

Computação Evolucionária

Prof. Heitor Silvério Lopes

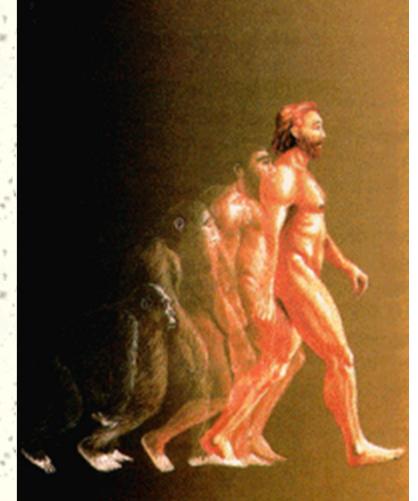
hslopes@utfpr.edu.br



CE2018-1

I A x I C

Inteligência Artificial (I A)



- Utiliza modelos computacionais inspirados no raciocínio humano

Inteligência Computacional (I C):

- Utiliza modelos computacionais inspirados na natureza (métodos bioinspirados)
- Também conhecida como Computação Natural ou *Soft Computing*

O “declínio” da Inteligência Artificial

Paradoxo 1:

- A IA imita a forma de pensar dos humanos.
- Não se sabe exatamente como os humanos pensam !
- Consequência: Os modelos computacionais utilizados são superficiais e incompletos.

Paradoxo 2:

- O objetivo da IA é simular computacionalmente o desempenho de um humano na execução de uma determinada tarefa
- Humanos “às vezes” falham !
- Consequência: Se os computadores imitam humanos, computadores deveriam falhar ?

Áreas da IC mais difundidas



Sistemas *Fuzzy*:

- Inspiração biológica: imprecisão do raciocínio humano

Redes Neurais Artificiais:

- Inspiração biológica: Funcionamento dos neurônios

Computação Evolucionária:

- Inspiração biológica: Evolução das espécies

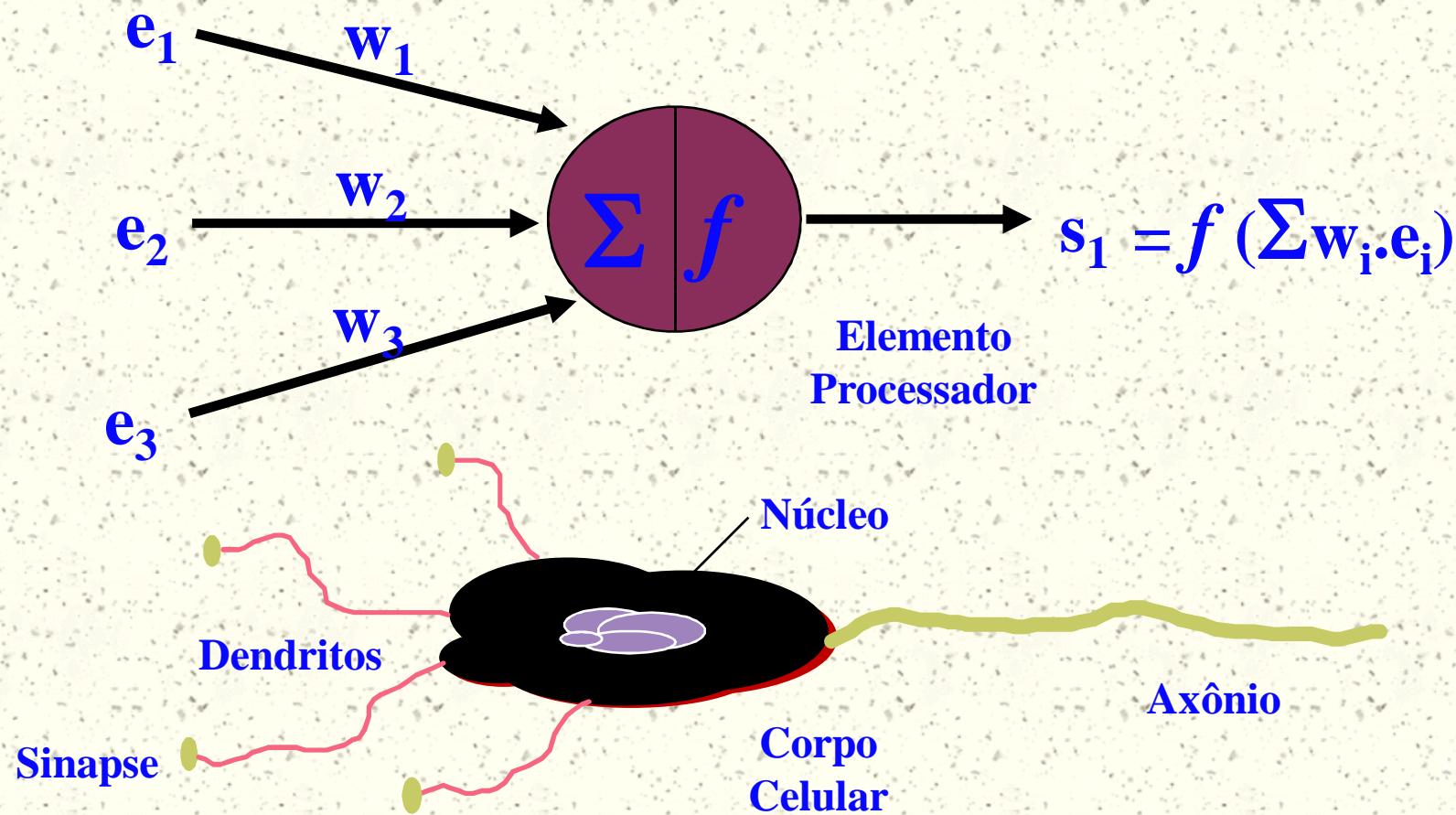
Inteligência de Enxames:

- Inspiração biológica: Comportamento emergente de grupos de animais

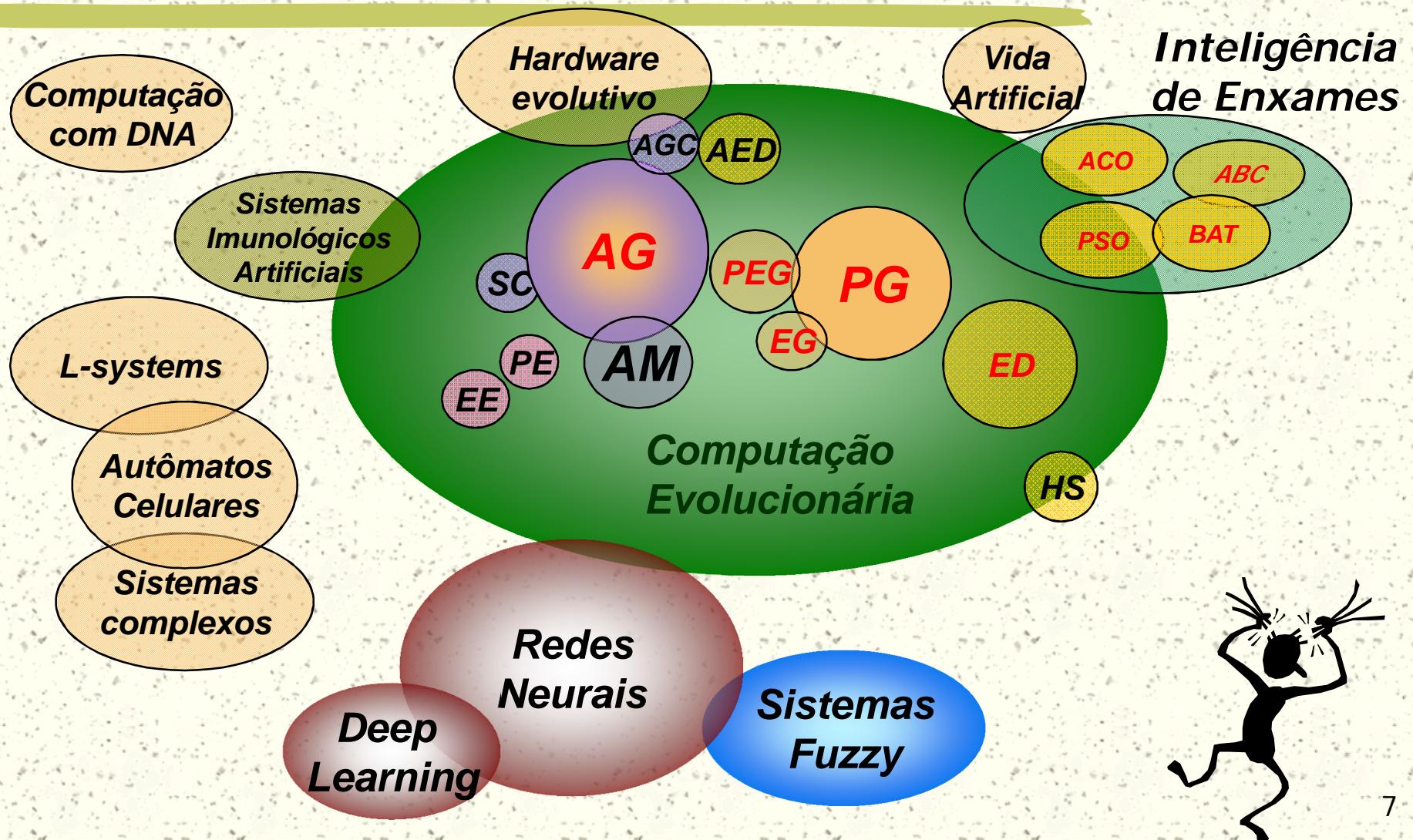
Sistemas fuzzy

- # Baseados na lógica *fuzzy* criada por Zadeh na década de 60
- # É uma forma de tratar incertezas e imprecisões inerentes ao conhecimento.
- # Utiliza variáveis linguísticas em vez de variáveis numéricas.
- # Tem um sólido embasamento teórico.
- # É mais natural.

