalidade, consideramos um alfabeto binário $\{0,1\}$. Motiva-se um esquema mais facilmente por anexar um símbolo especial neste alfabeto: $*$, que significa um símbolo "tanto faz". Com isto o alfabeto se extende a um alfabeto ternário $\{0,1,*\}$. O esquema $*0000$ corresponde a dois cromossomos $\{00000, 10000\}$. Um outro exemplo, o esquema $*111*$ descreve o subconjunto com quatro membros $\{01110, 01111, 11110, 11111\}$, ou também o esquema $0*1**$ corresponde a oito cromossomos de comprimento 5 que começa com 0 e tem um 1 na posição 3. A idéia de esquema dá uma maneira poderosa e compacta de falar sobre todas as similaridades bem definidas entre cromossomos de comprimento finito num alfabeto finito.

Do ponto de vista da teoria dos códigos algébricos, um esquema é uma classe lateral, vista no Apêndice B, pois

$$\begin{aligned} i_1, i_2 \in \{0,1\} \quad (*111*) &= i_1(10000) + i_2(00001) + (01110) \\ &= (i_1 i_2) \begin{pmatrix} 10000 \\ 00001 \end{pmatrix} + (01110) \end{aligned} \tag{2.6}$$

O resultado da Eq. 2.6 é semelhante ao da Eq. B.27, pois $\begin{pmatrix} 10000 \\ 00001 \end{pmatrix}$ é semelhante à matriz geradora $\mathbf{G}$ e (01110) é semelhante a uma cadeia de S^l, v, portanto

$$(i_1 i_2)\begin{pmatrix} 10000 \\ 00001 \end{pmatrix} + (01110) = i\mathbf{G} + v \tag{2.7}$$

Em geral, para alfabetos de cardinalidade "k"existem $(k+1)^l$ esquemas, no qual l é comprimento do cromossomo, por que cada uma das k posições pode ser $0, 1$ ou $*$. Considerando um cromossomo simples de extensão l, ele terá k^l esquemas, por exemplo, 11111 tem 2^5 esquemas porque ele pode assumir seu valor atual ou um símbolo "tanto faz". em geral cromossomos específicos contêm k^l esquemas. Como resultado, uma população de tamanho N possui entre k^l e $N \cdot k^l$ esquemas dependendo da diversidade da população. Pode-se verificar o efeito deste processo no AG.

2.3.1 Teorema Fundamental do Algoritmo Genético

Este teorema tem como objetivo determinar o número esperado de cópias do "esquema"na próxima geração.

Considere um esquema H com um alfabeto ternário $V = \{0, 1, *\}$. Nem todos os esquemas são criados igualmente, alguns são mais específicos que outros por exemplo, $011*1**$ é mais específico do que $0******$. Para quantificar esta idéia são introduzidas duas definições do esquema: conceitos de ordem e comprimento definido.

Definição 1 *A ordem de um esquema H, denotado por $o(H)$ é simplesmente o número de posições fixas presentes no modelo.*

Exemplo 3

- esquema $011*1** \rightarrow o(H) = 4$
- esquema $0****** \rightarrow o(H) = 1$

Definição 2 *O comprimento definido de um esquema H denotado por $\delta(H)$, é a distância entre a primeira e a última posição ocupadas por símbolos diferentes de $*$ do cromossomo.*

Exemplo 4

- esquema $011*1** \rightarrow \delta(H) = 5 - 1 = 4$
- esquema $0****** \rightarrow \delta(H) = 0$

Considera-se o efeito individual e combinado da reprodução, crossover e mutação no esquema contido de uma população de cromossomos. O efeito da reprodução no número esperado de esquema na população é particularmente fácil de determinar. Com o cromossomo A_i representado simbolicamente como:

$$A_i = a_{1i}a_{2i}\cdots a_{li} \tag{2.8}$$

sendo $i = 1, 2, \cdots, N$. suponha que a uma dada iteração t existem m exemplares de um esquema particular H contidos na população $P(t) = \{A_1, \cdots, A_N\}$, o qual se escreve como $m = m(H, t)$(existem possivelmente quantidades diferentes de esquemas diferentes H em iterações t diferentes). Durante a reprodução um cromossomo A_i é selecionado com probabilidade $p_i = \frac{f_i}{\sum f_j}$. Espera-se ter $m(H, t+1)$ representantes do esquema H na população na iteração $t+1$ como dado na equação 2.9

$$m(H, t+1) = m(H, t) \cdot N \cdot \frac{f(H)}{\sum f_j} \tag{2.9}$$

na qual $f(H)$ é a adequabilidade média dos cromossomos representando o esquema H na iteração t. Como a média das adequabilidades da população inteira pode ser escrita como

$$\overline{f} = \frac{\sum f_j}{N}, \tag{2.10}$$

portanto

$$m(H, t+1) = m(H, t)\frac{f(H)}{\overline{f}} \tag{2.11}$$

De acordo com 2.11 um esquema em particular cresce de acordo com a razão entre a adequabilidade média do esquema pela adequabilidade média da população. entre outras palavras, esquemas com adequabilidades maiores que a média da população receberão um número crescente de amostras na próxima geração, ao passo que esquemas com adequabilidades menores receberão um número decrescente de amostras na próxima geração. Todos os esquemas na população crescem ou decaem de acordo com

suas média de esquemas sob o operador de reprodução. Qualitativamente falando, esquema acima da média cresce e esquema abaixo da média é extinto. Quantitativamente falando, reprodução aloca um número de tentativa crescente(decrescente) exponencialmente para cima(para baixo) da média do esquema.

Cruzamento é uma troca de informações aleatórias estruturadas entre cromossomos. Segundo este operador o esquema H é destruído com probabilidade $p_d = \frac{\delta(H)}{l-1}$ e sobrevive com probabilidade $p_s = 1 - p_d$. Se o crossover é executado aleatoriamente com probabilidade p_c, a probabilidade de sobrevivência pode ser expressa por

$$p_s \geq 1 - p_c \frac{\delta(H)}{l-1} \tag{2.12}$$

Combinando os dois operadores: reprodução e crossover

$$m(H, t+1) = m(H, t) \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{l-1}\right] \tag{2.13}$$

O efeito combinado deles é obtido ao multiplicar o número esperado de esquemas pela probabilidade de sobrevivência p_s. O esquema H cresce ou decai dependendo de um fator multiplicativo. Logo, deste fator depende se o esquema está acima ou abaixo da média da população e se o esquema tem comprimento curto ou longo relativamente.

O último fator a ser considerado é a mutação, que é a troca aleatória de uma posição simples do cromossomo com probabilidade p_m. Para um esquema H sobreviver todas as posições definidas devem sobreviver. Portanto, desde que cada alelo sobreviva com probabilidade $(1 - p_m)$, e cada uma das mutações é estatisticamente independentes, um esquema particular sobrevive quando cada um dos $o(H)$ posições fixas dentro do esquema sobrevive. Multiplicando a probabilidade de sobrevivência $(1 - p_m)$ por ela mesma $o(H)$ vezes, obtem-se a probabilidade de sobrevivência à mutação, $(1-p_m)^{o(H)}$. Para pequenos valores de $p_m (p_m \ll 1)$, a probabilidade de sobrevivência do esquema pode se aproximada pela expressão $1 - o(H)p_m$.

Logo o número esperado de cópias na próxima geração é dado pela Equação 2.14.

$$m(H, t+1) = m(H, t) \frac{f(H)}{\bar{f}} \left[1 - p_c \frac{\delta(H)}{l-1} - o(H)p_m\right] \tag{2.14}$$

A adição da mutação muda a conclusão prévia um pouco. Esquema acima da média, curto e de baixa ordem recebe tentativas crescentes exponencialmente na gerações subsequentes.

Este é o Teorema Fundamental do Algoritmo Genético. Este teorema mostra o número esperado de cópias do "esquema"na próxima geração. Maiores detalhes a respeito deste teorema pode ser visto em Goldberg [5].

2.4 Conclusão

Neste capítulo foi apresentado o algoritmo genético, sua estrutura básica, seus operadores principais, alguns fundamentos matemáticos e um exemplo de seu funcionamento. Apesar do AG apresentar um bom desempenho em algumas funções de otimização, este apresenta resultados não desejados em outra classe de funções.

No próximo capítulo são propostos dois novos algoritmos combinando AG com propriedades dos códigos algébricos. O desempenho dos novos algoritmos é investigado com a sua aplicação na solução de problema difíceis [9].

Capítulo 3

Algoritmo Genético Guiado por Códigos Algébricos

3.1 Introdução

Dois fatores principais tornam complicada a otimização de algumas funções: irregularidade local (não-diferenciabilidade) resultante de grandes oscilações e a existência de diversos pontos extremos. Devido a suas complexidades, estas funções denominam-se "funções-armadilha". Apesar do algoritmo genético ter conseguido bom desempenho em diversos problemas, há funções em que seu desempenho deixa a desejar [9].

Usando argumentos da teoria da informação, Wolpert mostrou que qualquer algoritmo de busca tem em média o mesmo desempenho considerando o universo de todos os problemas de busca, assim é natural que haja problemas em que os AGs exibem baixo desempenho. O resultado é expresso no teorema abaixo.

Teorema 1 *Para dois algoritmos distintos a_1 e a_2, tem-se que:*

$$\sum_f P(d_m^y|f,m,a_1) = \sum_f P(d_m^y|f,m,a_2) \tag{3.1}$$

no qual $d_m^y(i)$ é uma amostra de possíveis soluções visitadas, y é a adequabilidade, f é a função objetivo, m é o número de iterações realizadas pelos algoritmos e

$P(d_m^y(i)|f,m,a)$ é desempenho de um dado algoritmo a, com m iterações realizadas sobre uma função objetivo f.

A prova deste teorema é encontrada em Wolpert [10]. Portanto, $P(d_m^y(i)|f,m,a)$ é independente de a quando é avaliado sobre todas as classes de funções. Este teorema afirma o quão perigoso é comparar algoritmos por seus desempenhos em uma amostra pequena de problemas. De fato é necessário ter um dicionário problema×algoritmo.

Para solucionar alguns destes problemas, neste capítulo, são apresentados dois novos algoritmos utilizando códigos algébricos:

- AG Guiado por Códigos Algébricos(AGGC) - Partindo do AG padrão descrito, um novo algoritmo é obtido usando as propriedades dos códigos algébricos. Este novo algoritmo obteve adequabilidades melhores mas com tempo de processamento superior apesar de usar um espaço de busca menor.

  Analisando o comportamento do AG na busca do ponto "ótimo"ou "sub-ótimo", pode-se perceber que por se constituir em um algoritmo de busca cego algumas vezes seu desempenho não é o desejável, ou seja, o AG converge a soluções sub-ótimas aquém do resultado esperado. Em consequência disto, utilizamos a teoria de códigos algébricos para ajudar a guiar o AG no espaço de busca. As relações que justificam a pesquisa foram apresentadas em Assis [3] e são mostradas na Seção 3.2.

- AG Guiado por Códigos Modificado(AGGCM) - Neste outro algoritmo, é introduzida a definição de classes laterais. Esse novo algoritmo mostra adequabilidades melhores que o AG e o AGGC e também mostra que existem regiões do espaço de busca nas quais desempenhos melhores são atingidos.

Na Seção 3.2 são apresentados alguns parâmetros do AG e Códigos Algébricos e suas relações. Na Seção 3.3 apresentamos o AGGC. Na Seção 3.4 o AGGCM é introduzido. Finalmente a Seção 3.5 conclui o capítulo.

3.2 Relações Algébricas entre o Algoritmo Genético e Códigos

Nesta seção são mostradas algumas relações entre AG e os Códigos Algébricos e assim é justificada a introdução dos códigos no AG. A definição dos códigos algébricos, bem como suas características e propriedades são apresentadas no Apêndice B.

3.2.1 Códigos Algébricos

Visualisam-se pequenas esferas que contenham cada uma das palavras-códigos de um código como centro, cada esfera com o mesmo raio. Aumenta-se o raio destas esferas até que elas não possam ser maiores sem causar interseção entre algumas esferas. Este raio é chamado de raio de empacotamento do código visto na Figura 3.1.

Definição 3 *Raio de empacotamento pac(C) é o maior número inteiro tal que exista uma palavra-código dentro de uma esfera de Hamming com uma palavra-código $c \in C$ como centro*

$$b(c, p(C)) \cap = \{c\} \tag{3.2}$$

Considere que o raio(Figura 3.1) continua a crescer até que todos os pontos em S^l, no qual $S = \{0, 1\}$ é o corpo finito dos inteiros mod 2, estejam contidos em pelo menos uma esfera. Este raio é chamado de raio de cobertura do código. A Figura 3.2 mostra graficamente o raio de cobertura.

Definição 4 *Raio de cobertura cov(C) é o menor número inteiro tal que cada l-upla binária tenha distância de Hamming de no máximo cov(C) à alguma palavra-código de C.*

Estas definições estão melhor detalhadas em Sloane [11], Blahut [12], Araújo [13].

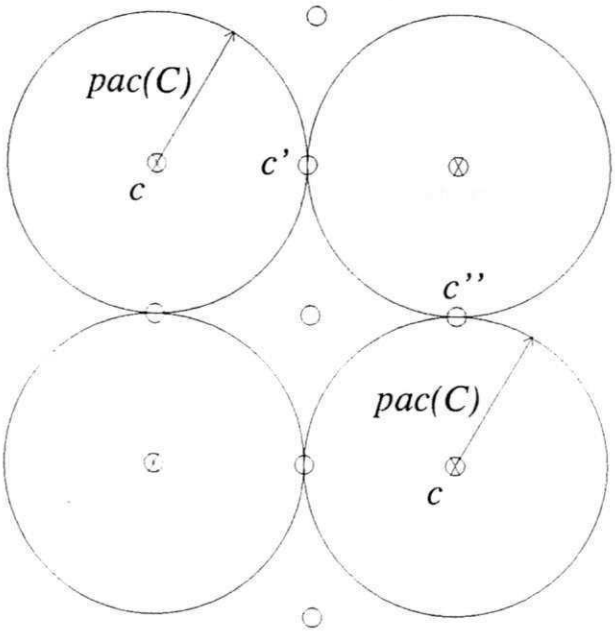

Figura 3.1: Raio de Empacotamento

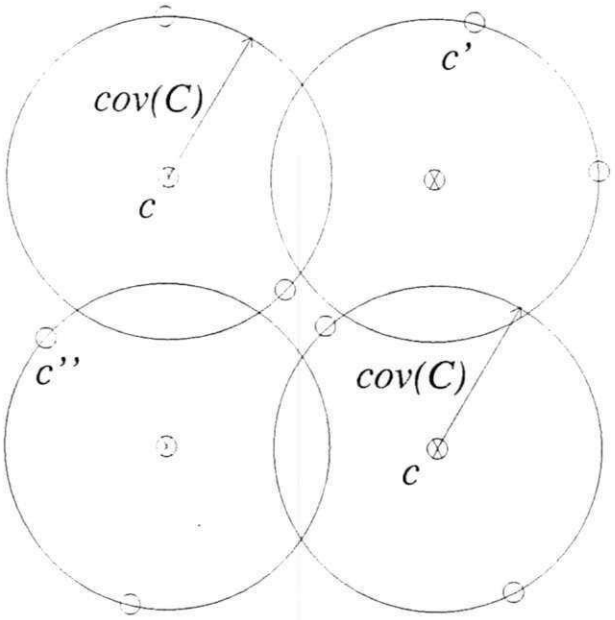

Figura 3.2: Raio de Cobertura

3.2.2 Algoritmo Genético

Nesta seção são vistas algumas definições para posteriormente relacinar o AG com os códigos algébricos.