Redes Neurais



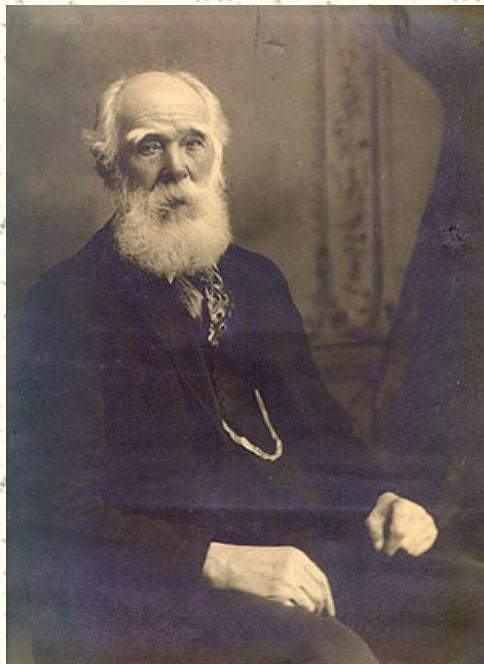
Inteligência Computacional



Origens - Evolucionismo



Charles Darwin, 1859

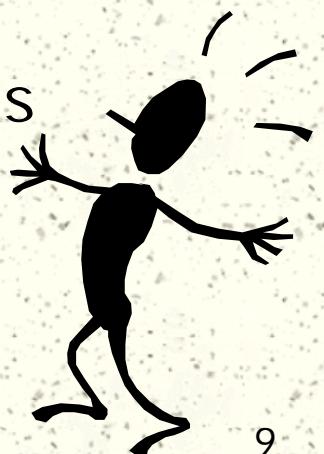


- “Sobre a origem das espécies por meio de seleção natural”
- A evolução dos seres vivos é baseada no Princípio da Seleção Natural:
 - # Indivíduos mais fortes e mais bem-adaptados ao ambiente têm maior chance de sobrevivência e de dar continuidade à sua espécie

Seleção natural e evolução

Importante!

- # A seleção natural é probabilística
- # A seleção natural atua sobre os indivíduos de uma espécie
- # A consequência a longo prazo é a evolução da espécie
- # Evolução (adaptação) como um processo “inteligente” de otimização
- # Inspiração para a construção de paradigmas computacionais que imitem a evolução das espécies, como forma de otimização

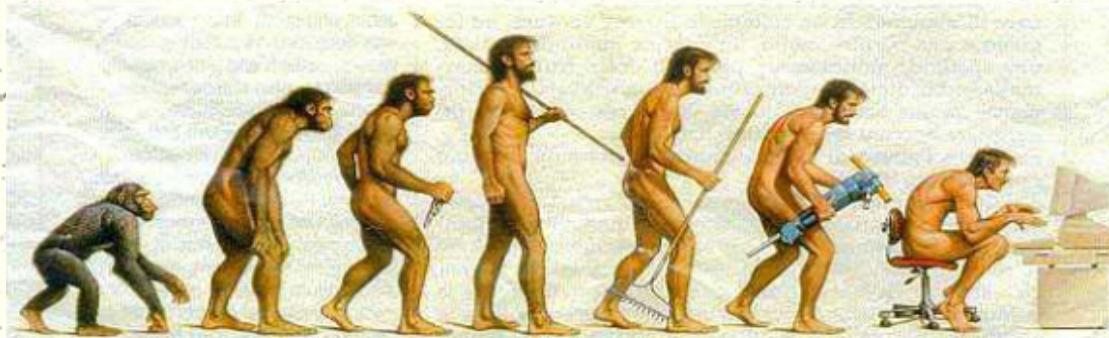


Hipóteses de Darwin

- # O número de filhos geralmente é maior do que o número de pais
- # O número de indivíduos de uma mesma espécie tende a ser mais ou menos constante
- # Dentro de uma mesma espécie os indivíduos apresentam pequenas diferenças entre si
- # Algum processo de variação continuada deve ser responsável pela introdução de novas informações na carga genética dos indivíduos
- # Não há limite para a introdução sucessiva de variações genéticas

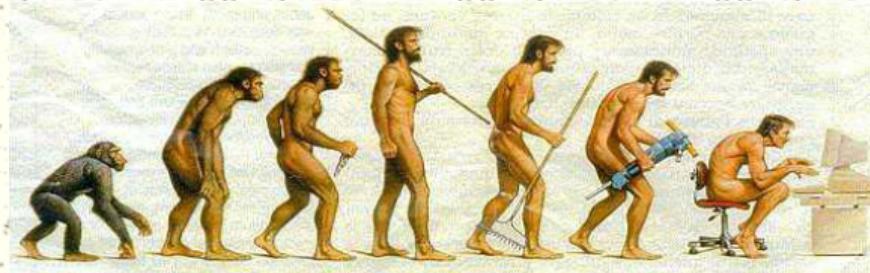
Precursors da Computação Evolucionária

- # Friedberg (1958): uso de processos evolutivos para resolver problemas computacionais
- # Box (1957): “operações evolutivas” para *design* e análise de equipamentos industriais.
- # Bremermann (1962): Aplicação de evolução simulada para problemas numéricos
 - Dificuldades: pouco ou nenhum embasamento teórico; suporte computacional incipiente; possibilidade de aplicação restrita; ceticismo da comunidade científica.



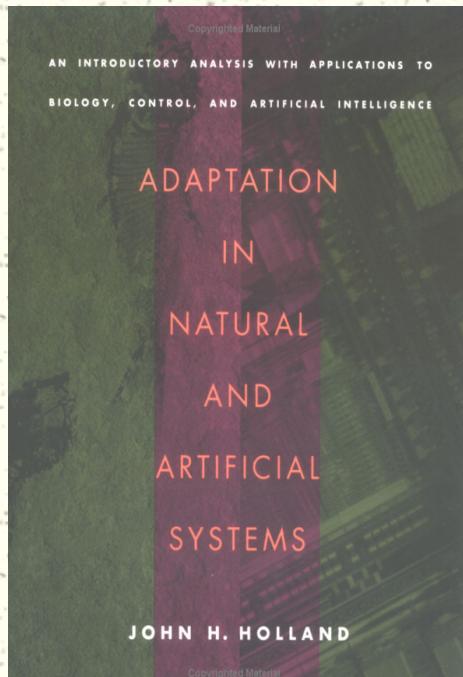
Evolução da Computação Evolucionária

- # As primeiras ideias surgiram em 1957-1962, porém havia pouco ou nenhum embasamento teórico, suporte computacional incipiente, e ceticismo da comunidade científica.
- # Décadas de 60/70 houve a emergência, de forma independente, das principais linhas de pesquisa:
 - Algoritmos Genéticos (AG)
 - Programação Evolucionária (PE)
 - Estratégias Evolucionárias (EE)



Computação Evolucionária – histórico

Algoritmo Genético (AG)



- Criado por John Holland na Universidade de Michigan (EUA) na **década de 60**
- Publicou o seu livro em **1975**: "**Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence**"
- "Sistema adaptativo artificial"; "planos reprodutivos generalizados"
- Foi o orientador de vários pesquisadores importantes da área, incluindo Rick Riolo, John Grefenstette, David Goldberg, Erick Goodman, Kenneth DeJong e John Koza



Prof. Heitor Silvério Lopes – UTFPR 2018



Computação Evolucionária – histórico



Programação Evolucionária (PE)

- Desenvolvido por Lawrence Fogel, Owens e Walsh em 1966 na Universidade da Califórnia em San Diego
- O objetivo da PE era aplicar à evolução simulada de máquinas de estados finitos para a previsão de séries não-estacionárias.
- Teve seu auge na década de 80, mas acabou caindo em desuso

Estratégias Evolucionárias (EE)

- Desenvolvido por Bienert, Ingo Rechenberg e Hans-Paul Schwefel em 1965 na Universidade Técnica de Berlin
- EE foi utilizado para otimização numérica em problemas de *design* industrial

Escopo da Computação Evolucionária

- O escopo de aplicação de CE é em problemas de Busca e Otimização
- Grandes áreas de aplicação:
 - Engenharias
 - Computação
 - Matemática aplicada
- Problemas de outras naturezas:
 - Modelagem como um problema de otimização.
 - P.ex. Aprendizado: $\max(\text{qtd. info. absorvidas})$; $\min(\text{qtd. erros conceituais})$; $\max(\text{capacidade de extrapolação})$, $\min(\text{tempo aprendizado})$



Algumas áreas de aplicação de CE

Computação

- Mineração de dados, aprendizado de máquina, engenharia de software, robótica, processamento de imagens, ...

Engenharias

- Eletro-eletrônica, civil, mecânica, desenho industrial

Matemática

- Pesquisa operacional, otimização, séries temporais

Biologia

- Bioquímica, ecologia

Economia

- Econometria, modelagem

Física

- Nuclear, óptica

Artes (!!)

- Música, pintura

CE como método de engenharia

- # Otimização \leftrightarrow Engenharia
- # Projeto de engenharia:
 - Muitos parâmetros a serem otimizados
 - Muitos graus de liberdade
 - Muitas restrições a serem satisfeitas
- # CE:
 - Facilita a tarefa de projeto gerando soluções subótimas ou mesmo ótimas em um tempo razoável
- # CE emprega um método intuitivo:
 - Processo criativo
 - Refinamento iterativo

Vantagens das técnicas de CE

- Não requerem um conhecimento matemático profundo do problema ao qual é aplicado.
- Têm robustez
- Requerem pouco esforço de implementação
- São facilmente hibridizáveis com outras técnicas
- São facilmente adaptáveis a muitas classes de problemas, inclusive problemas multiobjetivos
- São implicitamente e explicitamente parallelizáveis
- São capazes de manipular de restrições

Desvantagens das técnicas de CE

- Não garantem encontrar a solução ótima em tempo finito
- Há pouco embasamento teórico - a prática se desenvolveu mais do que a teoria
- O ajustes dos parâmetros de controle requer conhecimento prévio, calibração com experimento fatorial ou tentativa-e-erro
- Não são intrinsecamente melhores do que qualquer outro algoritmo de otimização (*"No free-lunch theorems"*)

Métodos para otimização

Métodos fortes:

- Para problemas específicos onde há linearidade, diferenciabilidade, estacionariedade
- Em geral garantem a obtenção da solução ótima

Métodos específicos:

- Para problemas muito particulares

Métodos fracos:

- Para problemas genéricos onde pode existir não-linearidade, não-estacionariedade (etc).
- Não garantem a obtenção da solução ótima, só eventualmente podem fornecer uma solução satisfatória

Métodos de otimização

Numéricos:

- Analíticos: derivadas parciais = 0
- Diretos: técnicas de gradiente (*steepest descent* ou *hill-climbing*)



Enumerativos:

- Busca exaustiva
- Programação dinâmica



Probabilísticos: →Heurísticas

- Busca aleatória
- *Simulated annealing*
- Computação Evolucionária & Inteligência de Enxames



Métodos enumerativos

- # Excelentes para um grande número de problemas, entretanto:
 - Aplicável somente a problemas de “dimensões pequenas”
 - Aceitável quando envolve tempos computacionais “razoáveis”
- # Tendem a ser cada vez mais utilizados, à medida que a capacidade computacional disponível aumenta
- # Não servem para problemas com complexidade Não-Polinomial (NP)

Quando pode ser interessante utilizar uma técnica de CE ?



- # Quando há um método forte/específico para o problema, mas que é inviável a sua aplicação
- # Quando a complexidade do problema torna inviável a sua formulação matemática
- # Quando o número de possíveis soluções a serem examinadas leva à uma explosão combinatória intratável
- # Quando o problema é fortemente não-estacionário
- # Quando não existir outra alternativa viável !

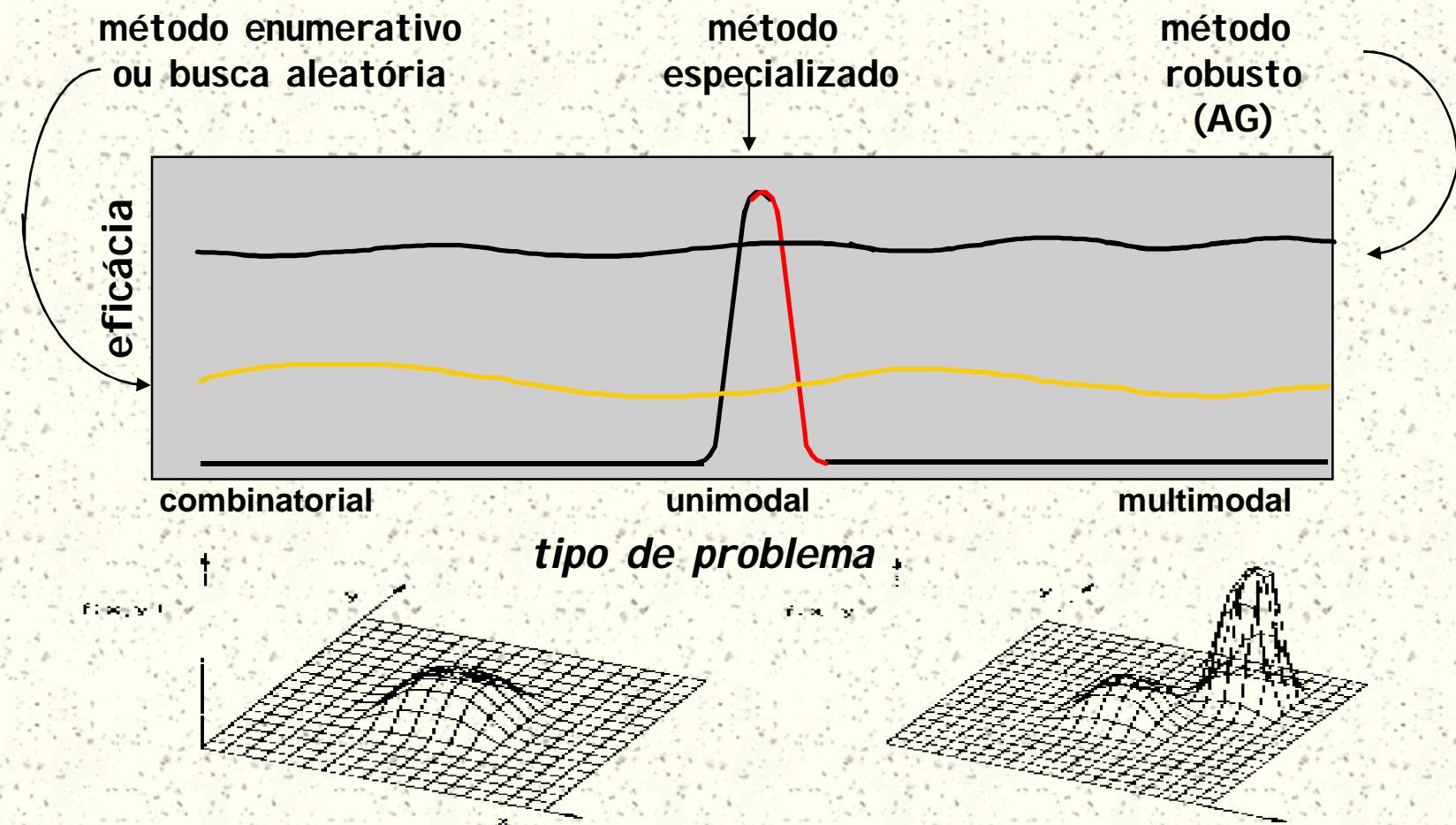


Necessidade de métodos heurísticos

- Alto custo computacional para métodos fortes
- Alta complexidade no equacionamento e resolução de problemas reais
- Métodos heurísticos são executados em tempos aceitáveis, porém não garantem a obtenção da solução ótima, nem mesmo garantem a obtenção de uma solução factível
- Entretanto, o objetivo de um método heurístico é tentar encontrar uma solução “aceitável” de maneira simples e rápida.

Robustez de métodos de otimização

Compromisso entre eficiência e eficácia



Algoritmo evolutivo: refinamentos sucessivos

Abordagem “intuitiva”: tentativas e refinamentos sucessivos:

- 1 gere soluções para o problema
- 2 avalie as soluções
- 3 se a melhor solução satisfaz, pare
- 4 selecione as melhores soluções
- 5 construa novas soluções utilizando partes das melhores soluções
- 6 eventualmente modifique as soluções
- 7 volte ao passo 2

Diferenças entre AG e outros métodos de otimização

- # Codificação: (fenótipo \leftrightarrow genótipo)
 - ─ AG's trabalham com conjuntos de parâmetros codificados e não os parâmetros propriamente ditos.
- # Busca paralela: (paralelismo implícito)
 - ─ AG's realizam a busca utilizando uma população de pontos simultaneamente, não um único ponto
- # Busca probabilística
 - ─ AG's utilizam regras de transição probabilísticas, não determinísticas.

“Ingredientes” de problemas de otimização

Função objetivo:

- a qual se quer maximizar ou minimizar.
- exemplos: $\max(\text{lucro})$, $\min(\text{custo})$
- pode não existir ou ser múltipla

Conjunto de variáveis:

- que afetam o valor da função objetivo
- em problemas interessantes, este conjunto pode ser muito grande

Conjunto de restrições:

- não permite que o conjunto de variáveis assuma determinados valores

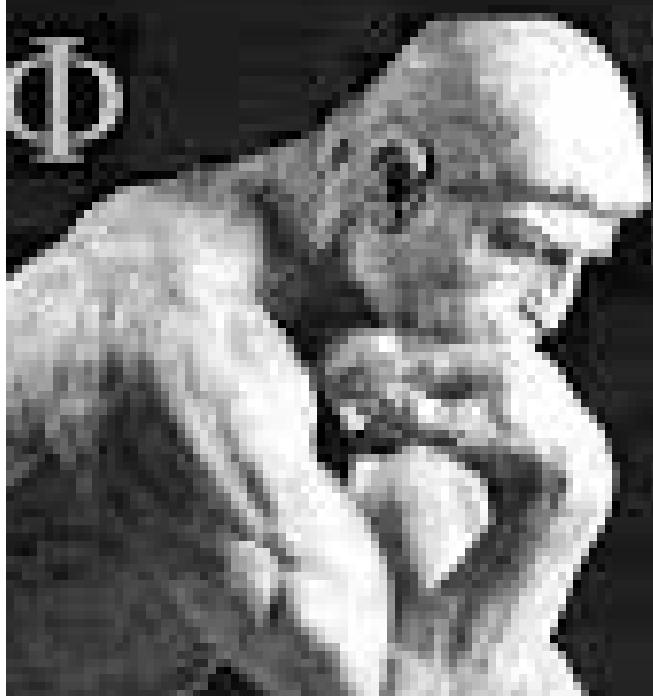
“Ingredientes” de problemas difíceis

- # Multimodalidade (máximos locais)
- # Grande conjunto de restrições
- # “Ruído”
- # “Deceptividade” (*deception*)
- # Isolamento do ótimo desejado (*Royal Road*)
- # Não-estacionariedade (variação dinâmica)
- # Espaço de busca **arbitrariamente** grande ***



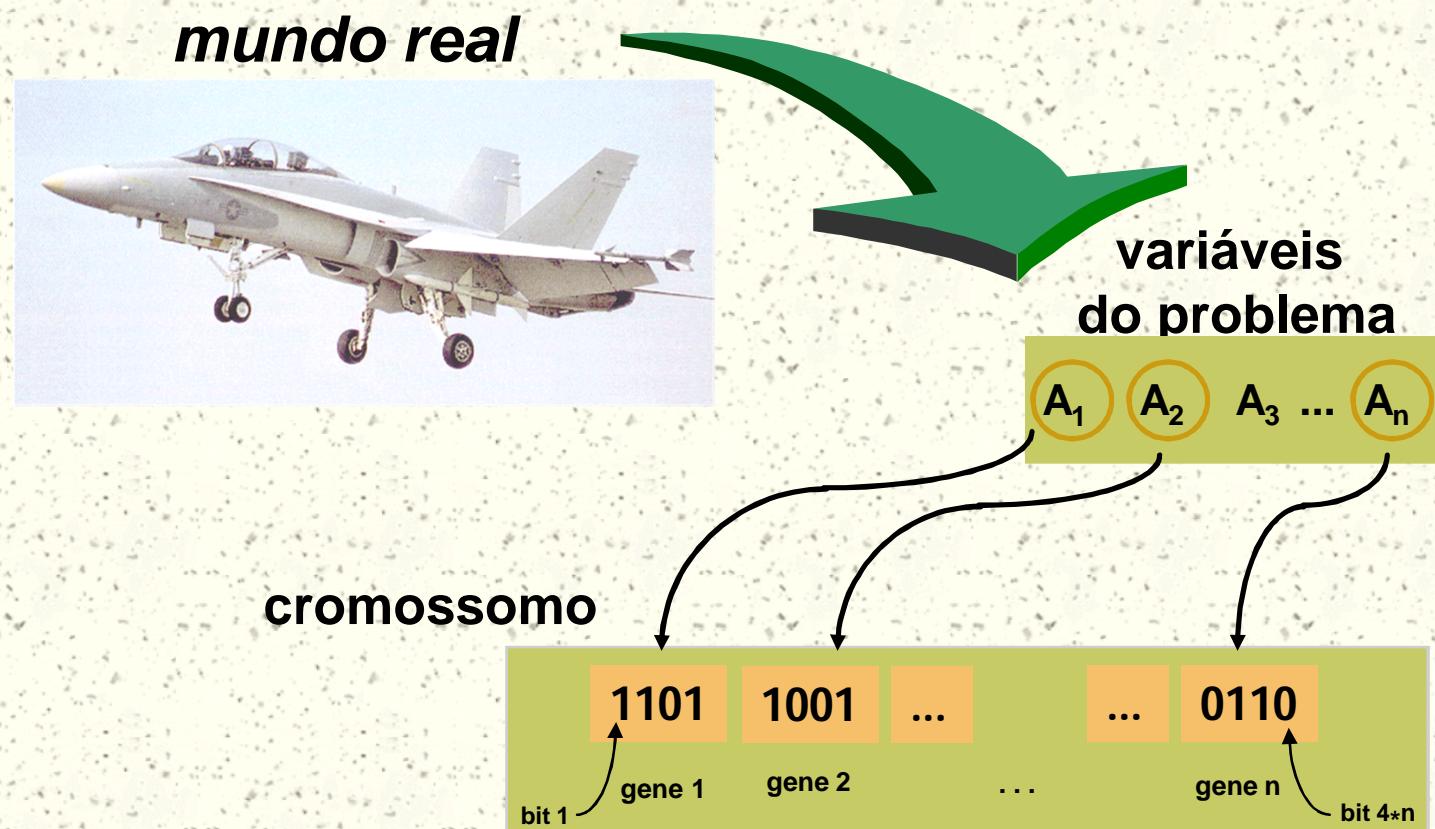
"Não existem métodos
fáceis para resolver
problemas difíceis."