Definição 5 *Seja $\mathcal{E}_{\mathcal{A}} \subseteq S^l$ o conjunto de seqüências calculadas durante a execução do algoritmo $\mathcal{A}$.*

Definição 6 *"Thoroughness" τ de um algoritmo $\mathcal{A}$ é o mínimo R tal que*

$$\forall x \in S^l, \exists y \in b(x, R) \cap \mathcal{E}_{\mathcal{A}} \tag{3.3}$$

com probabilidade maior que $\frac{1}{2}$

Esta medida revela algo sobre a verossimilhança de esconder um pico local de $\mathcal{A}$ usando uma codificação particular.

Definição 7 *"Sparsity" σ de um algoritmo $\mathcal{A}$ é o mínimo R tal que*

$$\forall x \in \mathcal{E}_{\mathcal{A}}, \exists y \neq x, y \in b(x, R) \cap \mathcal{E}_{\mathcal{A}} \tag{3.4}$$

com probabilidade maior que $\frac{1}{2}$

Esta medida revela o quão rapidamente $\mathcal{A}$ cobre as regiões nas quais sua busca é apresentada.

Para colocar as relações entre os códigos e o AG é necessário supor que as seqüências examinadas pelo algoritmo são palavras-código de um código binário $(l, k, d(C))$, ou seja, $\mathcal{E}_{\mathcal{A}}$ é um conjunto igual a C. De acordo com Assis [14], pode-se relacionar o AG e os códigos através das próximas inequações:

$$\tau \leq cov(C) \tag{3.5}$$

$$\sigma > 2pac(C) \tag{3.6}$$

3.3 AGGC

O que distingue AG de outro método de busca é a sua habilidade inerente de explorar amostras. Esta habilidade é conhecida como paralelismo implícito, ou seja, quantos esquemas sejam amostrados de acordo com as suas adequabilidades observadas por ações combinadas de seleção e recombinação.

Segundo Schaffer [15], este comportamento exploratório torna AG sensível a certas correlações entre esquemas que contribuem para o desempenho e esquemas que são parasitas, os quais se não combatidos, direcionam o AG à convergência prematura, ou seja a soluções sub-ótimas. Estas correlações são inerentes ao AG, não podendo ser eliminadas, mas podemos explorar maneiras de diminuir seu efeito.

Considerando isso desenvolveu-se um algoritmo que combina as propriedades do código, a "varredura" que o mesmo faz sobre a espaço de busca a partir da palavra de informação com a aleatoriedade e evolução genética do AG. Este novo algoritmo deve respeitar todas as propriedades dos códigos algébricos bem como a evolução genética e seus principais operadores.

Por se tratar também de um AG, este algoritmo com o uso dos operadores poderia quebrar a estrutura do código algébrico. É necessário então, trabalhar dentro da palavra de informação e só depois codificá-la usando a Eq. B.11 com a matriz geradora sendo pré-definida e determinada a partir da matriz de paridade.

A intenção da introdução dos códigos algébricos na estrutura do AG foi de resolver o problema das funções armadilha descrito anteriormente. A principal diferença deste algoritmo para o AG padrão é no cálculo da adequabilidade, no qual cada indivíduo a população deve ser multiplicado pela matriz geradora e assim determinar sua adequabilidade.

A definição do AGGC é apresentada como uma modificação de um algoritmo genético padrão. No algoritmo 1, $P(t)$ representa uma população com N cromossomos na geração t. A expansão (offspring) $P'(t)$ é gerada aplicando-se os operadores genéticos tradicionais de recombinação e mutação, bem como a reprodução. A avaliação dos elementos da expansão é feita calculando a adequabilidade, que depende do problema estudado.

Algoritmo	Espaço de Busca
AG	2^{31} indivíduos
AGGC	2^{26} indivíduos

Tabela 3.1: Comparação entre os Espaços de Busca

A modificação importante introduzida no algoritmo 1 que define o algoritmo guiado por código algébrico concerne ao tamanho do espaço de busca. No AG, a população $P(t)$ é um subconjunto de $\{0,\ 1\}^l$ enquanto que para AGGC esta população é extraída de $\{0,\ 1\}^k$, um conjunto de cardinalidade exponencialmente menor. Com isto, todas as operações genéticas e de reprodução passam a ser aplicadas a cadeias (strings) de $k < l$ símbolos (genes). Usando um código de Hamming $(31, 26)$ tem-se na Tabela 3.1 o espaço de busca que cada algoritmo possui.

Um único ponto onde cadeias de comprimento l são utilizadas é na avaliação pelo cálculo da adequabilidade. O pseudo-código do AGGC é mostrado no algoritmo 2, lembrando que a condição final é igual ao algoritmo 1, ou seja, número máximo de gerações. O símbolo $P(t)\,G$ no algoritmo 2 denota a codificação, ou seja a multiplicação de cada elemento da população pela matriz geradora do código (Eq. B.11).

3.4 AGGCM

Sabendo que o espaço de busca do AGGC é menor que o do AG padrão, e como pode ser aumentado, usa-se então o conceito de classe lateral descrita na Seção B.3.2. Foi construido um segundo algoritmo no qual houve a introdução das classes laterais utilizando a Eq. B.27. Antes de mais nada, sem perda das propriedades dos códigos, este algoritmo opera também dentro da palavra de informação, mas quando calcula as adequabilidades dos indivíduos da população, usa a matriz geradora e introduz as classes laterais.

Este algoritmo é baseado em duas modificações em relação ao original:

a) Fazer mudanças de classes laterais ao longo do espaço de busca no intuito de descobrir as melhores classes.

b) Aplicar o algoritmo nas melhores classes apontadas no passo anterior.

É imperativo salientar que no passo b, a população não irá aumentar, mas o AG irá operar com o código com população N e as melhores classes laterais, também com população N, simultaneamente, sem interferência entre elas. Este algoritmo tenta evidenciar empiricamente que o uso do código permite varrer o espaço de busca melhor que o AG padrão como sugere Assis [3]. Percebe-se inuitivamente que neste algoritmo compara-se o AGGC com o AGGCM, porque nele, como visto anteriormente, trabalha-se o código em paralelo com as melhores classes laterais.

O interesse de usar apenas as classes laterais consideradas melhores é de manter o espaço de busca do AGGCM menor que o do AG padrão, e assim avaliá-lo, visto que o tempo de processamento teoricamente é maior devido ao uso, no cálculo das adequabilidades dos indivíduos, da Eq. B.11 em todas as gerações.

A única diferença do AGGCM para o anterior é que este avalia $P'(t)\,G+v$, ou seja, apenas a classe lateral correspondente.

Algoritmo 2 AGGC

Inicialização

- $t \leftarrow 0$;
- *inicializar* $P(t) \subset \{0,\ 1\}^k$;
- *avaliar* $P(t)\,G$;

Iteração

Enquanto $Final = FALSO$ **faça**

- $P'(t) \leftarrow$ *variação* $P(t)$;
- *avaliar* $P'(t)\,G$;
- $P(t+1) \leftarrow$ *selecionar* $P(t)$;
- $t \leftarrow t+1$;

Fim

Fim

Algoritmo 3 AGGCM

Inicialização

- $t \leftarrow 0$;
- *inicializar* $P(t) \subset \{0,\ 1\}^k$;
- *avaliar* $P(t)\,G$;

Iteração

Enquanto $Final = FALSO$ **faça**

- $P'(t) \leftarrow$ *variação* $P(t)$;
- *avaliar* $P'(t)\,G+v$;
- $P(t+1) \leftarrow$ *selecionar* $P(t)$;
- $t \leftarrow t+1$;

Fim

Fim

3.5 Conclusão

Neste capítulo foram apresentados dois novos algoritmos que combinam AGs com a teoria dos códigos algébricos, bem como também foram mostradas algumas relações que justificam esta combinação.

Estes novos algoritmos foram criados visando à resolução dos problemas que o AG padrão apresentava em funções de difícil solução, ou funções do tipo "armadilha".

No capítulo seguinte é apresentada uma aplicação da engenharia de telecomunicações que serve de análise do desempenho do AG contra o AGGC. Na seção seguinte, o outro algoritmo AGGCM é testado em funções de difícil solução e comparado com o AG e o AGGC.

Capítulo 4

Análise de Resultados

4.1 Aplicações do AGGC

Nesta seção uma comparação entre os desempenhos dos algoritmos 1 e 2 é apresentada. Foi utilizado para a comparação um problema de engenharia de telecomunicações, cujos detalhes podem ser encontrados em Aquino [16] e Sousa [17].

4.1.1 Construção de Constelações Resistentes ao Desvanecimento

Considere um conjunto de vetores reais V, L-dimensionais, com $\mid V \mid$ vetores. Estes conjuntos denominam-se constelações e são utilizados na especificação de certos sistemas de transmissão por rádio digital. Nestes sistemas a figura de mérito, ou função objetivo, calculada pela Eq. 4.1 deve ser maximizada para obtenção de constelações mais eficientes para determinado tipo de transmissão [16].

$$F(V) = \min_{(x,\hat{x})\in V(x\neq\hat{x})} \prod_{i=1}^{L} (x_i - \hat{x}_i)^2 \tag{4.1}$$

no qual $x = \{x_1, x_2, \cdots, x_L\}$ e $\hat{x} = \{\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_L\}$ são vetores de v.

A maximização de $F(V)$ deve ser realizada no entanto utilizando apenas rotações generalizadas nos $\frac{L(L-1)}{2}$ possíveis planos do espaço L-dimensional [1].

[1] O método visa gerar sistemas de modulação robustos ao desvanecimento, que é um efeito altamente

Para $L = 2$, a forma inicial da constelação é

$$V = \begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \\ -1 & -1 \end{bmatrix} \tag{4.2}$$

Para obter as rotações, inicialmente representa-se os pontos de V como linhas de uma matriz $\mathcal{V}_{|V|\times L}$ e multiplica-se esta matriz por um produto de matrizes $L \times L$ ortogonais A dado por:

$$A = \left(\prod_{i \le i \le L-1,\, i+1 \le j \le L} G(i, j, \theta_{ij}) \right) \tag{4.3}$$

nesta equação, $G(i, j, \theta_{ij})$ são matrizes de Givens.

Define-se uma matriz de Givens por:

$$G = \begin{pmatrix} 1 & & & & & & \\ & \cdots & & & & & \\ & & c & \cdots & s & & \\ & & \vdots & & \vdots & & \\ & & -s & \cdots & c & & \\ & & & & & \cdots & \\ & & & & & & 1 \end{pmatrix} \tag{4.4}$$

Na Eq. 4.4, $c = \cos\theta_{ij}$, $s = \text{sen}\theta_{ij}$ são ângulos de rotação no plano definido pelos eixos i e j da base canônica do espaço L-dimensional. Na matriz de Givens os elementos G_{rr} da diagonal principal são todos iguais a 1, exceto quando $r = i$ ou $r = j$. Fora da diagonal principal os únicos elementos não-nulos são aqueles iguais a c ou $\pm s$.

O objetivo é determinar conjuntos V rotacionados (que maximizem $F(VA)$) variando os $\frac{L(L-1)}{2}$ ângulos de acordo com alguma discretização. É importante ressaltar que o tamanho do espaço de busca aumenta rapidamente. Assim, por exemplo, para variações de 1° nos ângulos, o tamanho do espaço de busca é $360^{\frac{L(L-1)}{2}}$ para um espaço com L dimensões!

prejudicial aos sistemas de comunicações móveis digitais.

A implementação em linguagem de alto nível "C"do AG e do AGGC foi realizada utilizando o código de Hamming binário de comprimento $l = 31$ e $k = 26$ com a matriz geradora mostrada na Eq. 4.5. Deve-se notar que todas as operações no AG utilizam cromossomos de comprimento 31, enquanto que para o AGGC todas as operações são implementadas usando cromossomos (índices) de comprimento 26, exceto durante o cálculo da adequabilidade, quando previamente se realiza o produto $i\mathbf{G}$, para cada um dos indivíduos da população.

$$\mathbf{G} = \begin{pmatrix}
1 & 0 & \cdots & & 0 & 1 & 1 & 1 & 1 & 1 \\
0 & & & & & 1 & 1 & 1 & 1 & 0 \\
 & & & & & 1 & 1 & 1 & 0 & 1 \\
 & & & & & 1 & 1 & 0 & 1 & 1 \\
 & & & & & 1 & 0 & 1 & 1 & 1 \\
 & & & & & 0 & 1 & 1 & 1 & 1 \\
 & & \ddots & & & & & & & \\
 & & & & & 1 & 1 & 1 & 0 & 0 \\
 & & & & & 1 & 1 & 0 & 1 & 0 \\
 & & & & & 1 & 0 & 1 & 1 & 0 \\
 & & & & & 0 & 1 & 1 & 1 & 0 \\
 & & & & & 1 & 1 & 0 & 0 & 1 \\
 & & & & & 1 & 0 & 1 & 0 & 1 \\
 & & & & & 0 & 1 & 1 & 0 & 1 \\
\vdots & & 1 & & \vdots & & & & & \\
 & & & & & 1 & 0 & 0 & 1 & 1 \\
 & & & & & 0 & 1 & 0 & 1 & 1 \\
 & & & & & 0 & 0 & 1 & 1 & 1 \\
 & & & & & 1 & 1 & 0 & 0 & 0 \\
 & & & & & 1 & 0 & 1 & 0 & 0 \\
 & & & \ddots & & & & & & \\
 & & & & & 0 & 1 & 1 & 0 & 0 \\
 & & & & & 1 & 0 & 0 & 1 & 0 \\
 & & & & & 0 & 1 & 0 & 1 & 0 \\
 & & & & & 0 & 0 & 1 & 1 & 0 \\
 & & & & & 1 & 0 & 0 & 0 & 1 \\
 & & & & & 0 & 1 & 0 & 0 & 1 \\
 & & & & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & & \cdots & 0 & 1 & 0 & 0 & 0 & 0 & 1
\end{pmatrix} \tag{4.5}$$

Os resultados são apresentados nas Tabelas 4.1 a 4.4 [1]. A segunda coluna de cada tabela mostra o valor da adequabilidade (figura de mérito) alcançada pelo algoritmo indicado na primeira coluna. O valor do ângulo final e o tempo (em segundos) dispendidos podem ser lidos nas terceira e quarta colunas, respectivamente. Para L=2, os resultados encontram-se na Tabela 4.1. Neste caso, os algoritmos determinam o mesmo valor de adequabilidade. Verifica-se que o AG consumiu cerca de 60% do tempo utilizado pelo AGGC. Isto pode ser explicado devido o AGGC ter que realizar o produto $i\mathbf{G}$ toda vez que é necessário calcular a figura de mérito, fazendo com que seja exigido um maior esforço computacional apesar do espaço de busca ser menor. Para

Método	$F(C)$	Ângulo (o)	Tempo(s)
AG	3.199	31.716	22.6
AGGC	3.199	31.716	39.7

Tabela 4.1: Comparação para L=2

Método	$F(C)$	Ângulo (o)	Tempo(s)
AG	0.74473	[19 48 25]	58
AGGC	0.7496	[25 72 20]	111.9

Tabela 4.2: Comparação para L=3

L=3, os resultados são mostrados na Tabela 4.2. O tamanho do espaço de busca é, neste caso, igual a $360^3 = 4.66 \times 10^7$. As adequabilidades obtidas foram praticamente iguais. Apesar do AGGC conseguir uma adequabilidade ligeiramente melhor, o AG foi novamente mais rápido. É importante notar que os resultados apresentados nas Tabelas 4.3 e 4.4 não mostram o valor do ângulo final encontrado, devido a uma melhor organização dos resultados, visto que a figura de mérito e o tempo são mais importantes para esta análise dos resultados. Entretanto agora o espaço de busca seja constituido de 2.18×10^{15} possíveis soluções. Para L=5 os resultados são vistos na Tabela 4.4. A adequabilidade obtida pelo AGGC foi praticamente 2 vezes melhor que a do AG, embora tenha consumido um tempo ainda maior pelos motivos já explicados. Verifica-se que, entretanto, ao passo que L aumenta, os tempos de processamento dos dois algoritmos se aproximam como visto na Tabela 4.4, portanto é fácil concluir que para um espaço de busca bem brande os tempos serão bem próximos.