René Descartes



Algoritmo genético #1: Codificação das variáveis do problema

■ Mapeamento fenótipo x genótipo



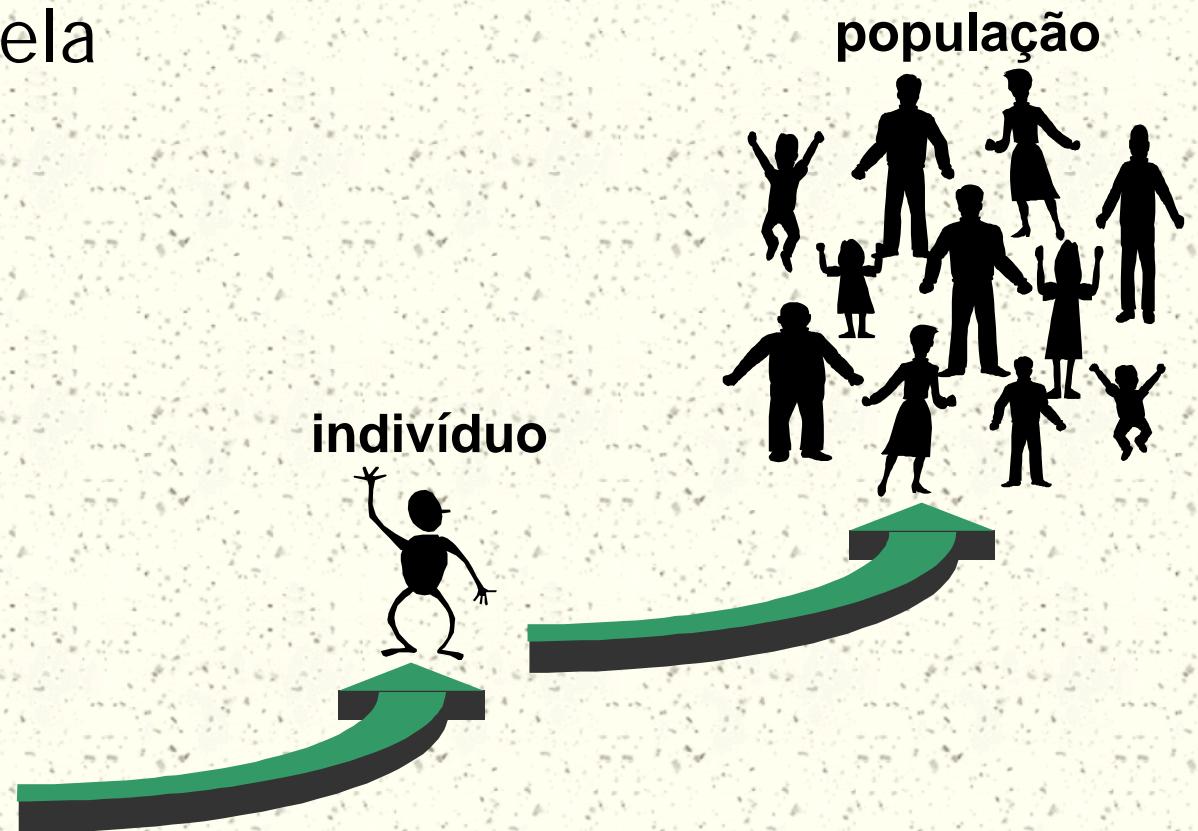
Algoritmo genético #2: População" de possíveis soluções

■ Busca paralela

cromossomos

1101	1001	0110
1001	0110	1101

indivíduo



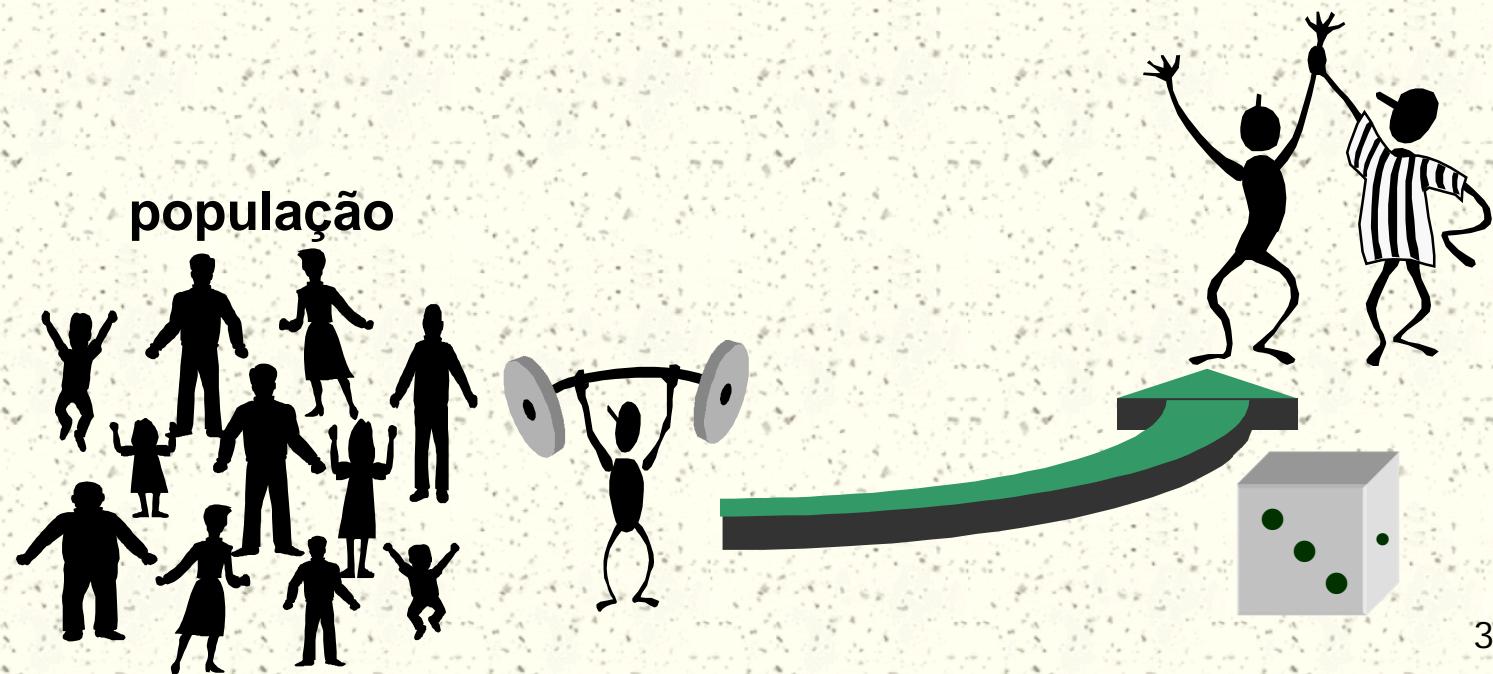
Algoritmo genético #3: Função de *fitness*

- Avalia a qualidade da solução para o problema



Algoritmo genético #4: Seleção

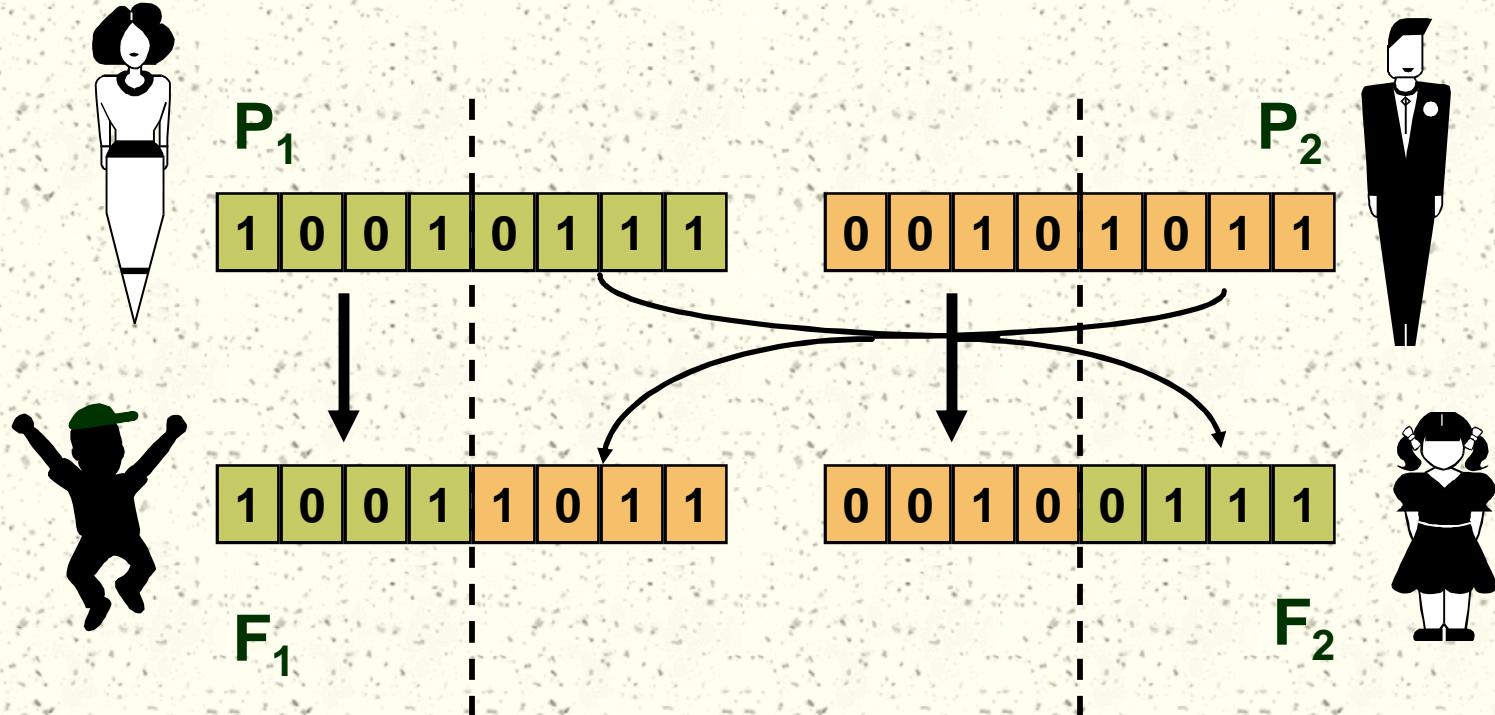
- Implementa e guia o processo evolutivo
- Seleciona os indivíduos mais aptos a se reproduzir e passar o seu material genético
- Métodos probabilísticos ou determinísticos



Algoritmo genético #5:

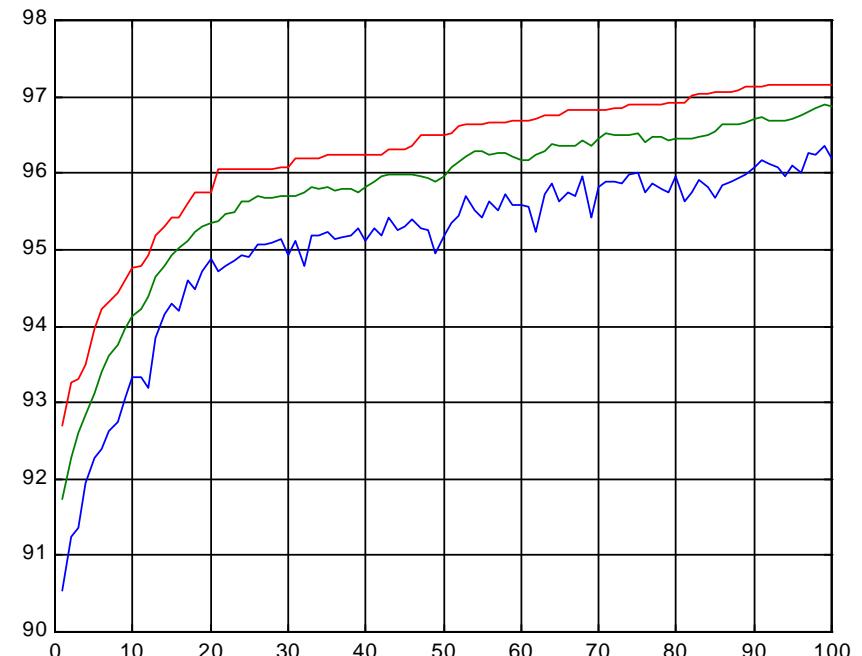
Reprodução e operadores genéticos

- Crossover: recombinação de material genético de dois pais - realiza busca local
- Mutação: variação aleatória - realiza busca global



Algoritmo genético #6: Critério de parada do algoritmo

- Número predeterminado de gerações
- Quando não há mais melhora
- Quando não há mais diversidade
- Quando atingiu o ótimo



Otimização

Definição de otimização:

- “Processo de melhoramento iterativo de uma solução para um problema, com respeito a uma função objetiva específica.”

Problemas típicos da área de otimização:

- Maximização (ou minimização) de funções algébricas
- Problemas combinatoriais

- ex. Problema do caixeiro viajante, problema da mochila

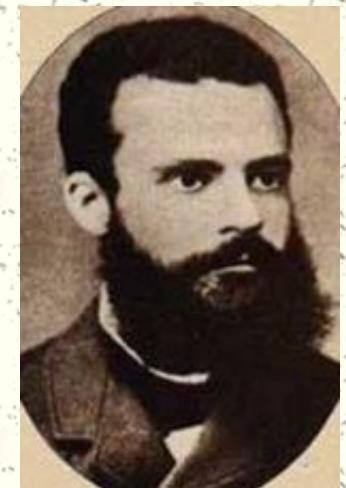
- Projetos de engenharia:

- maximização de desempenho
 - minimização de custo

Otimalidade de Pareto

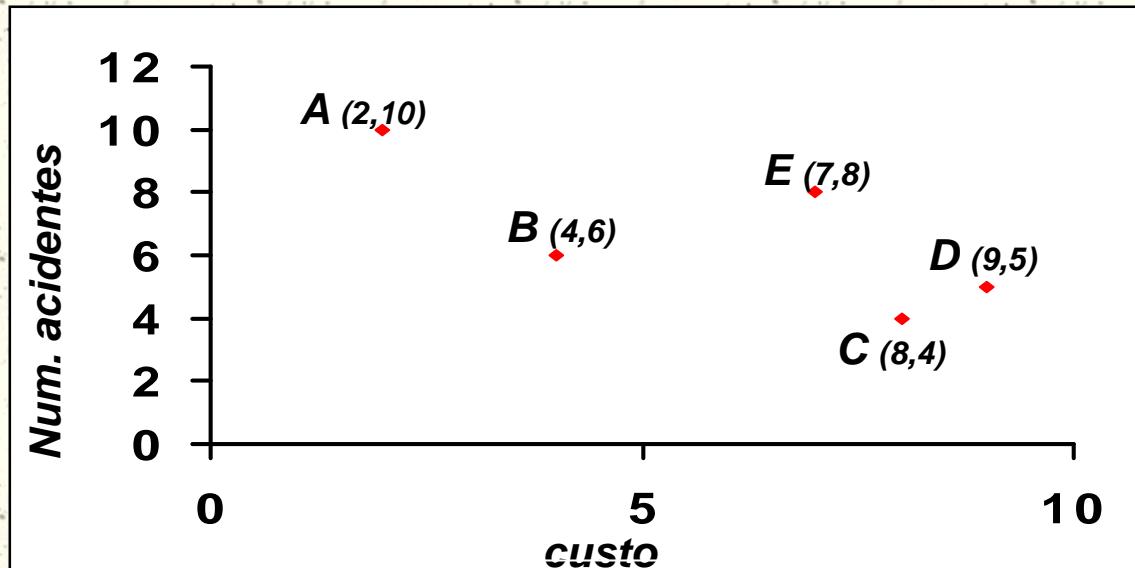
Vilfredo Pareto:

- Engenheiro, político, sociólogo e economista franco-italiano
- A **Lei de Pareto** (também conhecido como **princípio 80-20**), afirma que para muitos fenômenos, 80% das consequências advém de 20% das causas.
- Uma situação é ótima, no sentido de Pareto, se não for possível melhorar a utilidade de um agente sem degradar a utilidade de qualquer outro agente do sistema.



Otimização multiobjetivos

- # O conceito de ótimo não é óbvio e deve respeitar a individualidade de cada critério
- # Optimalidade de Pareto:

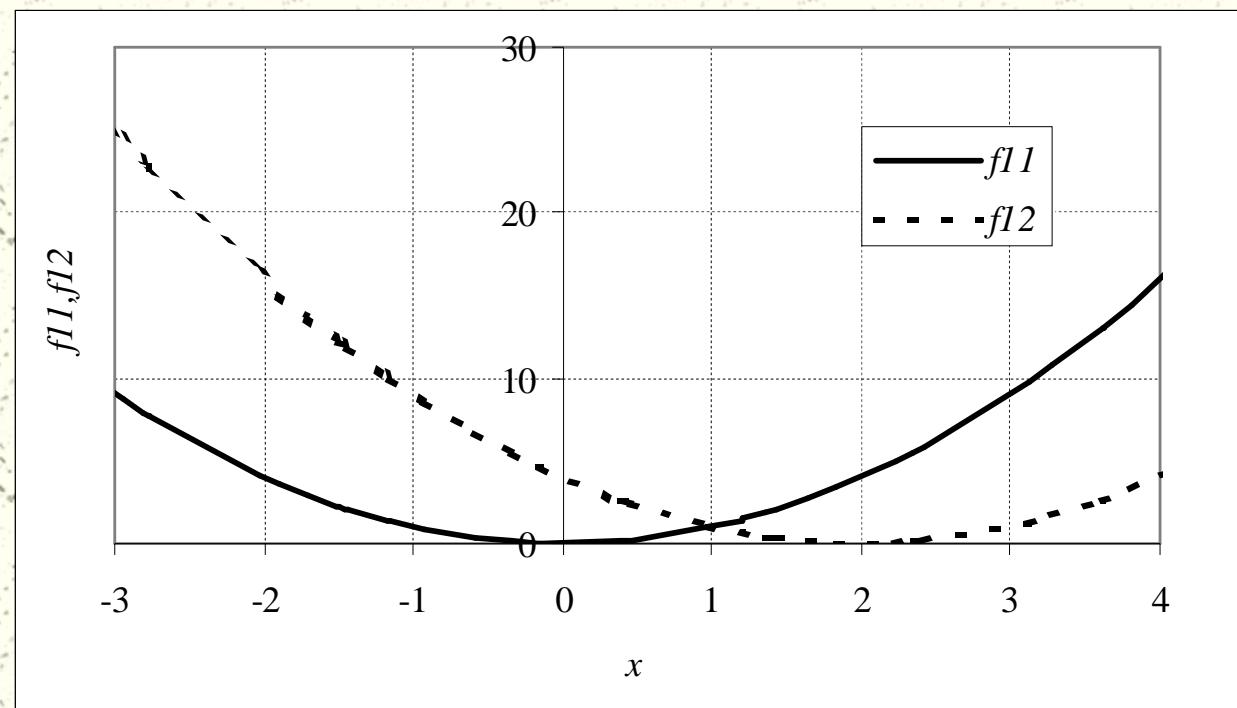


- # Dominância:
 - nesta caso: P-ótimo={A,B,C}

$$(x < y) \Leftrightarrow \forall i (x_i \leq y_i) \wedge \exists i (x_i < y_i)$$

Otimização multiobjetivos

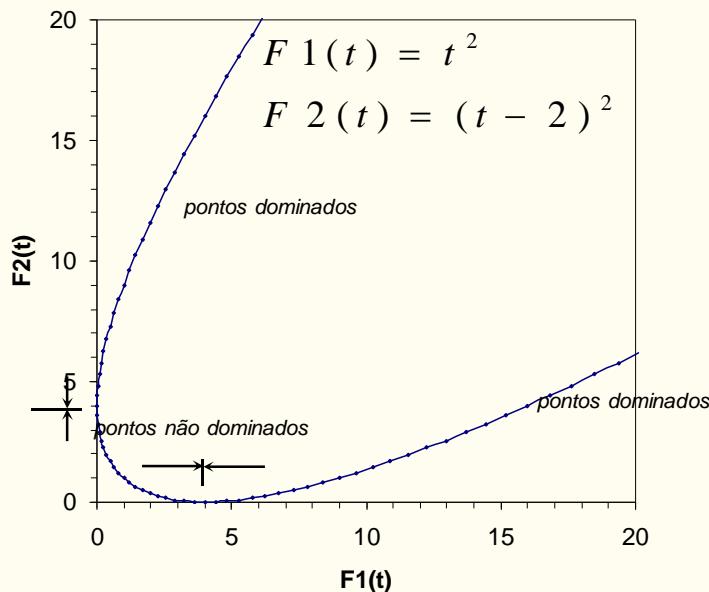
- # Exemplo de duas funções que definem objetivos diferentes



Otimização multiobjetivos

plano de Pareto

- # Mostra as regiões dominadas e não dominadas
- # Problemas que exigem uma solução P-ótima devem utilizar técnicas de nichos
- # Técnica tradicional: ponderação.