Método	$F(C)$	Tempo(s)
AG	0.01056	91.5
AGGC	0.03751	133

Tabela 4.3: Comparação para L=4

Método	$F(C)$	Tempo(s)
AG	0.00058	173.1
AGGC	0.00122	208.9

Tabela 4.4: Comparação para L=5

4.2 Aplicações do AGGCM

Nesta seção uma comparação entre os desempenhos dos algoritmos 1, 2, 3 é apresentada. Foram utilizadas funções de difícil solução para o AG, cujos detalhes podem ser encontrados em Lin [9], Schwefel [18] e Fogel [19]. Mostra-se também os gráficos destas funções para o caso de $n = 2$ nas Figuras 4.1 a 4.5.

As funções que devem ser minimizadas são as seguintes:

- Modelo da Esfera

$$f_1 = \sum_{i=1}^{n} x_i^2, \tag{4.6}$$

$$\text{para } -100 \leq x_i \leq 100$$

$$min(f_1) = f_1(0, \cdots, 0) = 0$$

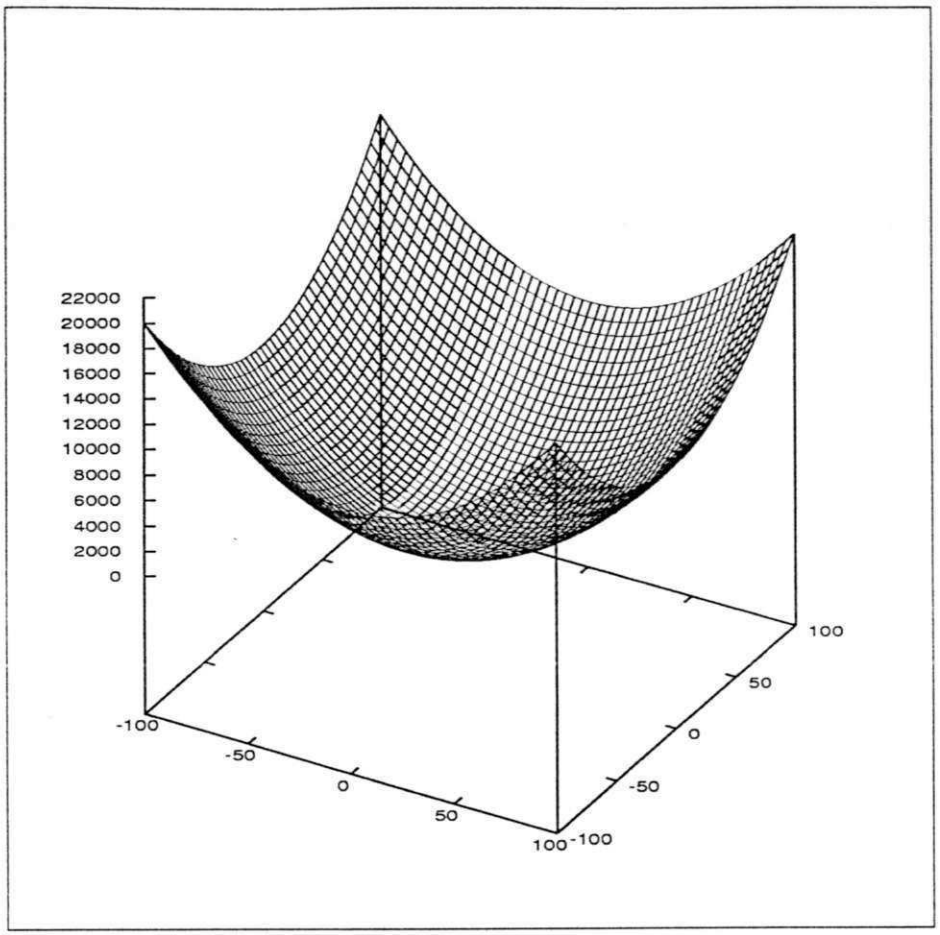

Figura 4.1: Versão bidimensional de f_1

- Problema de Schwefel

$$f_2 = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|, \tag{4.7}$$

$$\text{para } -10 \leq x_i \leq 10$$

$$min(f_2) = f_2(0, \cdots, 0) = 0$$

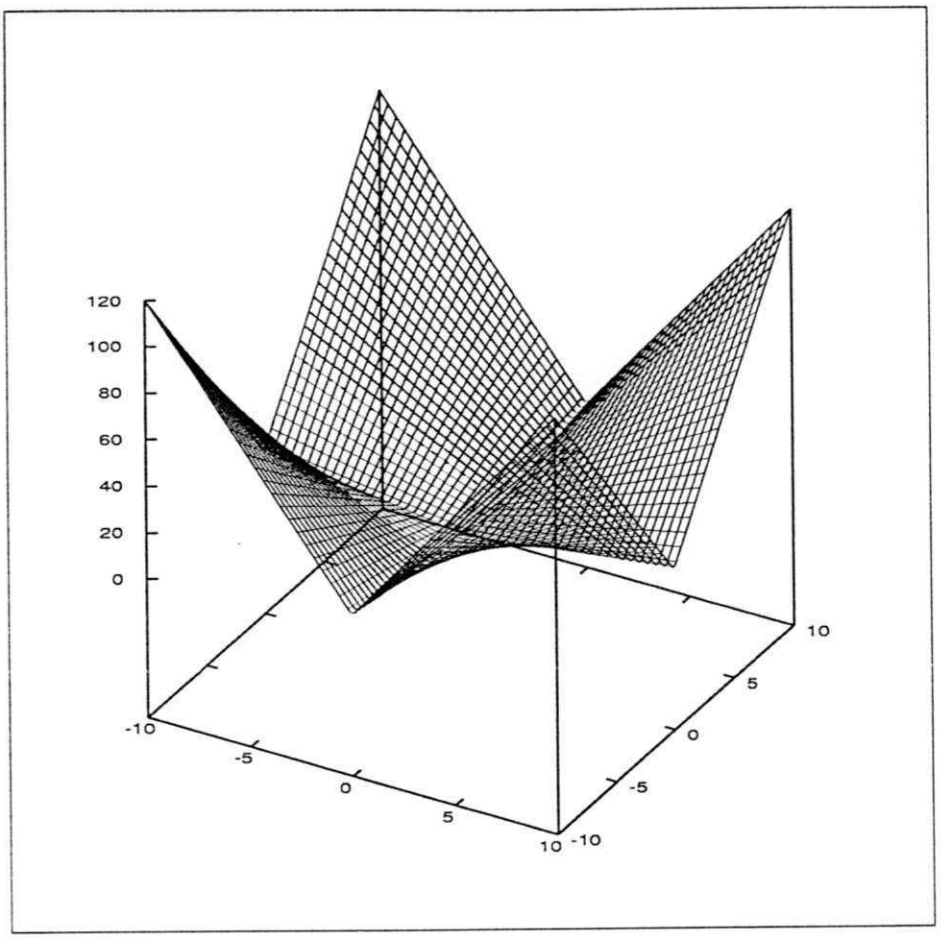

Figura 4.2: Versão bidimensional de f_2

- Problema 2 de Schwefel

$$f_3 = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2, \tag{4.8}$$

$$\text{para } -100 \leq x_i \leq 100$$

$$min(f_3) = f_3(0, \cdots, 0) = 0$$

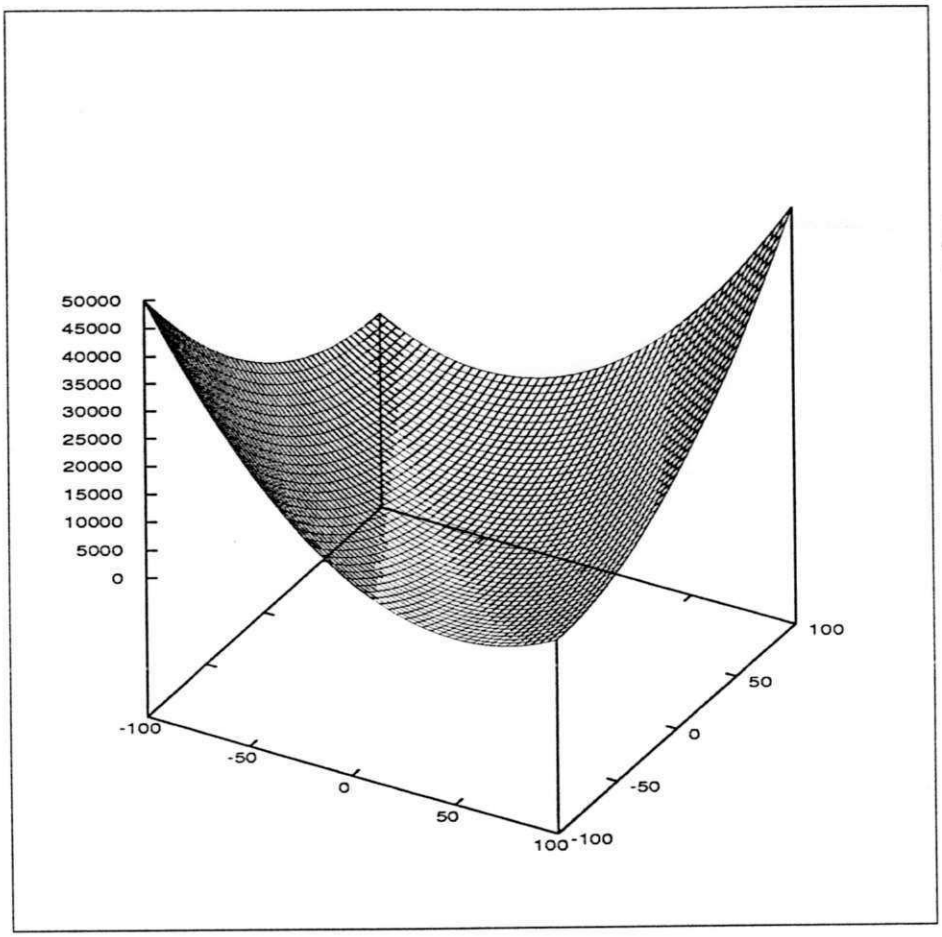

Figura 4.3: Versão bidimensional de f_3

- Problema Generalizado de Schwefel

$$f_4 = \sum_{i=1}^{n} - x_i \sin(\sqrt{|x_i|}), \tag{4.9}$$

$$\text{para } -500 \leq x_i \leq 500$$

$$min(f_4) = f_4(420.9687, \cdots, 420, 9687) = -(n \cdot 418.95)$$

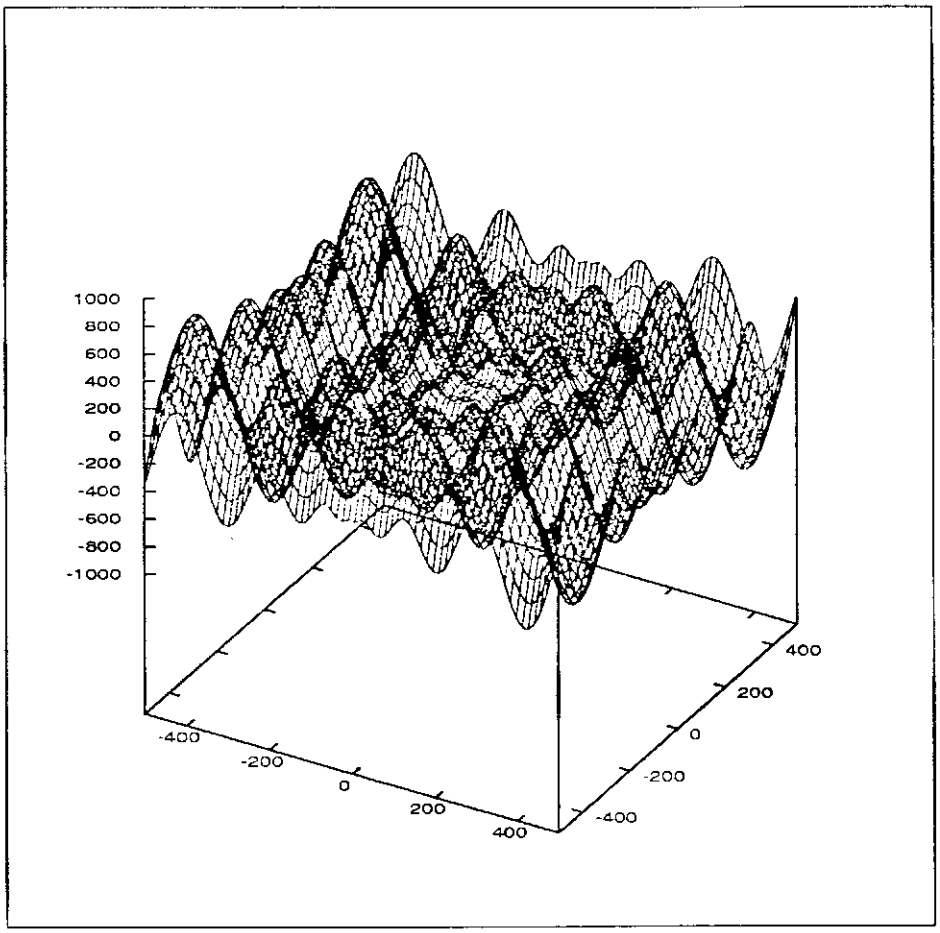

Figura 4.4: Versão bidimensional de f_4

- Problema Generalizado de Rastrigin

$$f_5 = \sum_{i=1}^{n} ({x_i}^2 - 10cos(2\pi x_i) + 10), \tag{4.10}$$

$$\text{para } -5.12 \leq x_i \leq 5.12$$

$$min(f_5) = f_5(0, \cdots, 0) = 0$$

Estas funções foram escolhidas pois se constituem de funções de difícil solução o que as tornam bastante difíceis para a análise do desempenho do AG. As funções foram

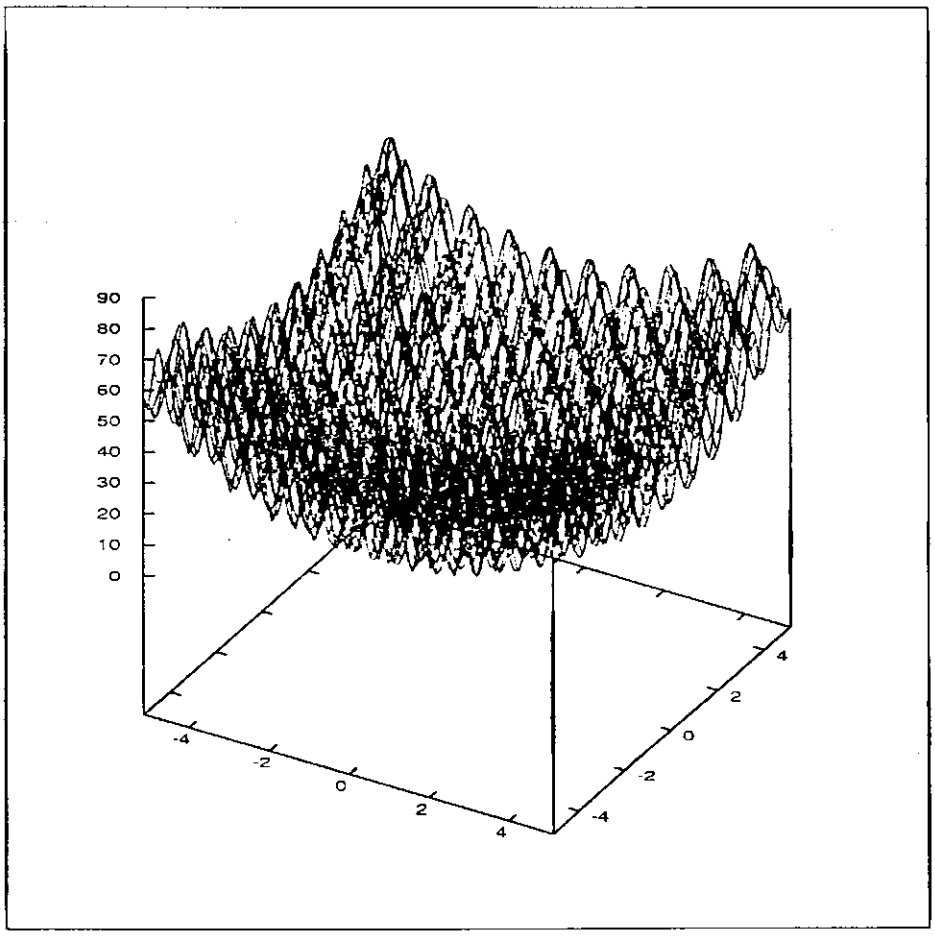

Figura 4.5: Versão bidimensional de f_5

minimizadas para n=30, por ser considerado um número compatível com os recursos computacionais disponíveis em nosso laboratório. A implementação em linguagem de alto nível "C"do AG e do AGGC foi realizada utilizando o código de Hamming binário de comprimento $l = 31$ e $k = 26$. Deve-se notar que todas as operações no AG utilizam cromossomos de comprimento 31, enquanto que para o AGGC todas as operações são implementadas usando cromossomos (índices) de comprimento 26, exceto durante o cálculo da adequabilidade, quando previamente se realiza o produto iG, para cada um dos indivíduos da população.

Os resultados são apresentados na Tabela 4.5 [2]. O desempenho médio dos respectivos algoritmos para cada função é mostrado nas colunas 2, 4 e 6.

Função	AG		AGGC		AGGCM	
	$\overline{f}$	σ	$\overline{f}$	σ	$\overline{f}$	σ
f_1	0.223	0.038	0.207	0.02	0.1086	0.0924
f_2	0.344	0.023	0.110	0.11	0.0932	0.0449
f_3	0.119	0.011	0.005	0.05	0.0017	0.004
f_4	-7877.52	292.32	-8136.21	2435.59	-5524.59	2332.65
f_5	112.02	41.88	89.43	36.02	68.9601	63.581

Tabela 4.5: Comparação entre os algoritmos

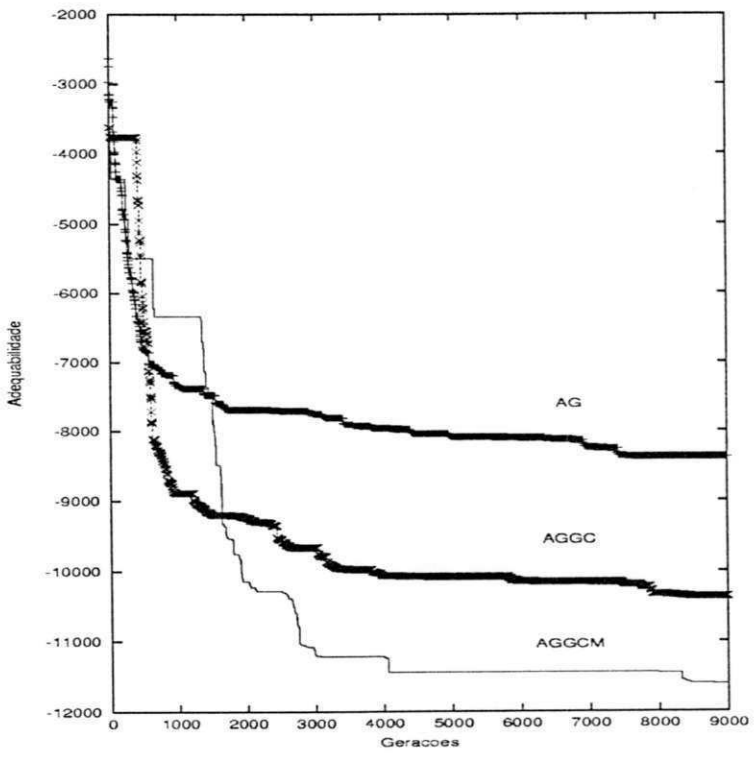

Figura 4.6: Adequabilidade X Número de Gerações em f_4

Foi necessário investigar quais classes laterais eram mais eficientes. Portanto foi feita uma análise geral em todas as classes laterais e assim determinadas as classes que apresentaram um desempenho desejável em relação à classe lateral 0, que representa o Código. Como neste caso se está trabalhando com um Código (31,26), existem $2^{l-k}-1$ classes laterais além do código. Com cada líder de classe tendo peso de Hamming igual a 1. Verificou-se que as classes laterais v_{28} e v_{29} eram mais eficientes, em que,

$$v = b_{31}b_{30}b_{29}b_{28}\cdots b_2b_1$$

e v_{28} significa que o bit 28 tem valor 1 e todos os outros com valor 0, o mesmo ocorre com v_{29}. Apesar destas classes obterem resulatados melhores não necessariamente terão os mesmos desempenhos em outra classe de função. Portanto, é feito um sorteio da classe lateral a ser avaliada e assim sucessivamente.

Pode-se perceber que em f_1, f_2 e f_3 os algoritmos apresentam resultados próximos. Isto se explica devido às funções serem unimodais, como pode ser visto nos gráficos bidimensionais apresentados nas Figuras 4.1 a 4.3. O mesmo não ocorre para f_4 e f_5 pois estas funções são multimodais e principalmente na Tabela 4.5 é mostrada a diferença entre os algoritmos, no qual mostra o desempenho melhor do AGGC. Vale salientar que o mau resultado apresentado pelo AGGCM é explicado devido às classes laterais que foram avaliadas apresentarem um péssimo resultado em relação ao código, mas o mesmo não ocorreu em f_5.

A Figura 4.6 apresenta a evolução do desempenho médio dos três algoritmos ao longo das gerações no caso da função f_4. Verifica-se que o AGGCM apresenta adequabilidades melhores ao longo das gerações do que os outros algoritmos.

4.3 Conclusão

Foi verificado que o desempenho do AG em funções de difícil solução não é o desejado. Com a introdução dos códigos algébricos no AG pode-se observar uma melhora significativa no seu desempenho.

Na aplicação de telecomunicações o AGGC apresentou adequabilidades melhores que o AG padrão. O mesmo ocorre com funções-armadilha cujo desempenho do

AGGCM superou ao do AG e AGGC em todas as funções analisadas. Esta melhora de desempenho, apesar de usar um espaço de busca menor, pode ser explicado porque com o código o AG percorre o espaço de busca mais ordenadamente e de forma mais eficiente que o AG padrão.

No capítulo seguinte são apresentadas as considerações finais e também mostradas algumas perspectivas futuras.

Capítulo 5

Conclusões e Perspectivas Futuras

Neste trabalho foram apresentados dois algoritmos que guiam o AG em funções de otimização. Estes algoritmos foram analisados sobre o ponto de vista da eficiência, tempo de processamento e regiões de atuação.

Devido aos problemas de otimização estarem intrinsicamente ligados aos valores das adequabilidades, foram avaliadas a maximização e a minimização destes problemas. No AGGC, o tempo de processamento foi superior ao AG mas em compensação obteve adequabilidades melhores. Isto pode mostrar que o código ajudou a guiar o AG sobre o espaço de busca, ou seja, mesmo com a aleatoriedade do AG, o código algébrico mostrou ter um espalhamento(o espaço de busca é melhor percorrido) sobre o espaço de busca melhor que o AG, trabalhando em regiões onde o AG padrão não tinha pesquisado.

No AGGCM, com o uso das classes laterais, o algoritmo mostrou ser mais eficiente que o AGGC e o AG mesmo usando um espaço de busca menor que o AG com um ligeiro aumento no esforço computacional. O AGGCM mostrou, empiricamente, que existem regiões do espaço de busca que obtiveram resultados mais significativos que o AG tradicional.

Supondo que a adequabilidade e o tamanho do espaço de busca sejam importantes na resolução de problemas de otimização, a introdução dos códigos nos AGs é justificada.

A seguir são feitas algumas sugestões para continuação das atividades de pesquisa:

- Utilizar outros códigos lineares e de preferência não-binários para uma melhor avaliação dos resultados encontrados.
- Inserir outros conceitos nos Algoritmo Genéticos como o de Cadeias de Markov que são um modelamento de sistemas que descrevem o sistema com um processo estocástico.Deste ponto de vista o sistema modelado é caracterizado pelos seus estados e a forma pela qual eles se alternam. Cadeias de Markov tem sido usadas para modelar características específicas do AGs [20] [21] [22] [23] [24] [25], tais como seleção, tendência genética, entre outros.
- Mostrar analiticamente a verdadeira contribuição dos Códigos Algébricos nos AG.
- Estabelecer critérios para avaliar analiticamente o desempenho dos Algoritmos Genéticos, ou evolucionários.

Apêndice A

Corpos Finitos

Uma variedade de códigos importantes baseados em estruturas de polinômios em anel e corpos finitos foram descobertas devido a busca por bons códigos para controle de erros ter confiado em larga escala em estruturas poderosas da álgebra moderna.

A.1 Grupos

Um conjunto é uma coleção arbitrária de objetos, ou elementos, sem nenhuma operação pré-definida entre eles. Um conjunto pode ser finito, infinito contável ou infinito incontável.

A característica primária de um conjunto é sua cardinalidade, que é definida como o número de objetos contidos no conjunto. A situação torna-se matematicamente complicada quando uma operação binária é imposta no conjunto, pois existem um conjunto de regras que restrigem os resultados da operação. O primeiro nível de complexidade encontrada quando uma operação binária é imposta no conjunto é o grupo.

Definição 8 *Um grupo é um conjunto de objetos G no qual uma operação binária "." é definida*

A operação binária toma 2 elementos quaisquer em G e gera como seu resultado um elemento que também está em G(fechamento). A operação deve satisfazer aos seguintes requerimentos se G é um grupo.

1. *Associatividade :* $(a \cdot b) \cdot c = a \cdot (b \cdot c), \forall a, b, c \in G$.

2. *Identidade: existe* $e \in G | a \cdot e = e \cdot a = a, \forall a \in G$.

3. *Inversa :* $\forall a \in G$ *existe um único elemento* $a^{-1} \in G | a \cdot a^{-1} = a^{-1} \cdot a = e$.

Por exemplo, o conjunto $\Re^*$ *(números reais sem o elemento* 0*) e a operação binária soma(ou multiplicação) formam um grupo. O conjunto de todos os número reais é fechado sobre a adição e multiplicação(quaisquer dois números reais somados ou multiplicados produzem um número real). Adição e Multiplicação são associativos no conjunto dos números reais. O elemento identidade da adição é* 0 *enquanto que na multiplicação é* 1.

Um grupo é dito ser comutativo se ele também satisfizer:

4. *Comutatividade :* $\forall a, b \in G, a \cdot b = b \cdot a$.

A operação do grupo para o grupo comutativo é usualmente representada usando o símbolo "+", e o grupo é algumas vezes dito ser "aditivo".

Um grupo é chamado de cíclico se existe um elemento no conjunto cuja potência constitui todods os elementos diferentes de zero do conjunto.

A ordem de um grupo é definida por ser a cardinalidade do grupo. Deveria ser anotado que a ordem de um grupo sozinho não é suficiente para especificar completamente o grupo ao menos que nós nos restrinjamos à uma operação particular.

Um dos métodos mais simples para construir grupos finitos encontra-se na aplicação da aritmética modular no conjunto de inteiros. Adição *modulo* m(ou $\mod m$) é frequentemente expresso da seguinte maneira:

$$a + b \equiv c \mod m \tag{A.1}$$

O resultado c é obtido somando a e b e usando adição de inteiros padrão e dividindo o resultado para $\mod m$; c é o resto positivo.

Exemplo 5 $13 + 18 \equiv 11 \mod 20$

$9 + 5 \equiv 4 \mod 10$

ou podemos representar como:

$$c = (a + b) \mod m \tag{A.2}$$

onde c é o resto positivo.

Exemplo 6 $11 = (13 + 18) \mod 20$
$4 = (9 + 5) \mod 10$

A.2 Anéis

Grupos foram obtidos aplicando uma simples operação binária e poucas condições restritivas para um conjunto de elementos. Se adicionarmos uma segunda operação binária ao primeiro, isto nos levaria ao domínio de anéis e corpos.

Definição 9 *Um anel é uma coleção de elementos A com duas operações binárias "+"e "·"embasado nas três seguintes propriedades*

1. A forma um grupo comutativo abaixo +. O elemento identidade aditivo é nomeado "0".

2. A operação · é associativa: $(a \cdot b) \cdot c = a \cdot (b \cdot c), \forall a, b, c \in A$

3. A operação · distribui sobre +: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

Um anel é dito ser um anel comutativo se

4. A operação · comuta (i.e., $a \cdot b = b \cdot a$*).*

Um anel é dito ser um anel com indentidade se

5. A operação · tem um elemento identidade, o qual é nomeado "1".

E se um anel satisfizer ambas as propriedades 4 e 5, ele é dito ser um anel comutativo com identidade.

A.3 Corpos

Definição 10 *Seja F uma conjunto de objetos no qual duas operações + e · são definidas. F é dito ser um corpo se e somente se*

1. F forma um grupo comutativo fechado sobre +. O elemento identidade aditivo é nomeado "0".

2. F - 0(o conjunto F com o elemento identidade removido) forma um grupo comutativo fechado sobre ·. O elemento identidade multiplicativo é nomeado "1".

3. As operações + e · distribuem: $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

Por exemplo, o conjunto $\{0, 1, 2, 3, 4, 5, 6\}$ adição e multiplicação módulo 7 forma um corpo, pois:

1) O conjunto é fechado sobre adição módulo 7

$$3 + 4 = 7 \mod 7 = 0$$
$$2 + 6 = 8 \mod 7 = 1$$

Note que na adição módulo X, $a+(b+c)$ é sempre igual a $(a+b)+c$ e $a+b = b+a$.