$$F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix}$$

$n \geq 2$

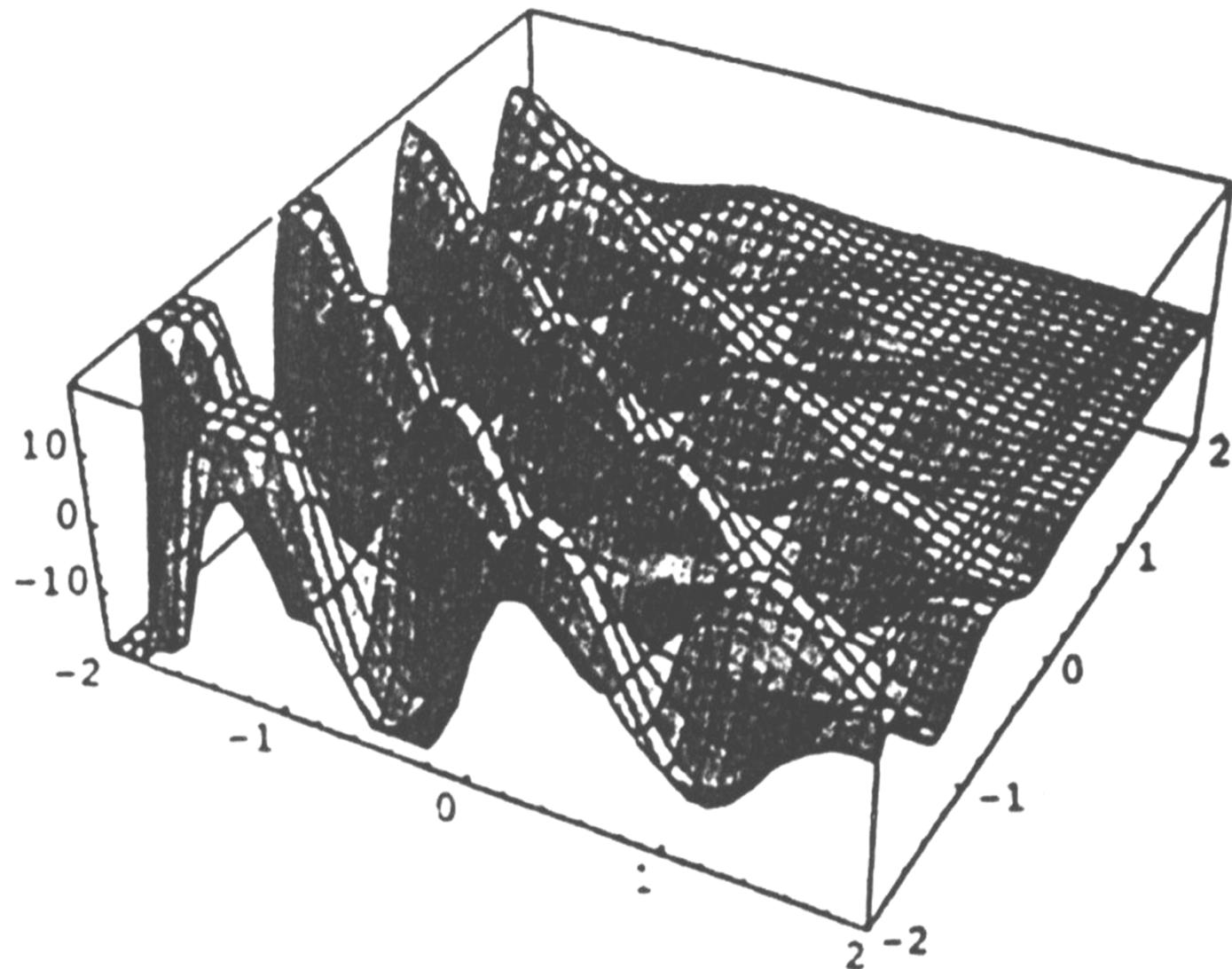
$$\min_{x \in C} F(x) = \sum_{i=1}^n \alpha_i f_i(x)$$

onde $\alpha_i > 0, i=1, K, n$

Funções de teste

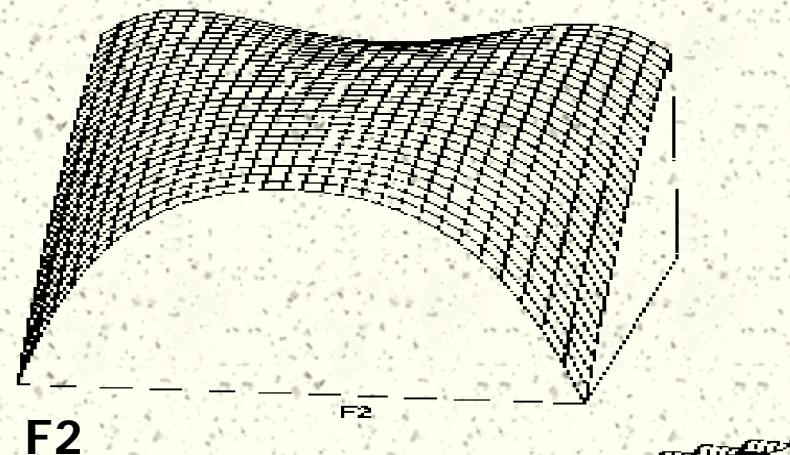
- # Existem inúmeras funções de teste para algoritmos de otimização:
 - Funções de DeJong (F1...F5)
 - Langermann, Griewangk, Rastrigin, Eason, Michalewicz, Rosenbrock, etc...
- # Em geral é definido um conjunto de funções, definidas para o espaço n -dimensional
- # São frequentemente utilizadas como *benchmark* para algoritmos heurísticos de otimização (p.ex. AG)

Exemplo de função multimodal (3D)