2) O conjunto de todos elementos difierentes de zero é fechado sobre a multiplicação mod7

$$3 * 4 = 12 \mod 7 = 5$$
$$2 * 6 = 30 \mod 7 = 2$$

Note que na multiplicação mod 7 $a*(b*c)$ é sempre igual a $(a*b)*c$ e $a*b = b*a$.

3) Finalmente, a propriedade distributiva

$$a * (b + c) = a * b + a * c \tag{A.3}$$

por exemplo

$$3 * (4 + 5) = 3 * 9 = 27 \mod 7 = 6$$

e

$$3 * 4 + 3 * 5 = 12 + 15 = 27 \mod 7 = 6$$

Um corpo pade também ser definido como um anel comutativo com identidade no qual todo elemento tem um inverso multiplicativo.

Adição *mod* 2	Multiplicação *mod* 2
$0+0=0$	$0*0=0$
$0+1=1$	$0*1=0$
$1+0=1$	$1+0=1$
$1+1=0$	$1*1=1$

Tabela A.1: Operações *mod* 2

A.4 Corpos Finitos

Como ferramenta matemática na construção de códigos e de sistemas de criptografia, foi desenvolvido por Évariste Galois um sistema algébrico chamado de corpo finito, ou campo de Galois(Galois Field). Corpos Finitos são denotados por $GF(q)$, onde q é algum número primo positivo e o conjunto $S = \{0, 1, 2, \cdots, q-1\}$ e as operações binárias $+$ e $*$ são adição e multiplicação módulo q respectivamente. Como na comunicação digital são utilizados bits, estamos interessados na subclasse de corpos finitos de q tal que q seja potência de 2. Isso é considerado principalmente porque em $GF(2)$ a representação em corpo finito é mapeada convenientemente no domínio digital.

Utilizando a notação $GF(q)$, a suclasse será denotada como $GF(2^n)$. Portanto $GF(2), GF(4), GF(8)$ e etc são corpos finitos $GF(2^n)$.

O campo de Galois $GF(q)$, em que q é primo, é chamado de corpos primos. Qualquer corpo no qual q seja primo pode ter adição e multiplicação módulo q como suas duas operações binárias. O campo de Galois, $GF(q^n)$, é chamado de extensão de corpo com $GF(q)$ sendo o corpo de base $GF(q^n)$.

O campo de Galois $GF(2)$ é chamado de corpo binário. Por exemplo, $GF(2)$ contém o conjunto $\{0, 1\}$ e utiliza adição e multiplicação módulo 2.

Note que a adição módulo 2 é equivalente a operação XOR e a multiplicação módulo 2 é equivalente a porta AND.

O polinômio em $GF(2)$ pode ser representado como coeficientes de $GF(2)$

$$a_0 + a_1 x^1 + a_2 x^2 + \cdots + a_n x^n \tag{A.4}$$

em que $a_i = 0$ ou 1 para $0 \leq i \leq n$.

Um polinômio é de grau n se não existe nenhum termo x no polinômio com petência maior que n.

Propriedades

Corpos Finitos podem ser obtidos a partir de anéis polinomiais. Seja $F[x]$ um anel polinomial sobre o corpo F, escolhendo qualquer polinômio $p(x)$ pertencente a $F[x]$.

Teorema 2 *Para qualquer polinômio mônico em F, o anel dos polinômios módulo $p(x)$ é o conjunto de todos os polinômios com grau menor que $p(x)$ junto com adição e multiplicação polinomial módulo $p(x)$. Se o polinômio $p(x)$ for primo então o anel de polinômios é um corpo.*

Dessa forma encontrando um polinômio primo $p(x)$ de grau n em $GF(q)$, pode-se construir um corpo de Galois com q^n elementos.

Antes de introduzir $GF(2^n)$, algumas definições são necessárias. Um polinômio $p(x)$ de grau n sobre $GF(2^n)$ é um polinômio da forma:

$$p(x) = p_0 + p_1 x + p_2 x^2 + \cdots + p_n x^n \tag{A.5}$$

em que os coeficientes p_i são elementos de $GF(2) = \{0, 1\}$. Polinômios em $GF(2)$ podem ser adicionados, subtraidos, multiplicados e divididos na forma usual.

Uma propriedade útil dos polinômios em $GF(2^n)$ é que"

$$p(x) = (p_0 + p_1 x + \cdots + p_n x^n)^2 = p_0 + p_1 x^2 + \cdots + p_n x^{2n} = p(x^2) \tag{A.6}$$

A noção de polinômio irredutível é agora introduzida.

Definição 11 *Um polinômio $p(x)$ sobre $GF(2)$ de grau n é irredutível se $p(x)$ não for divisível por nenhum polinômio sobre $GF(2)$ de grau menor que n e maior que zero.*

Para gerar a extensão do corpo $GF(2^n)$, um polinômio $p(x)$ mônico e irredutível de grau n em $GF(2)$ é escolhido. O conjunto de 2^n polinômios de grau menor que n em $GF(2)$ é formado e chamado de F. Pode ser provado que, quando a adição e

Notação Polinomial	Notação Binária	Notação Inteira	Notação Exponencial
0	00	0	0
1	01	1	x^0
x	10	2	x^1
$x+1$	11	3	x^2

Tabela A.2: Notações

multiplicação desses polinômios são tomados módulo $p(x)$, o conjunto F forma um corpo com 2^n elementos, denotado por $GF(2^n)$. Note que $GF(2^n)$ é uma extensão de $GF(2)$ de modo análogo ao modo que os números complexos são formados a partir dos número reais, nesse caso $p(x) = xx^2 + x + 1$. Os elementos do corpo são representados pelo conjunto $\{0, 1, x, 1+x\}$. Na Tabela A.2 são mostrados diversos tipos de notações para esses elementos.

Se $p(x)$ é um polinômio primitivo de grau n sobre $GF(2)$ então ele divide $x^m + 1$, em que $m = 2^n - 1$. A Tabela A.3 mostra uma lista de polinômios primitivos sobre $GF(2)$ com número mínimo de termos.

Teorema 3 *Para cada elemento não nulo $\alpha \in GF(q), ord(\alpha)$ divide $q-1$.*

Podemos facilmente observar isso pelo algoritmo de Euclides para divisão, em que $t = ord(\alpha)$

$$\begin{aligned} q - 1 &= \alpha t + r \\ \alpha^{q-1} &= \alpha^{\alpha t + r} \\ \alpha^{q-1} &= \alpha^{\alpha t}\alpha^r \end{aligned} \tag{A.7}$$

Como $\alpha^t = 1$ e $\alpha^{q-1} = 1$ então:

$$1 = 1 \cdot \alpha^r \tag{A.8}$$

Assim, $\alpha^r = 1$, ou seja, $r = 0$.

Teorema 4 *O grupo de elementos não nulos diferentes de $GF(q)$ é um grupo cíclico sobre a multiplicação.*

Grau n	Polinômio Primitivo
3	$1+x+x^3$
4	$1+x+x^2$
5	$1+x^2+x^5$
6	$1+x+x^6$
7	$1+x+x^7$
8	$1+x^2+x^3+x^4+x^8$
9	$1+x^4+x^9$
10	$1+x^3+x^{10}$
11	$1+x^2+x^{11}$
12	$1+x+x^4+x^6+x^{12}$
13	$1+x+x^3+x^4+x^{13}$
14	$1+x+x^6+x^{10}+x^{14}$
15	$1+x+x^{15}$
16	$1+x+x^3+x^{12}+x^{16}$

Tabela A.3: Polinômios Primitivos

Seja $GF(q)$ um corpo e $GF(Q)$ sua extensão ($q^m = Q$). seja β elemento de $GF(Q)$. O polinômio f de menor grau tal que $f(\beta) = 0$ é chamado de *polinômio mínimo de* β. Esse polinômio sempre existe, é único e irredutível. O polinômio mínimo é utilizado para gerar palavras códigos, pois a palavra $P(x)$ só é código se e somente se $P(\beta_1) = 0$.

Para $\alpha \in GF(q)$, seja t o menor inteiro tal que $\alpha^{p^t} = \alpha$. Define-se *conjugado de* α

$$C = \{\alpha, \alpha^{p^1}, \alpha^{p^2}, \alpha^{p^3}, \cdots, \alpha^{p^t}\}$$

Para representar os 2^n elementos o conceito de base é introduzido. A escolha de como representar elementos de corpo sobre bases ou potências de elementos primitivos usualmente depende da implementação em hardware ou software que seja adotada.

A.5 Bases

Um conjunto de n elementos linearmente independentes $x = \{x_0, x_1, x_2, \cdots, x_{n-1}\}$ em $GF(2^n)$ é chamado de base de $GF(2^n)$. Dessa forma, um elemento $x \in GF(2^n)$ pode ser representado unicamente como uma soma ponderada dessa base sobre $GF(2)$(para um mapeamento no corpo binário)

$$a = a_0x_0 + a_1x_1 + \cdots + a_{n-1}x_{n-1} \tag{A.9}$$

onde $a_0, \cdots, a_{n-1} \in GF(2)$.

A.5.1 Base Polinomial

Seja $p(x)$ um polinômio irredutível em $GF(2)$. Tomando α como raiz de $p(x)$, então $A = \{1, \alpha, \cdots, \alpha^{n-1}\}$ é uma base polinomial de $GF(2^n)$.

A.5.2 Base Dual

Sejam $\{\lambda_i\}$ e $\{\mu_i\}$ bases de $GF(2^n)$ e seja f uma função linear tal que $GF(2^n) \rightarrow GF(2)$ e $\beta \in GF(2^n)$. Então $\{\lambda_i\}$ e $\{\mu_i\}$ são duais com respeito a f e β:

$$f(\beta\lambda_i\mu_j) = \begin{cases} 1 \; se \; i = j \\ 0 \; se \; i \neq j \end{cases} \tag{A.10}$$

Nesse caso, $\{\lambda_i\}$ é a base padrão e $\{\mu_i\}$ é a base dual.

A.5.3 Base Normal

Uma base normal é um base da forma $B = \{\beta, \beta^2, \cdots, \beta^{2^{m-1}}\}$, em que $\beta \in GF(2^n)$. Para cada corpo existe no mínimo uma base normal. Ela é bastante útil quando a situação exige potenciação, pois se $a = (a_0, a_1, \cdots, a_{n-1})$ é a representação na base normal, então $a^2 = (a_{-1}, a_0, \cdots, a_{n-2})$.

Apêndice B

Códigos Algébricos

B.1 Introdução

Um sistema de comunicações conecta uma fonte de dados ao destino através de um canal. Conexões de microondas, cabos coaxiais, circuitos telefônicos e até fitas magnéticas são exemplos de canais de comunicação. Um projeto do Sistema de Comunicações desenvolve dispositivos que preparam a entrada(fonte) e a saída do canal. Para caracterizar um sistema de comunicações o diagrama de blocos da Figura B.1 foi desenvolvido [26] [27] [28]. Os dados que entram no sistema de comunicações da fonte de dados é processado em um codificador de fonte para representá-los de forma mais compacta. Então os dados são processados no codificador de canal, que é uma seqüência nova e mais longa chamada de palavra-código do canal, que tem mais redundância que a palavra-código da fonte. Cada símbolo é representado por 1 bit ou um grupo de bits. Depois o modulador converte cada símbolo da palavra-código do canal em um símbolo correspondente analógico. esta seqüência é transmitida pelo canal, que está sujeito a todos os tipos de ruído, distorção e interferência e em conseqüência disto a saída do canal difere da entrada. O demodulador converte cada sinal recebido da saída do canal em uma seqüência dentro do alfabeto dos símbolos das palavras-código do canal. Cada símbolo demodulado é uma estimativa melhor do símbolo transmitido, mas o demodulador faz algunbs erros por causa do ruído do canal. A seqüência demodulada de símbolos é chamada de palavra-código recebida. Por causa deste erros, os símbolos

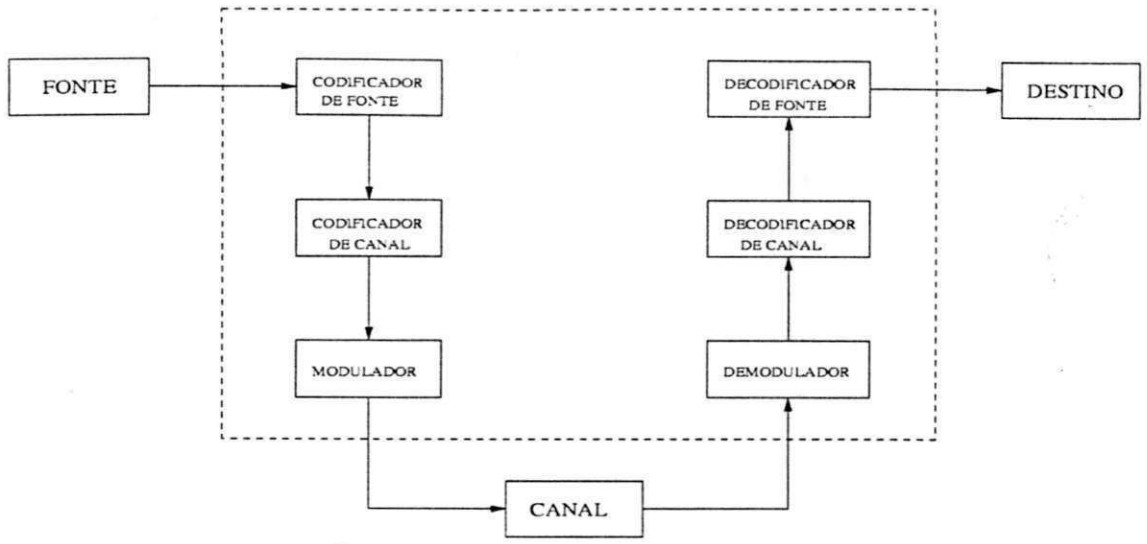

Figura B.1: Sistema de Comunicações

da palavra-código recebida não combinam com aqueles da palavra-código do canal.

O decodificador de canal usa a redundância na palavra-código do canal para corrigir os erros na palavra-código recebida e então produzir uma estimativa da palavra-código da fonte. Se todos os erros são corrigidos, a palavra-código da fonte estimada combina com a original. O decodificador da fonte desempenha a operação inversa do codificador de fonte e entrega sua saída para o usuário. Isto retrata uma codificação para controle de erros.

Shannon em 1948 [29] publicou um artigo intitulado *A Mathematical Theory of Communication* no qual definiu as bases dos atuais sistemas de comunicações digitais. As funções de entropia, discriminação e informação mútua definidas nesse artigo, bem como os importantes conceitos de capacidade de canal e taxa de distorção de uma fonte levam a definição de dois tipos distintos de códigos.

- **Códigos de fonte:** Os códigos de fonte são utilizados para remover as redundâncias que ocorrem naturalmente em qualquer fonte de informação, e assim possibilitam representar a informação de forma mais compacta.

- **Códigos de canal:** Os códigos de canal são utilizados para aumentar a imunidade das informações ao ruído. Isto é realizado através da introdução de uma

redundância controlada nos dados que serão transmitidos através do canal, fornecendo ao receptor a capacidade de detectar e possivelmente corrigir os erros causados pelo ruído do canal de comunicação.

Shannon também mostrou que esta codificação associada com qualquer canal de comunicação é um número C(medido em bits por segundo), chamado de capacidade do canal. Toda vez que a taxa de transmissão R(em bits/s) requerida de um sistema de comunicações é menor que C, é possível projetar para o canal um sistema de comunicações usando codificação para controle de erro, do qual a probabilidade de erro da saída é tão pequena quanto desejada. A teoria da informação de Shannon nos disse que é desperdício construir um canal que seja perfeito; é melhor e mais econômico fazer o uso de um código, mas não nos indicou códigos apropriados, apenas provou que eles existem.

Neste apêndice é apresentada a teoria básica de códigos corretores de erros. Na Seção B.2 é apresentada a teoria de códigos de blocos, bem como definições de alguns parâmetros dos códigos. Na Seção B.3 é introduzida uma classe dos códigos de bloco que são os códigos de bloco lineares e ao mesmo tempo mostra-se um exemplo desta classe que são os códigos de Hamming [26] [30].

B.2 Códigos de Blocos para Controle de Erros

Procura-se igualmente estabelecer uma equivalência entre termos de uso comum, daqueles que lidam com algoritmos genéticos e aqueles do jargão dos que trabalham mais diretamente com os códigos. Por exemplo, no caso binário, o termo *palavra código* refere-se a um vetor com l componentes (bits). Cada componente representa um *símbolo* a ser transmitido por um sistema de comunicações. Já no contexto deste trabalho, uma palavra código é interpretada como sendo um cromossomo e um símbolo equivale a um gene.

Um Código de Bloco $\mathcal{C}$ caracteriza-se em um conjunto de M palavras-código $\{c_0, c_1, \cdots, c_{M-1}\}$, no qual cada palavra-código é a da forma $c = (c_0, c_1, \cdots, c_{l-1})$; se as coordenadas individuais assumirem valores do corpo finito $GF(q)$(Apêndice A), então o código $\mathcal{C}$ é dito ser q-ário. Vale salientar que o comprimento desta palavras-códigos podem ser variáveis.

O processo de codificação nessa classe de códigos consiste em segmentar a seqüência de dados em blocos e mapear esses blocos nas palavras-código em **C**. Esse mapeamento é geralmente um-a-um, de forma que o processo de codificação possa ser revertido no receptor, recuperando o bloco de dados original.

Se os símbolos na seqüência de dados assumem valores em GF(q), então o conjunto de todas as possíveis k-úplas $\mathbf{m} = (m_0, m_1, m_2, ..., m_{k-1})$ formam um espaço vetorial sobre GF(q) contendo q^k vetores. Da mesma forma, o conjunto de todas as possíveis l-úplas sobre GF(q) forma um espaço vetorial sobre GF(q) contendo q^l vetores. Sendo assim, existem blocos com l símbolos que não estão associados a qualquer bloco de dados e portanto, não são palavras-código. O código **C** contém redundância se o número de palavras-código for menor que o número de possíveis blocos com n símbolos. A redundância r é expressa por:

$$r = l - \log_q M \tag{B.1}$$

Se $M = q^k$, então r é calculado pela diferença $r = (n - k)$ entre o comprimento da palavra-código e o comprimento do bloco de dados. A redundância é também freqüentemente expressa em termos da **taxa do código**.

Exemplo 7 (Código de Paridade Simples) *é uma classe de códigos binários bem conhecida, dada uma palavra-fonte de k bits, a palavra-código se construída adicionando um $(k+1)$-ésimo bit tal que o número total de 1's na palavra-código seja par. Portanto, por exemplo, com k=3 na Tabela B.1 ,*

Este é um código $(l = k+1, k)$. As palavras-fonte são selecionadas tal que qualquer mensagem possa ser convertida em uma sequüência de blocos sem criar ambigüidade ao decodificador da saída.

Tabela B.1: Código de Paridade Simples

Palavra-fonte		Palavra-código
000	$\leftrightarrow$	0000
001	$\leftrightarrow$	0011
010	$\leftrightarrow$	0101
011	$\leftrightarrow$	0110
100	$\leftrightarrow$	1001
101	$\leftrightarrow$	1010
110	$\leftrightarrow$	1100
111	$\leftrightarrow$	1111

Definição 12 A taxa de um código

Seja M o número de palavras-código, com comprimento l, em um código C. A taxa de C é

$$R = \frac{log_q M}{l} \tag{B.2}$$

A taxa do código se simplifica a $R = \frac{k}{l}$ para os casos no qual $M = q^k$, como no Exemplo 7.

No canal de comunicação onde é inserida a palavra-código, normalmente é um ruído aditivo como na Figura B.2. A quantidade de ruído no canal determina a verossimilhança que cada um dos q^l possíveis erros padrões e ocorrem. Este modelo é particularmente útil com canais binários, nos quais a adição destes erros padrões é desempenhado em *modulo*2. A determinação de se erros estão presentes na palavra recebida é chamada de detecção de erros e extremamente necessário também na correção de erros que vai depender dos erros que foram inseridos.

Durante a transmissão, o canal produz distorções nas palavras-código. Essas distorções podem ser modeladas por um processo aditivo, conforme ilustrado na Figura B.2.

Na recepção, geralmente o decodificador de canal tem como primeira tarefa a detecção de erros. Para isso, o decodificador analisa se a palavra recebida é uma palavra-código. Se a palavra recebida não for uma palavra-código, o decodificador assume que

um ou mais erros ocorreram na transmissão.

Nem todos os padrões de erros podem ser detectados no decodificador. Para isso, basta que os erros na transmissão alterem a palavra-código transmitida para uma outra palavra-código válida. Dado que uma palavra-código $\mathbf{c}$ foi transmitida, existem $(M-1)$ palavras-código diferentes de $\mathbf{c}$ que podem chegar no receptor e, portanto, existem $(M-1)$ padrões de erros não detectáveis em qualquer código $\mathbf{C}$.

O decodificador pode reagir a um erro detectado com uma das seguintes ações:

- Solicitar uma retransmissão da palavra-código;
- Sinalizar a palavra como sendo incorreta;
- Tentar corrigir os erros na palavra recebida.

A primeira estratégia listada acima é denominada **pedido de repetição automática (ARQ)**, sendo muito utilizada em sistemas de transferência de dados que prezam por uma alta confiabilidade nas informações recebidas.

A segunda opção é usualmente referenciada como *muting*. Ela é típica de aplicações que não disponibilizam de tempo suficiente para permitir a retransmissão da informação, e que consideram melhor atribuir um valor mudo a palavra recebida que tentar corrigir os erros da transmissão.

A última opção é referenciada como **correção direta de erros (FEC)**. Essa estratégia utiliza estruturas algébricas e aritméticas para determinar a palavra mais provável de ter sido enviada dentre as palavras-código válidas.

É possível que um decodificador **FEC** encontre para um determinado padrão de erros, uma palavra-código diferente daquela que foi realmente transmitida. Neste caso, é dito que o decodificador cometeu um **erro de decodificação**. Apesar do padrão de erros, que causa o erro de decodificação, poder ser detectado, não é possível detectar o próprio erro de decodificação. Usualmente, o número de padrões de erros que causam erros de decodificação é maior que o número de padrões de erros que não são detectáveis. Na verdade, se o código é escolhido cuidadosamente, os padrões de erros mais prováveis de ocorrer irão alterar as palavras-código transmitidas de tal forma que as palavras recebidas sejam consideradas não válidas.

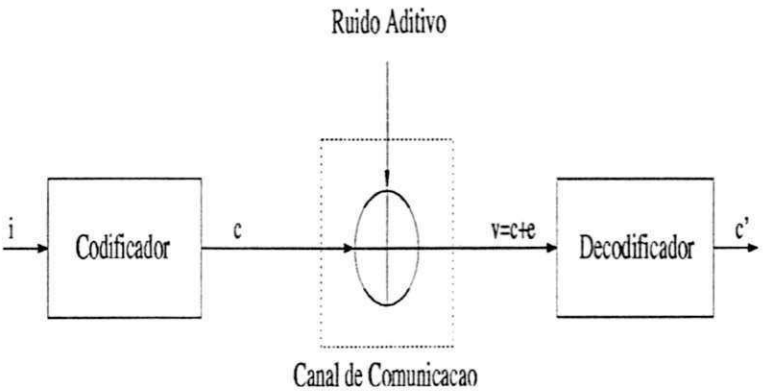

Figura B.2: Canal com Ruído Aditivo

Definição 13 *O peso de um código $w(c)$ é o número de coordenadas não-nulas na palavra código.*

Exemplo 8 Pesos

$w(1,0,0,1,1,1,1,0,0) = 5$

$w(\alpha^1, \alpha^{12}, 1, \alpha^3, \alpha^2, 0, 0, 1) = 6$

Considere os blocos $v = (\nu_0, \nu_1, \cdots, \nu_{n-1})$ e $w = (\omega_0, \omega_1, \cdots, \omega_{n-1})$.

A "distância" $d(v, w)$ entre esses dois blocos pode ser definida pela **distância de Hamming**.

Diferentemente da geometria Euclideana, no qual podemos determinar a distância Euclideana, nos códigos, usamos freqüentemente a distância de Hamming.

Definição 14 *Distância de Hamming entre os blocos v e w é o número de coordenadas nas quais os dois blocos se diferem.*

$$d(v, w) = |\{i | v_i \neq w_i, 0, 1, \cdots, l-1\}| \tag{B.3}$$

no qual $v, y \in S^l$.

Note que (S^l, d) é um espaço métrico com diâmetro l, em que S^l é o espaço de busca a ser percorrido. A distância de Hamming permite uma caracterização útil das capacidades de detecção de erros e correção de erros de um código de bloco como uma função da distância mínima do código.

A distância mínima de Hamming entre qualquer duas palavras-código é definida como a mínima distância do código, que é

Definição 15

$$d = d(c) = min_{x,y \in C}\{d(x,y)\} \tag{B.4}$$

Recorde que os únicos padrões de erros que não podem ser detectados são aqueles que transformam a palavra-código transmitida em uma outra palavra-código válida. Considere um código com distância mínima $d_{\min}$. Cada palavra transmitida desse código difere em no mínimo $d_{\min}$ coordenadas de qualquer outra palavra do código. Sendo assim, para que um padrão de erros não possa ser detectado, devem ocorrer no mínimo $d_{\min}$ erros nos símbolos da palavra-código transmitida.

Um código com distância mínima $d_{\min}$ pode detectar todos os padrões de erros com pesos menores ou iguais a $(d_{\min} - 1)$.

Note porém que um determinado código pode detectar também padrões de erros com pesos $\mathbb{W} \geq d_{\min}$. A afirmativa acima fornece simplesmente um limite sobre o peso para o qual pode-se detectar *todos* os padrões de erros.

Os sistemas **FEC** têm por meta a minimização da probabilidade de erros de decodificação. Seja $p(\mathbf{r}|\mathbf{c})$ a probabilidade do decodificador receber a palavra $\mathbf{r}$ sabendo que a palavra-código transmitida foi $\mathbf{c}$. Através da identificação da palavra-código $\mathbf{c}_i$ que maximiza $p(\mathbf{r}|\mathbf{c} = \mathbf{c}_i)$, o decodificador consegue minimizar a probabilidade de erro, provido que $p(\mathbf{c}) = \text{cte}$. Os decodificadores que seguem esse princípio são denominados de **decodificadores de máxima verossimilhança** [26].

A probabilidade condicional $p(\mathbf{r}|\mathbf{c})$ é igual a probabilidade de ocorrência do padrão de erros $\mathbf{e} = (\mathbf{r} - \mathbf{c})$ dado que $\mathbf{c}$ foi transmitido. Se for considerado que os padrões de erro mais prováveis são aqueles que possuem pesos $\mathbb{W}$ mais baixos (no canal **BSC** isto se verifica facilmente), então a palavra-código $\mathbf{c}_i$ que maximiza $p(\mathbf{r}|\mathbf{c}_i)$ é a palavra-código que minimiza $d(\mathbf{r}, \mathbf{c}_i) = \mathbb{W}(\mathbf{r} - \mathbf{c})$. Dessa forma, a palavra-código transmitida de máxima verossimilhança é aquela que possui a menor distância de Hamming da palavra recebida $\mathbf{r}$. Um decodificador que opera sobre este princípio comete um **erro de decodificação** se a palavra recebida está mais próxima de uma palavra-código

incorreta do que da palavra-código correta. Pela definição, as palavras-código incorretas estão no mínimo a uma distância $d_{\min}$ da palavra-código correta. Sendo assim, os erros de decodificação somente poderão ocorrer se os padrões de erros introduzidos pelo canal tiverem pesos maiores ou iguais a $d_{\min}/2$. Isto também pode ser dito como segue.

Um código com distância mínima $d_{\min}$ pode corrigir todos os padrões de erros com pesos menores ou iguais a $[(d_{\min}-1)/2]$.

$\lfloor (d_{\min}-1)/2 \rfloor$ é o limite superior sobre o peso para o qual pode-se corrigir *todos* os padrões de erros. Algumas vezes é possível corrigir $(\lfloor (d_{\min}-1)/2 \rfloor + 1)$ erros ou mais em certos blocos.

A capacidade de correção de erros de um código de bloco $\mathbf{C}$ também pode ser analisada sob uma perspectiva geométrica.

A interpretação geométrica da correção de erros pode oferecer algumas definições úteis. A esfera de Hamming de raio t contém todas os possíveis vetores recebidos por um decodificador que estão a uma distância de Hamming $\leq t$ de uma palavra-código, portanto

Definição 16 *Define-se a Esfera de Hamming de raio R de um código $\mathcal{C}$ por*

$$b(x,R) = \{y \in S^l : d(x,y) \leq R\} \tag{B.5}$$

O Volume da esfera de Hamming mostrada na Figura B.3 é dado pelo número de l-uplas, vetores, dentro da bola, que é

$$V(b) = |b(x,R)| = V_q(l,t) = \sum_{j=0}^{t} \binom{l}{j} (q-1)^j \tag{B.6}$$

no qual t é a capacidade de correção do código.

Pode-se ver um exemplo da Esfera de Hamming na Figura B.4 centrada na palavra-código 0000000 com raio igual a 1.

Se um vetor recebido $\mathbf{r}$ cai dentro de uma esfera de Hamming, o decodificador seleciona a palavra-código $\mathbf{c}$ que está no centro da esfera como sendo a palavra transmitida de acordo com suas regiões de decodificação de acordo com a Figura B.5. Se o vetor

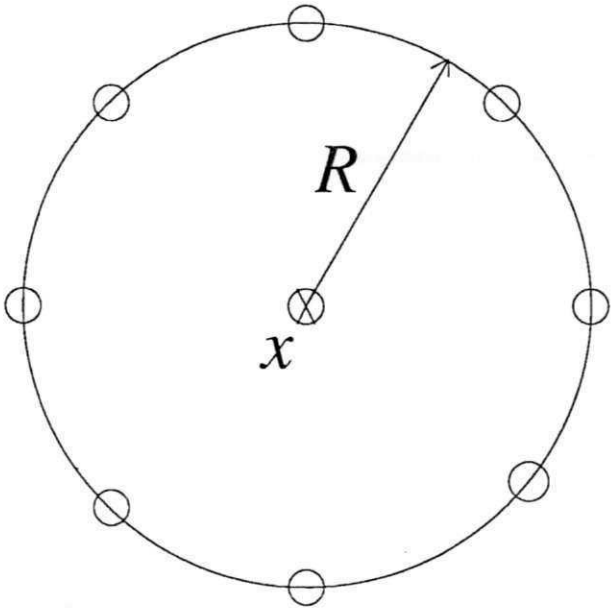

Figura B.3: Esfera de Hamming

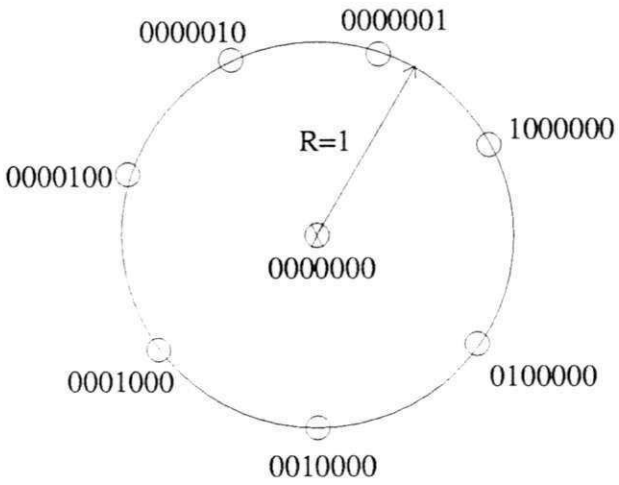

Figura B.4: Exemplo da esfera de Hamming

recebido cai em uma região entre as várias esferas de decodificação, o decodificador de máxima verossimilhança seleciona a palavra-código mais próxima como sendo a palavra transmitida como na Figura B.6 em caso de erro.