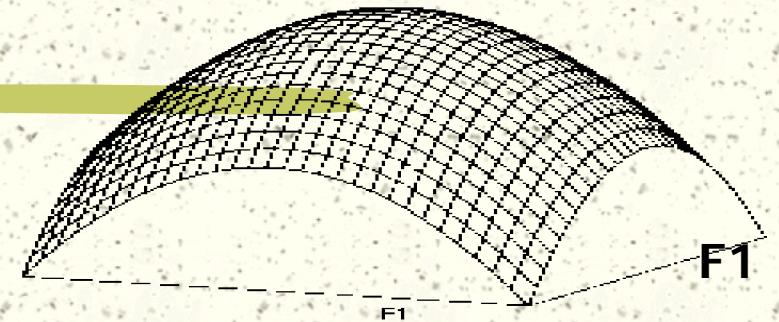
Funções “clássicas” de DeJong

- Contínuas/discretas
- Máximo único/múltiplos máximos

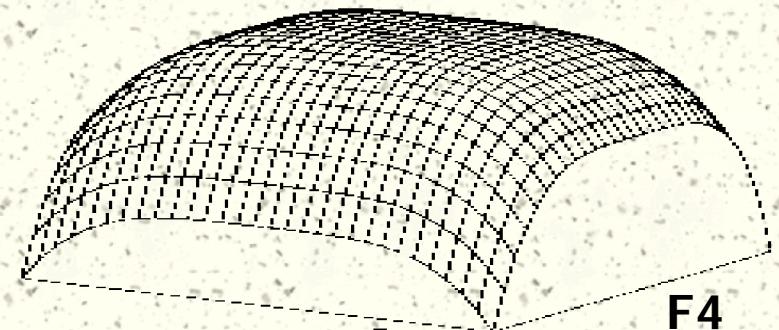


F2

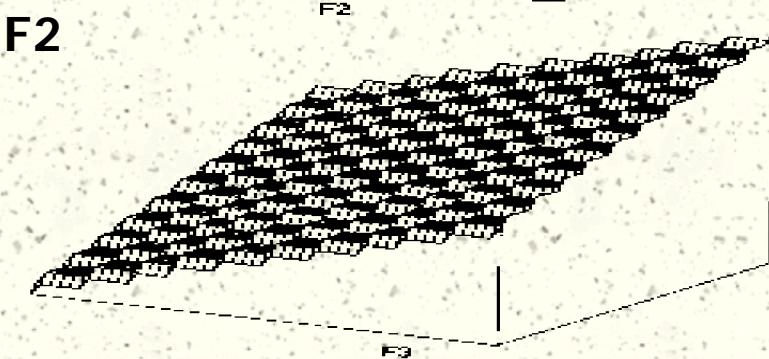
F3



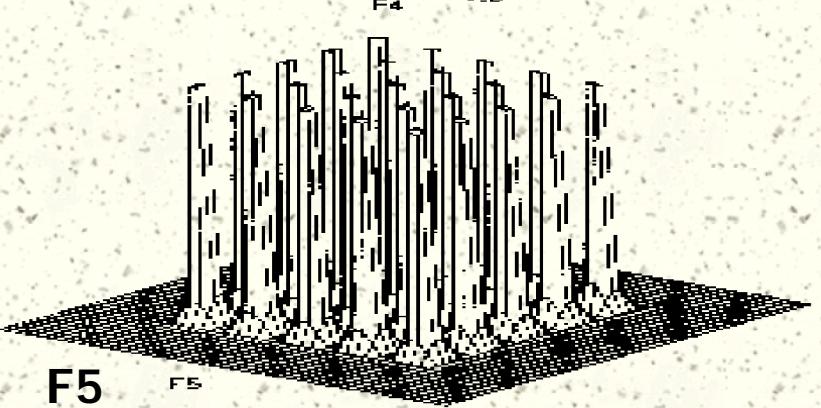
F1



F4

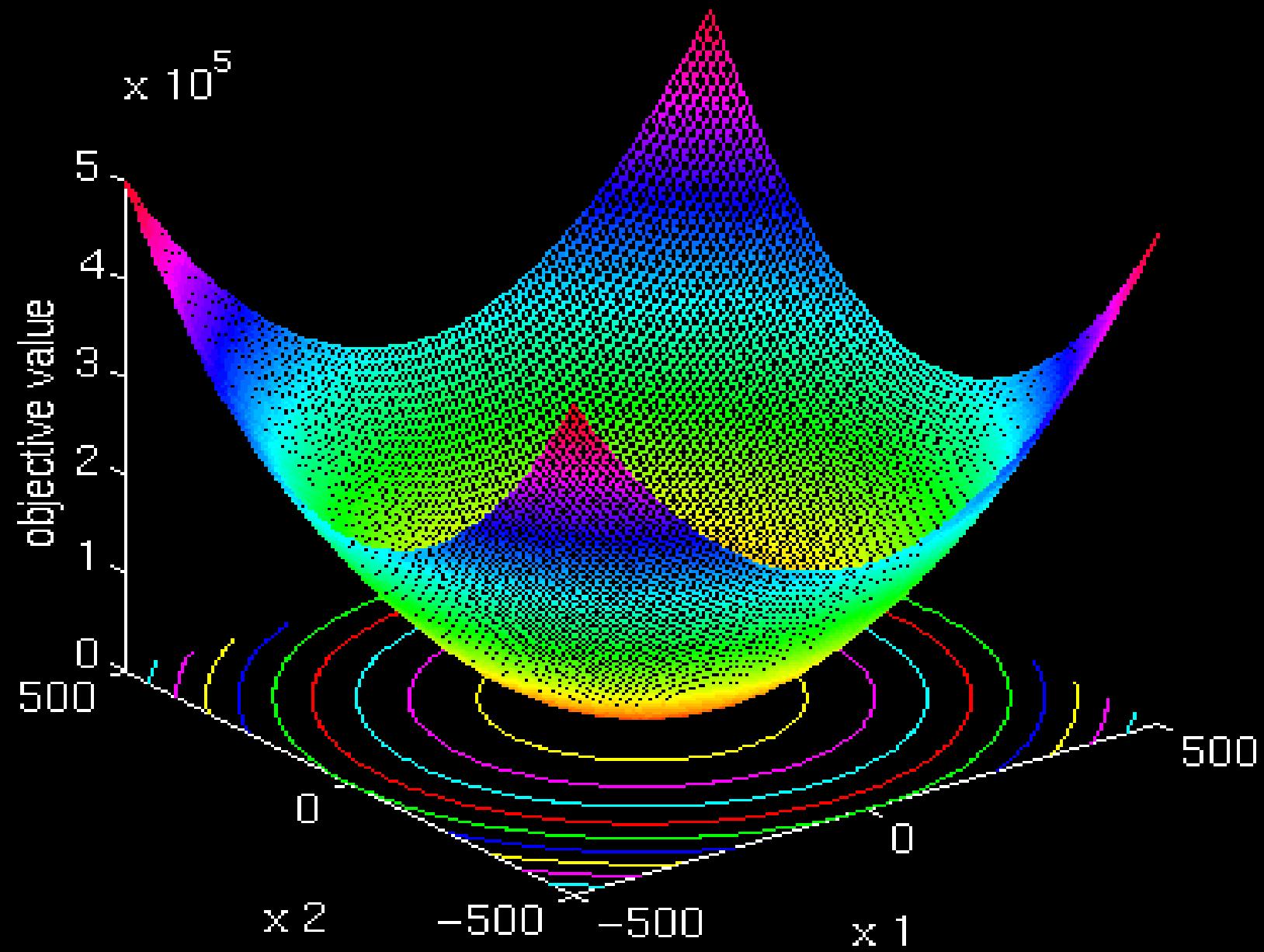