Um decodificador que corrige até o raio de empacotamento, melhor descrito no Capítulo 3, de um código tem probabilidade de decodificação correta dada por

$$p_c = \sum_{v=0}^{t} \binom{l}{v} P^v (1-P)^{l-v} \tag{B.7}$$

onde P é a probabilidade do canal cometer erros independentes.

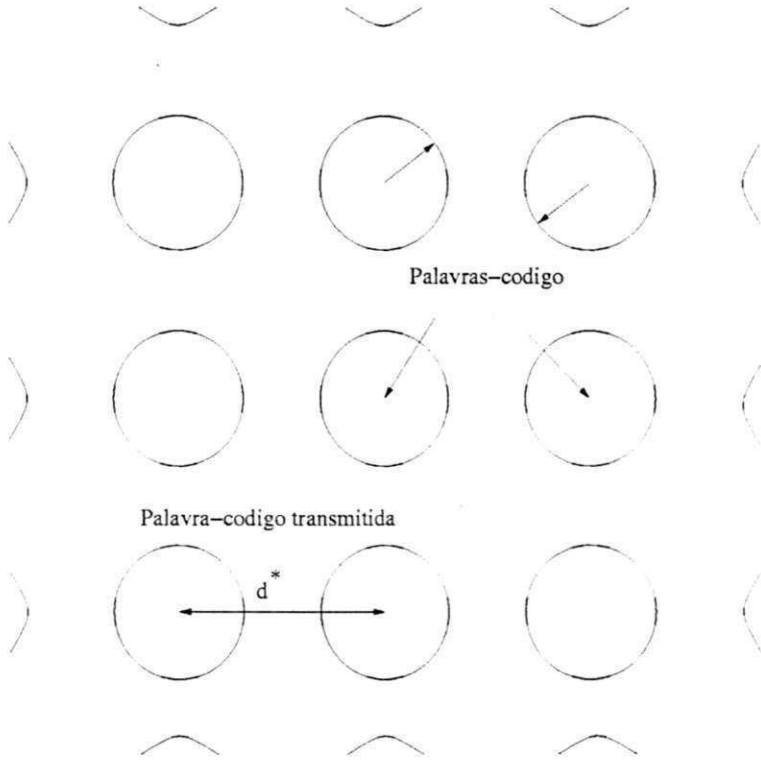

Figura B.5: Regiões de Decodificação

Um problema importante é determinar o número máximo de palavras- código $A_q(l, d_{\mathtt{min}})$ com comprimento fixo l e distância mínima $d_{\mathtt{min}}$ que podem ser escolhidas a partir de um espaço vetorial l-dimensional sobre GF(q). Este problema também pode ser formulado em função da redundância $\mathbf{r}$ do código como segue: Qual é a menor

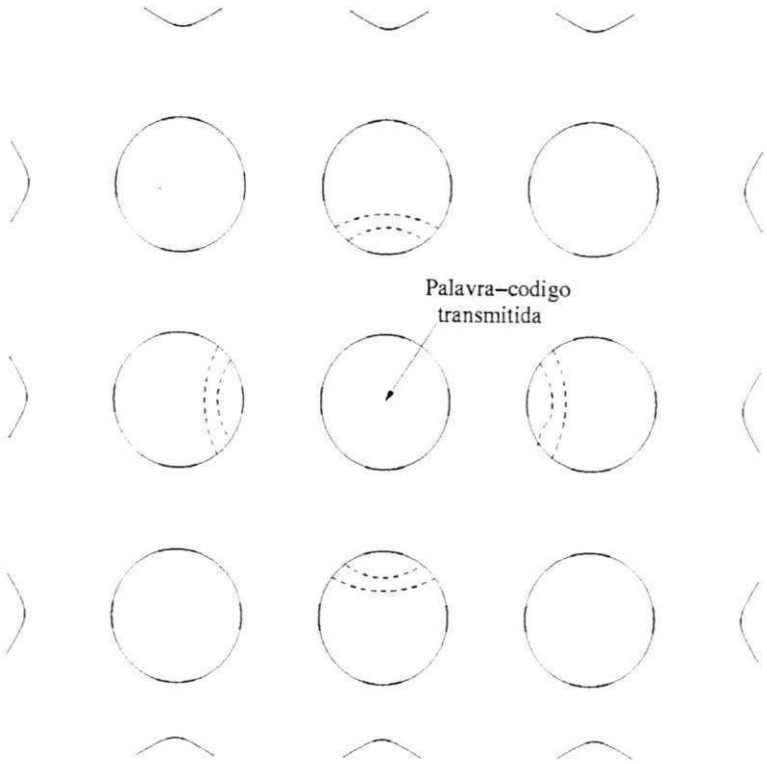

Figura B.6: Erro na decodificação

redundância requerida para um código q-ário de comprimento l com capacidade de corrigir t erros? O Teorema 5 responde essa questão.

Teorema 5 *Um código q-ário de comprimento l com capacidade de corrigir t erros deve possuir uma redundância r que satisfaça*

$$r \geq \log_q V_q(l,t) \tag{B.8}$$

Prova 1 *Cada uma das M palavras-código em* $\mathbf{C}$ *está associada a uma esfera de Hamming de raio t. As esferas não se sobrepõem, e portanto o volume total das esferas associadas a* $\mathbf{C}$ *não pode exceder o número total de vetores no espaço de todas as possíveis palavras recebidas. Segue então que* $M \cdot V_q(l,t) \leq q^l$ *logo,* $q^l/M \geq V_q(l,t)$. *Isto implica que* $r = l - \log_q M \geq \log_q V_q(l,t)$.

São poucos os códigos que satisfazem o Teorema 5 com igualdade. Tais códigos são denominados de **códigos perfeitos**.

B.3 Códigos de Blocos Lineares

A maioria dos bons códigos conhecidos pertencem a uma classe de códigos chamada de códigos lineares. Esta classe de códigos é definida por impor uma propriedade estrutural forte. Esta estrutura fornece direção na busca de códigos bons e também ajuda fazer codificadores práticos. Deve-se enfatizar que não estudamos códigos lineares porque eles são os melhores, mas porque até momento foram encontrados poucos indícios sobre como encontrar bons códigos não-lineares. A maioria das técnicas teóricas mais fortes são úteis apenas para códigos lineares e portanto a tentativa encontrar novos códigos são normalmente restritos a classe de códigos lineares.

Considere um código de bloco $\mathcal{C}$ que consiste de l-uplas $\{(c_0, c_1, \cdots, c_{l-1})\}$ de símbolos de $GF(q)$. $\mathcal{C}$ é um código linear q-ário se e somente se $\mathcal{C}$ forma um subespaço vetorial sobre $GF(q)$.

Os códigos lineares (não apenas binários) são ferramentas muito utilizadas na engenharia de comunicações digitais como forma de proteção das mensagens digitais contra erros eventualmente introduzidos durante a transmissão [30].

Os códigos lineares possuem algumas propriedades interessantes [26], sendo elas:

1. A combinação linear de qualquer conjunto de palavras-código é uma palavra-código. Uma conseqüência disso é que qualquer código linear possui um vetor nulo.

2. A distância mínima de um código linear é igual ao peso mínimo das palavras-código não-nulas.

Prova 2 *A distância mínima do código é calculada por*

$$d_{\min} = \min_{\mathbf{c},\mathbf{c}' \in \mathbf{C}, \mathbf{c} \neq \mathbf{c}'} d(\mathbf{c}, \mathbf{c}') = \min_{\mathbf{c},\mathbf{c}' \in \mathbf{C}, \mathbf{c} \neq \mathbf{c}'} \mathbb{W}(\mathbf{c} - \mathbf{c}'). \tag{B.9}$$

Desde que o código é linear, tem-se que $\mathbf{c}'' = (\mathbf{c} - \mathbf{c}')$ *é uma palavra-código e portanto*

$$d_{\min} = \min_{\mathbf{c}'' \in \mathbf{C}, \mathbf{c}'' \neq \mathbf{0}} \mathbb{W}(\mathbf{c}'') \tag{B.10}$$

3. Os padrões de erros que não podem ser detectados por um código linear são independentes da palavra-código transmitida e pertencem ao conjunto das palavras-código não nulas.

Prova 3 *Seja* $\mathbf{c}$ *uma palavra-código transmitida e* $\mathbf{c}'$ *a palavra-código recebida incorretamente. O correspondente padrão de erros* $\mathbf{e} = \mathbf{c} - \mathbf{c}'$ *deve ser uma palavra-código pela primeira propriedade e conseqüentemente não pode ser detectado.*

Qualquer conjunto de vetores bases para o subespaço pode ser usado como linhas para formar uma matriz $G_{k \times l}$ chamada de matriz geradora do código. O espaço de linha de G é o código linear $\mathcal{C}$; qualquer palavra-código é uma combinação linear de colunas de G. O conjunto de q^k palavras-código são linearmente independentes, e o número de linhas k é dimensão do subespaço.

Definição 17 *Código binário linear é um subspaço vetorial linear* $\mathcal{C} \subset S = \{0,1\}^l$ *no corpo finito dos inteiros* mod 2.

B.3.1 A Matriz Geradora e a Matriz de Paridade

Seja $\mathbf{i} = i_0, i_1, ..., i_{k-1}$ um bloco com k símbolos de informação codificado em uma palavra-código $\mathbf{c}$. A operação de codificação realizada por um codificador de bloco linear pode ser representada por

$$\mathbf{c} = \mathbf{iG} \tag{B.11}$$

sendo $\mathbf{G}$, denominada de matriz geradora para o código $\mathbf{C}$, definida por

$$\mathbf{G} = \begin{bmatrix} \mathbf{g_0} \\ \mathbf{g_1} \\ \vdots \\ \mathbf{g_{k-1}} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,l-1} \\ g_{1,0} & g_{1,1} & \cdots & g_{1,l-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,l-1} \end{bmatrix} \tag{B.12}$$

Note que qualquer palavra-código é simplesmente uma combinação linear dos vetores $\{\mathbf{g_j}\}$ de $\mathbf{G}$, isto é,

$$\mathbf{c} = (m_0, m_1, ..., m_{k-1}) \begin{bmatrix} \mathbf{g_0} \\ \mathbf{g_1} \\ \vdots \\ \mathbf{g_{k-1}} \end{bmatrix} = i_0\mathbf{g_0} + i_1\mathbf{g_1} + \cdots + i_{k-1}\mathbf{g_{k-1}} \tag{B.13}$$

Desde que o código linear (l,k) com q^k palavras-código é um sub-espaço de dimensão k, os vetores $\{\mathbf{g_j}\}$ da matriz geradora $\mathbf{G}$ devem ser linearmente independentes, ou seja, eles devem gerar um subespaço de dimensão k. Em outras palavras, os vetores $\{\mathbf{g_j}\}$ formam uma base para o código (l,k). Note que o conjunto de vetores da base não é único, e portanto, $\mathbf{G}$ também não é única. Observe também que a ordem da matriz $\mathbf{G}$ é k, desde que o subespaço tem dimensão k.

Qualquer matriz geradora de um código (l,k) pode ser transformada, através de operações sobre as linhas (e permutações sobre as colunas), em uma "forma sistemática", melhor explicada posteriormente.

$$\mathbf{G} = [\mathbf{I}_k|\mathbf{P}] = \left[\begin{array}{ccccc|ccc} 1 & 0 & 0 & \cdots & 0 & p_{11} & \cdots & p_{1l-k} \\ 0 & 1 & 0 & \cdots & 0 & p_{21} & \cdots & p_{2l-k} \\ \vdots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & p_{k1} & \cdots & p_{kl-k} \end{array}\right] \tag{B.14}$$

em que $\mathbf{I}_k$ é a matriz identidade k x k e $\mathbf{P}$ é uma matriz k x $(l-k)$ que determina os $l-k$ bits redundantes.

Observe que uma matriz geradora na forma sistemática gera um código de bloco linear no qual os primeiros k bits de cada palavra-código são idênticos aos bits de informação codificados e os $l-k$ bits restantes de cada palavra-código são combinações lineares dos k bits de informação. Esses $l-k$ bits redundantes são denominados de *bits de paridade.* O código (l,k) resultante é chamado de código sistemático.

É fácil verificar que, por ter sido o código definido como espaço vetorial linear de dimensão k, qualquer palavra código pode ser escrita como uma combinação linear das linhas da matriz geradora.

Exemplo 9

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{B.15}$$

O vetor de informação

$$i = [\,0 \quad 1 \quad 1\,] \tag{B.16}$$

portanto a palavra-código c é

$$c = [\,0 \quad 1 \quad 1\,] \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} = [\,0 \quad 1 \quad 1 \quad 1 \quad 0\,] \tag{B.17}$$

No contexto deste trabalho, os vetores $\mathbf{i}$ serão interpretados como *índices* que apontam para para um determinado vetor $\mathbf{c} = \mathbf{i}G$, este por sua vez encarado com um cromossomo binário.

Um código (l,k) gerado a partir de uma matriz $\mathbf{G}$ que não está na forma sistemática é chamado de código não sistemático. Entretanto, desde que toda matriz geradora é equivalente a uma matriz geradora na forma sistemática, os dois códigos lineares (l,k) gerados pelas duas matrizes geradoras equivalentes também são equivalentes. Então, cada código linear (l,k) é equivalente a um código linear sistemático (l,k).

Associado ao subespaço vetorial de dimensão k de qualquer código linear $\mathbf{C}$, existe um subespaço dual com dimensão $l-k$. Tal subespaço define o *código dual* $\mathbf{C}^{\perp}$ associado ao código $\mathbf{C}$. O código dual $\mathbf{C}^{\perp}$ é um código linear $(l, l-k)$ com q^{l-k} vetores-código. A matriz geradora para o código dual $\mathbf{C}^{\perp}$, denominada de matriz de paridade $\mathbf{H}$, consiste de $n-k$ vetores-código linearmente independentes pertencentes ao subespaço dual do código $\mathbf{C}$. Qualquer palavra-código $\mathbf{c}$ do código $\mathbf{C}(l,k)$ é ortogonal a qualquer palavra-código no código dual $\mathbf{C}^{\perp}(l, l-k)$. Sendo assim, as palavras-código do código $\mathbf{C}(l,k)$ são ortogonais as linhas da matriz de paridade, ou seja,

$$\mathbf{cH}' = \mathbf{0} \tag{B.18}$$

em que $\mathbf{0}$ denota um vetor linha nulo com $l-k$ elementos e $\mathbf{c}$ é uma palavra-código do código (l,k). Desde que as linhas de $\mathbf{G}$ pertencem ao código (l,k), tem-se que

$$\mathbf{GH}' = \mathbf{0} \tag{B.19}$$

em que $\mathbf{0}$ é agora uma matriz k x $(l-k)$ com elementos nulos.

A matriz de paridade $\mathbf{H}$ correspondente a uma matriz geradora $\mathbf{G}$ é definida por:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h_0} \\ \mathbf{h_1} \\ \vdots \\ \mathbf{h_{l-k-1}} \end{bmatrix} = \begin{bmatrix} h_{0,0} & h_{0,1} & \cdots & h_{0,l-1} \\ h_{1,0} & h_{1,1} & \cdots & h_{1,l-1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{l-k-1,0} & h_{l-k-1,1} & \cdots & h_{l-k-1,l-1} \end{bmatrix} \tag{B.20}$$

e na forma sistemática, é dada por:

$$\mathbf{H} = [-\mathbf{P}'|\mathbf{I}_{n-k}] \tag{B.21}$$

A matriz de paridade de um código fornece uma forma conveniente de se determinar a distância mínima do código.

Usando o Exemplo B.15 da matriz geradora G temos que

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{bmatrix} \tag{B.22}$$

Teorema 6 *A matriz geradora de C é a matriz de paridade do código dual $\underline{C}$*

como uma possibilidade para H. Como ela depende de G, que pode variar de acordo com as palavras-código componentes de sua matriz, o que implica que não existe apenas uma matriz geradora do código.

Teorema 7 *Considere um código de bloco linear* $\mathbf{C}$ *com matriz de paridade* $\mathbf{H}$. *A distância mínima de* $\mathbf{C}$ *é igual ao menor número de colunas distintas em* $\mathbf{H}$ *que somadas produzem o vetor nulo.*

Prova 4 *Desde que* $\mathbf{cH}' = \mathbf{0}$, *as colunas de* $\mathbf{H}$ *são linearmente dependentes. Seja* $\mathbf{c_j}$ *uma palavra-código de peso mínimo de um código linear* (l, k). *A palavra-código* $\mathbf{c_j}$ *deve satisfazer a condição* $\mathbf{c_j H}' = \mathbf{0}$. *Desde que o peso mínimo é igual a distância mínima, segue que a matriz* $\mathbf{H}$ *possui* $d_{\min}$ *colunas linearmente dependentes.*

A partir do Teorema 7 pode-se afirmar que não mais que $d_{\min} - 1$ colunas de $\mathbf{H}$ são linearmente independentes. Sabendo que a ordem de $\mathbf{H}$ é no máximo $l - k$, segue que $l - k \geq d_{\min} - 1$. Portanto, $d_{\min}$ possui um limite superior dado por

$$d_{\min} \leq l - k + 1 \tag{B.23}$$

Este limitante é conhecido como *cota de Singleton*. Códigos que atingem esta cota são conhecidos como **MDS** (*Maximum-Distance Separable*).

Dado um código (l, k) com distância mínima d^*, um novo código com os mesmos parâmetros pode ser obtido escolhendo dois componentes e transpondo os símbolos nestes dois componentes de todas as palavra-códigos. Isto, entretanto, nos dá um código que é diferente trivialmente do código original: ele é dito ser equivalente ao código original. Em geral, dois códigos lineares que são os mesmos exceto por uma permutação de componentes são chamados de códigos equivalentes.

As matrizes geradoras G e G' de códigos equivalentes podem ser relacionadas simplesmente. O código é um espaço de linha de G e portanto não é mudado por operações elementares entre as linhas de G. Permutação de componentes do código corresponde à permutação de colunas de G.

Com isso, dois códigos são equivalentes se e somente se as suas matrizes são relacionadas por

- Permutação de colunas
- Operações elementares de linhas

Com permutações de colunas, toda matriz geradora é equivalente a outra com uma matriz identidade $k \times k$ nas primeiras k colunas. Logo,

$$G = [\ I \ \mid \ P\ ] \tag{B.24}$$

e P é uma matriz $k \times l - k$. Toda matriz geradora pode ser reduzida para esta forma especial por uma seqüência de operações elementares entre linhas seguidas por uma seqüência de permutações de colunas. Denomina-se isto de forma sistemática da matriz geradora.

Supondo a Eq. B.24. Então claramente, a definição apropriada de um matriz de paridade sistemática é

$$H = [\ -P^{\mathcal{T}} \ \mid \ I\ ] \tag{B.25}$$

porque

$$GH^{\mathcal{T}} = [\ I \ \mid \ P\ ] \begin{bmatrix} -P \\ -- \\ I \end{bmatrix} = -P + P = 0 \tag{B.26}$$

Definição 18 *Um código sistemático inicia cada palavra-código com símbolos de informação não modificados. Os símbolos restantes são chamados de símbolos de paridade.*

B.3.2 Arranjo Padrão

O arranjo padrão é uma maneira de representar todas as bolas de Hamming centradas na palavras-códigos de um código linear. Seja $0, c_2, \cdots, c_{2^k}$ palavras-códigos em uma código (l,k). A tabela B.2 mostra um arranjo padrão genérico. A primeira linha da tabela contém as palavras-código. Das strings não usadas em S^l com distância de 1 das palavras-códigos nulas, escolhe-se qualquer palavra e chama-a de v_1 e escreve $0 + v_1, c_2 + v_1, \cdots, c_{2^k} + v_1$ na segunda linha. No l-ésimo passo, escolhe-se v_l, uma

0	c_2	c_3	$\cdots$	c_{2^k}
$0+v_1$	c_2+v_1	c_3+v_1	$\cdots$	$c_{2^k}+v_1$
$0+v_2$	c_2+v_2	c_3+v_2	$\cdots$	$c_{2^k}+v_2$
$\vdots$				$\vdots$

Tabela B.2: Arranjo Padrão

string previamente não usada que esta mais perto possível da palavra-código nula, e escreve $0+v_l, c_2+v_l, \cdots, c_{2^k}+v_l$ na n-ésima linha. O procedimento termina quando não existem string sobrando.

Pode se provar que no processo cada string de S^l é usada exatamente uma vez no arranjo padrão. As strings na primeira coluna são chamadas de líderes de classe. Nota-se que em cada coluna contém as strings da bola centrada na palavra-código do topo da coluna. Então se para completar o arranjo padrão foi necessário ultrapassar o raio t, a bola de Hamming interceptará uma outra bola.

Definição 19 *Classe Lateral é um subconjunto de S^l com elementos representados por*

$$r = c + v \tag{B.27}$$

em que v representa uma cadeia de S^l.

Pode ser verificado que o código e as suas *classes laterais* [30] são esquemas no sentido dado por Holland [4].

B.3.3 Síndrome

Considere um vetor recebido $u = c+e$, onde c é uma palavra-código e e um erro padrão incluso no canal. A matriz rH^T é o vetor síndrome s para o vetor recebido r. O vetor síndrome independe da palavra-código transmitida.

$$\begin{aligned} s &= rH^T \\ &= (c+e)H^T \\ &= cH^T + eH^T \\ &= 0 + eH^T \\ &= eH^T \end{aligned} \tag{B.28}$$

B.3.4 Código de Hamming

Códigos de Hamming foram a primeira grande classe de códigos binários lineares desenvolvidos para correção de erro. A sua primeira aplicação foi no controle de erro para telefonia a longa distância.

É comum referir-se a um código com parâmetros l, k, d, respectivamente comprimento da palavra código, dimensão e distância mínima, com *código* $(l,\ k,\ d)$. A primeira família não-trivial de códigos a surgir na literatura foi a dos códigos de Hamming. No caso binário, para cada inteiro $m \geq 2$ define-se um código de Hamming $(2^m - 1,\ 2^m - 1 - m, 3)$.

- Comprimento do Código $\rightarrow l = 2^m - 1$
- Número de símbolos de informação $\rightarrow k = 2^m - m - 1$
- Número de símbolos de paridade $\rightarrow l - k = m$
- Capacidade de correção de erro $\rightarrow t = 1$

As matrizes de paridade para os códigos de Hamming binários são muito fáceis de construir. Para um código de Hamming de comprimento $2^m - 1$, a matriz de paridade é formada por colunas construídas a partir de todas a m-úplas binárias não nulas.

O código de Hamming (7,4) pode, por exemplo, ser definido pela matriz de paridade apresentada na Equação abaixo.

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{B.29}$$

A ordem das colunas é arbitrária; um outro arranjo ainda define um código de Hamming (7,4), embora diferente do código associado a matriz de paridade apresentada acima. A matriz de paridade na Equação B.29 foi projetada especificamente para descrever um código sistemático. A correspondente matriz geradora apresentada na Equação B.30 mostra claramente que os bits de informação ocupam as primeiras 4 coordenadas de cada palavra-código.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{B.30}$$

Uma análise da matriz de paridade acima mostra que o menor número de m-úplas binárias distintas que somadas produzem o vetor nulo é sempre três, e portanto $d_{\min} = 3$, pelo Teorema 7. De forma geral, todos os códigos de Hamming possuem $d_{\min} = 3$ e são capazes de corrigir apenas um erro em cada palavra-código transmitida.

Será mostrado a seguir que os códigos de Hamming são códigos perfeitos, ou seja, satisfazem o limite de Hamming com igualdade (ver Teorema 5).

Teorema 8 *Todos os códigos de Hamming são perfeitos.*

Prova 5 *Todos os códigos perfeitos satisfazem a igualdade, para códigos binários,*

$$r = \log_q V_q(l,t) \qquad q = 2 \tag{B.31}$$

em que r é o número de bits de redundância e $V_q(l,t)$ é o volume da esfera de Hamming apresentado na Equação B.6.

Para um código binário $(2^m - 1, 2^m - m - 1)$ *com* $t = 1$, *o volume da esfera de Hamming pode ser calculado por:*

$$V_2(2^m - 1, 1) = \sum_{j=0}^{1} \binom{2^m - 1}{j} = 2^m \tag{B.32}$$

Desde que $r = l - k = m$, *o teorema fica provado.*

O Código de Hamming corrige apenas 1 erro pois possui $d = 3$. O código possui apenas 2^k palavras-código, portanto existem $2^{l-k} - 1$ classes laterais. O Código e todas as classes laterais constituem o espaço de busca.

Bibliografia

[1] A. L. Araújo e F. M. de Assis. "Algoritmo Genético Guiado por Código Algébrico". *Anais do Simpósio Brasileiro de Redes Neurais*, pages 79–83, Dezembro 1998.

[2] A. L. Araújo e F. M. de Assis. "Desempenho do Algoritmo Genético Guiado por Códigos em Funções de Difícil Solução". *Simpósio Brasileiro de Redes Neurais*, Novembro 2000.

[3] F. M. Assis. "Weight structure of binary codes and the performance of blind search algorithms ". *Simpósio Brasileiro de Redes Neurais*, pages 144–149, 2000.

[4] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Michigan, USA, 1975.

[5] D. E. Goldberg and K. Deb. "The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best". *Foundations of Genetic Algorithms*, pages 69–93, 1989.

[6] J. E. Baker. "Adaptive selection methods for genetic algorithms ". *Proceedings of an International Conference on Genetic Algorithms*, pages 100–111, 1985.

[7] L. D. Whitley. "The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best". *Proceedings of the Third International Conference on Genetic Algorithms, June*, pages 116–121, 1989.

[8] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Co., Reading, Mass., 1989.

[9] G.Lin X.Yao, Y.Liu. "Evolutionary Programming Made Faster". *IEEE Transactions on Evolutionary Computation*, 3(2):82–102, July 1999.

[10] D. Wolpert and W. Macready. "No Free Lunch Theorems for Optmization". *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[11] N. J. Sloane F. J. MacWilliams. *The Theory of Error-Correcting Codes.* North Holland, Amsterdam, New York, 1977.

[12] R. E. Blahut. *Principles and Practice of Information Theory.* Addison-Wesley Publishing Co., Reading, Mass., 1987.

[13] A. L. Araújo e F. M. de Assis. *Algoritmo Genético e Códigos Corretores de Erro.* UFPB, Brasil, 1998.

[14] F. M. de Assis. "Genetic Algorithms and Packing of Block Codes". *Proceedings of the International Conference on Telecommunications*, 3:1045–1047, April 1997.

[15] L. J. Eshelman J. D. Schaffer and D. Offutt. "Spurious Correlations and Premature Convergence in Genetic Algorithms ". *Foundations of Genetic Algorithms*, pages 102–112, 1995.

[16] F. M. de Assis and L. C. F. de Aquino. "Generating Fading-Resistant Constellations using Genetic Algorithm". *Proceedings of International Microwave and Optoelectronics Conference*, 2:719–723, August 1997.

[17] V. M. DaSilva and E. S. Sousa. "Fading-resistant Transmission from Several Antennas". *Proceedings of the The Third IEEE International Symposium on Personal, Indoor and Modbile Radio Communication - PIMRC'95*, 3:1218–1222, September 1995.

[18] H. P. Schwefel. *"Evolution and Optimum Seeking"* . John Wiley & Sons, Inc., New York, 1995.

[19] D. B. Fogel. "An Introduction to Simulated Evolutionary Optimization". *IEEE Transactions on Neural Networks*, 5(1):3–14, January 1987.

[20] K. A. De Jong and W. M. Spears. " Using Markov Chains to Analyze GAFOs ". *Foundations of Genetic Algorithms*, 3:115–137, 1997.

[21] H. Nein C. Lin and J. Hwu. "GA-based noisy speech recognition using two-dimensional cepstrum". *IEEE Transactions on Accoustic, Speech and Signal Processing*, 8(6):644–675, November 2000.

[22] E. Cantu-Paz,. "Markov chain models of parallel genetic algorithms". *IEEE Transactions on Evolutionary Computation*, 4(3):216–226, September 2000.

[23] R. Jonsson, J. Malec. "Towards computing the parameters of the simple genetic algorithm". *IEEE Transactions on Evolutionary Computation*, 1:516–520, January 2001.

[24] D. Ashlock, J. Davidson. "Texture synthesis with tandem genetic algorithms using nonparametric partially ordered Markov models". *IEEE Transactions on Evolutionary Computation*, 2:1099–1163, 1999.

[25] A. Agapie. "Adaptative genetic algorithms-modeling and convergence". *IEEE Transactions on Evolutionary Computation*, 1:1999–735, 1999.

[26] S. B. Wicker. *Error Control Systems for Digital Communication and Storage.* Prentice-Hall, Inc.,, 1995.

[27] J. G. Proakis. *Digital Communications.* McGraw-Hill Book Company, New York, 1989.

[28] J. M. Wozencraft and I. M. Jacobs. *Principles of Communication Engineering.* John Wiley & Sons, New York, 1965.

[29] C. E. Shannon. "A Mathematical Theory of Communication". *The Bell System Technical Journal*, 27:379–423, July 1948.

[30] R. E. Blahut. *Theory and Practice of Error Control Codes.* Addison-Wesley Publishing Company, Inc., Owego, NY, 1983.