F3

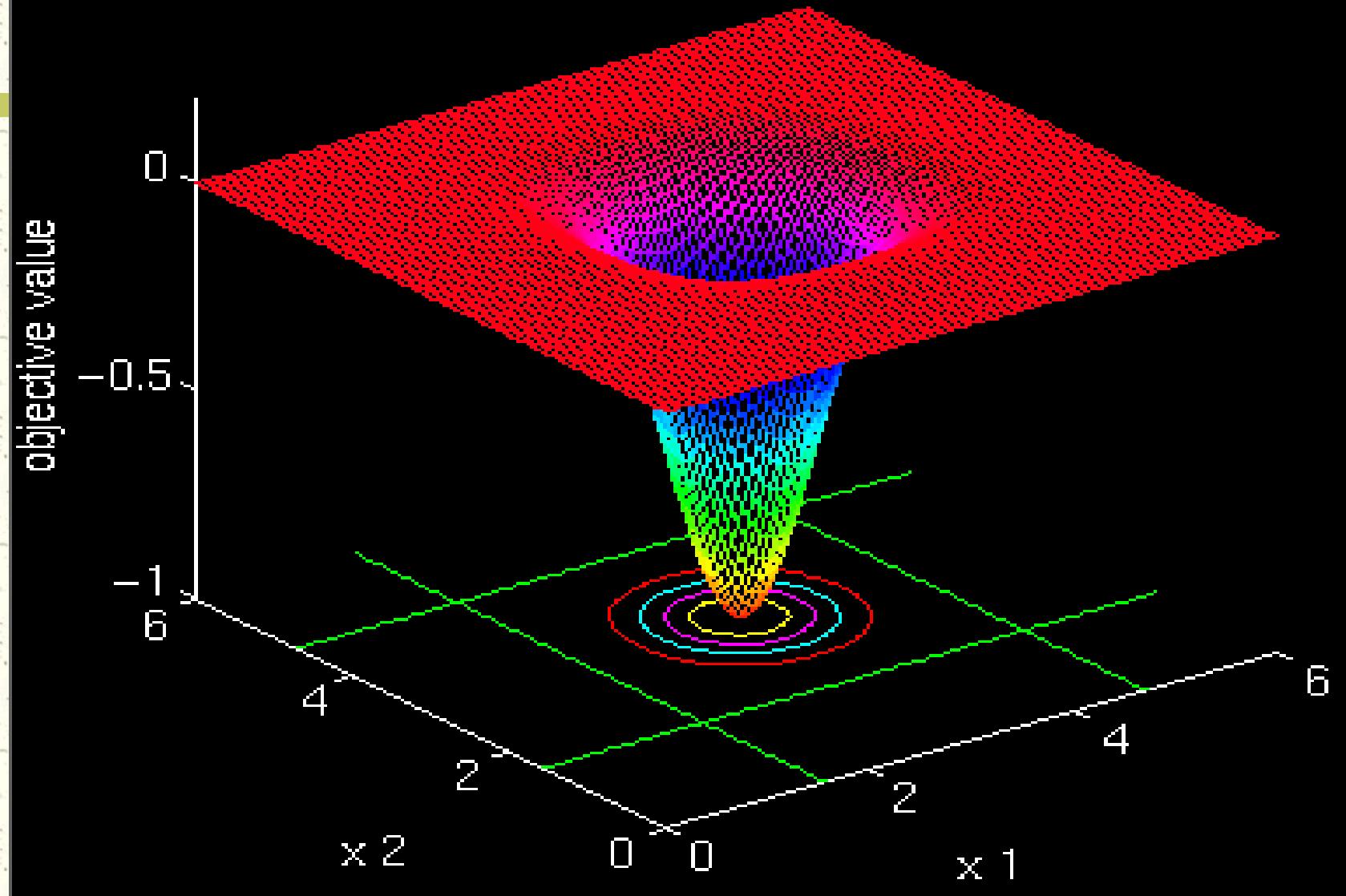


F5

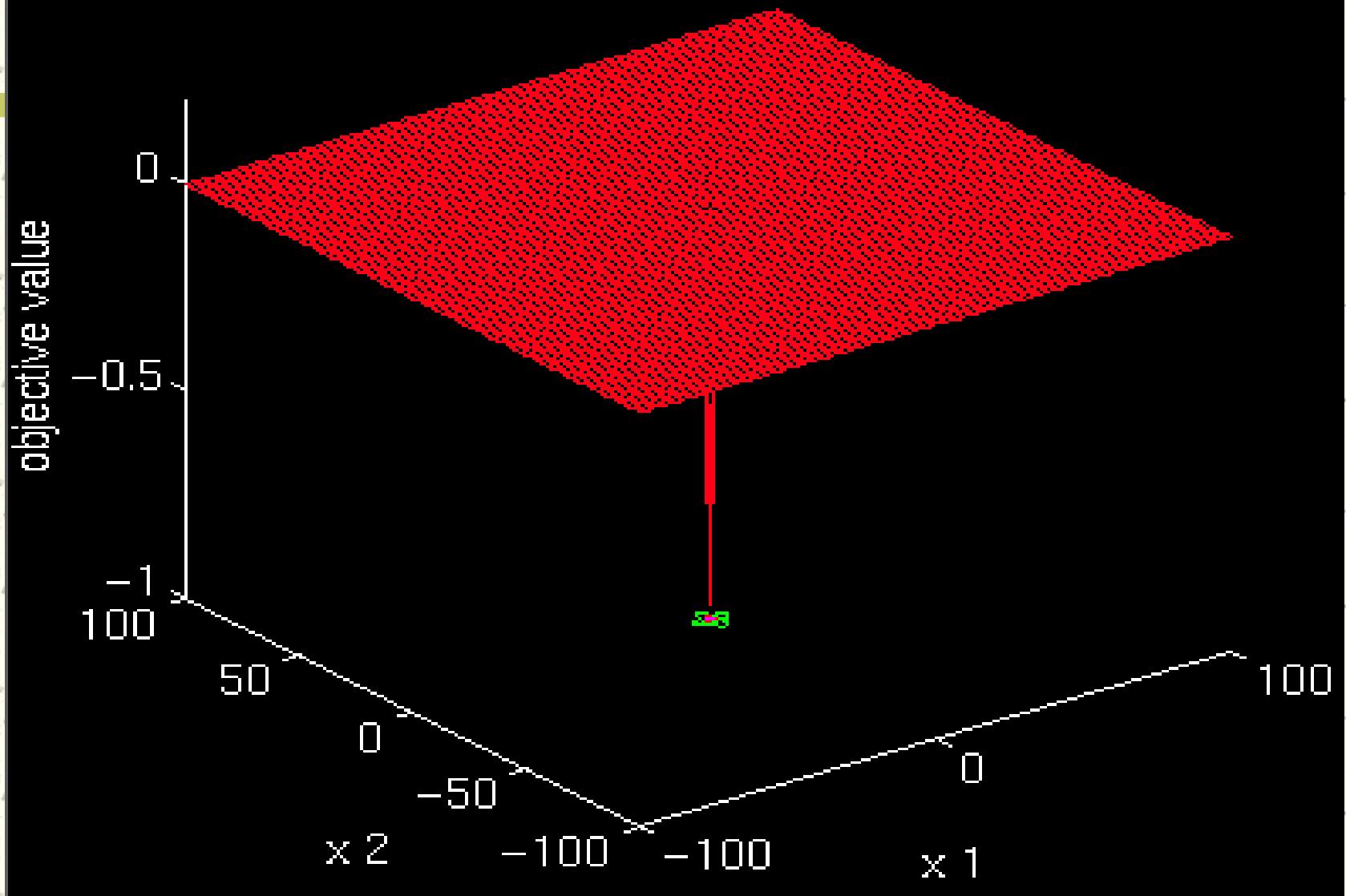
DE JONG function 1



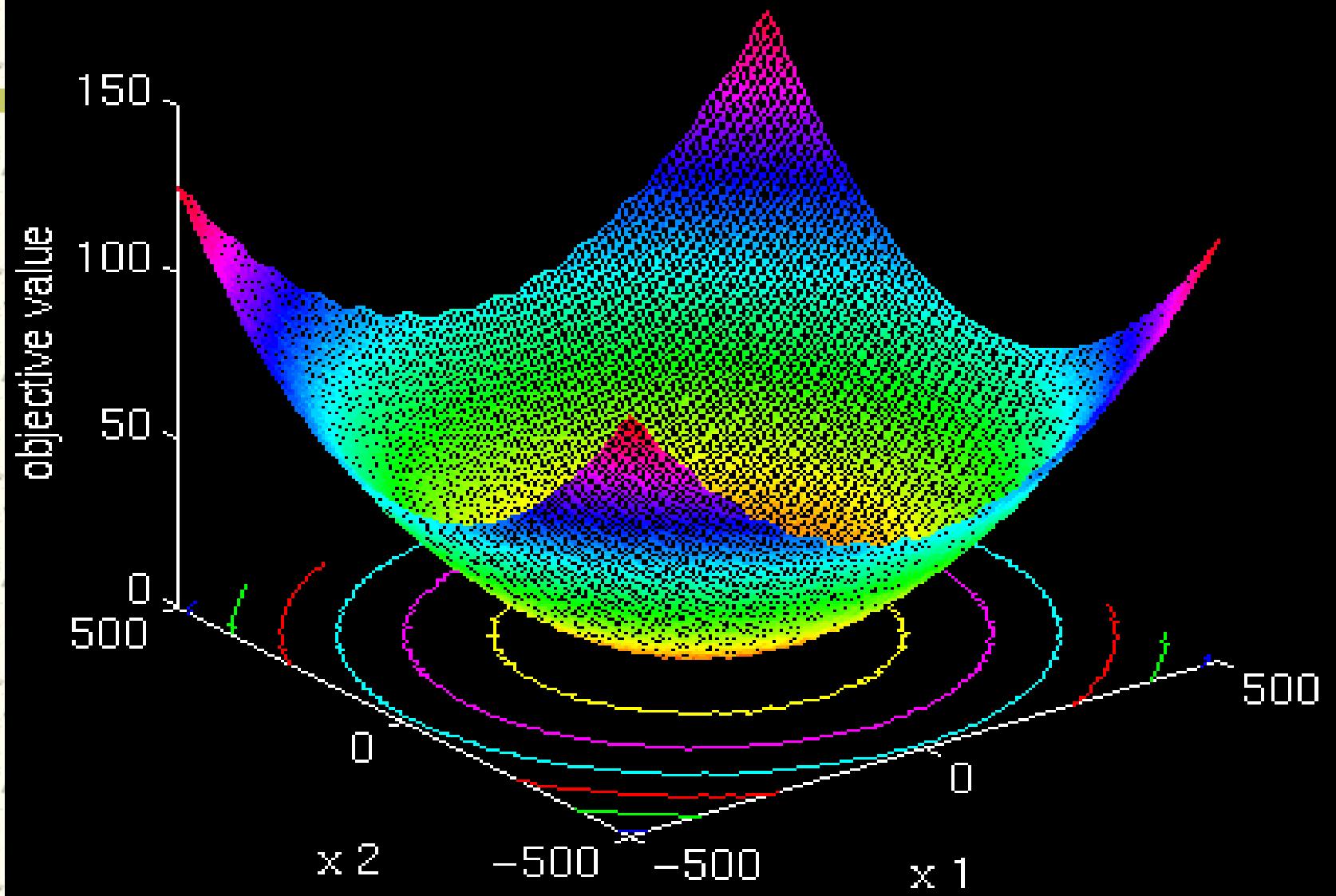
EASOM's function



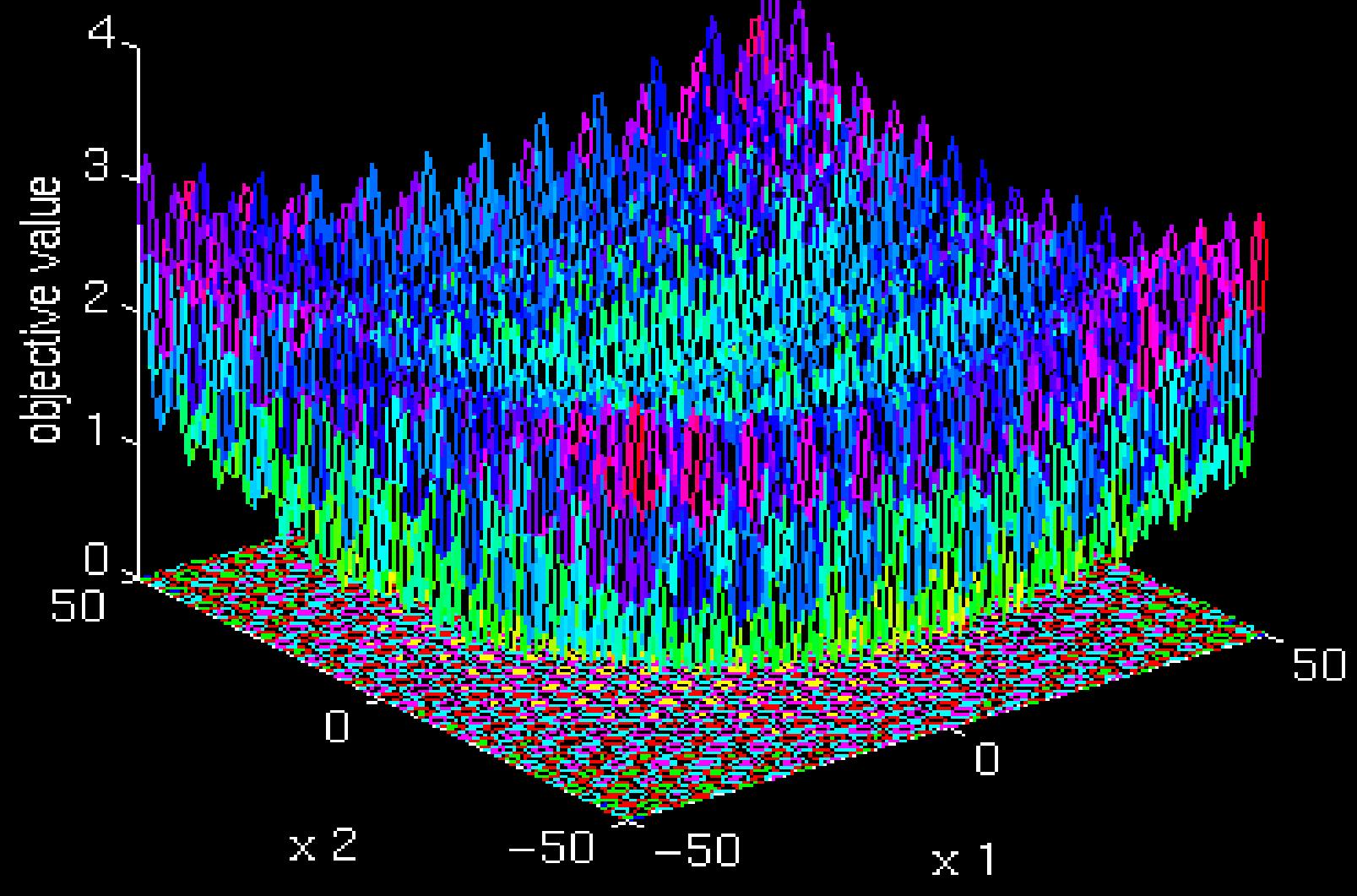
EASOM's function



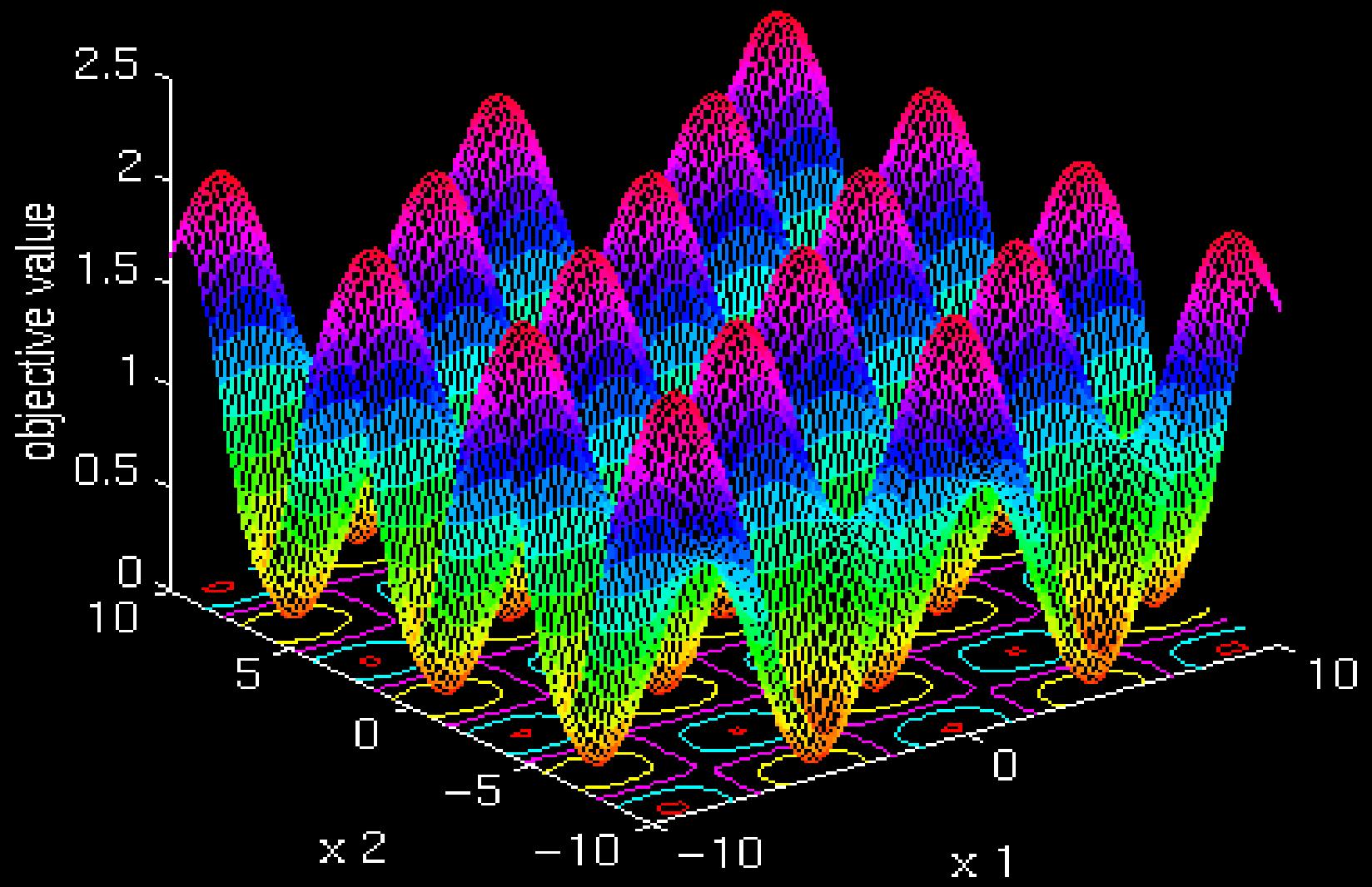
GRIEWANGKs function 8



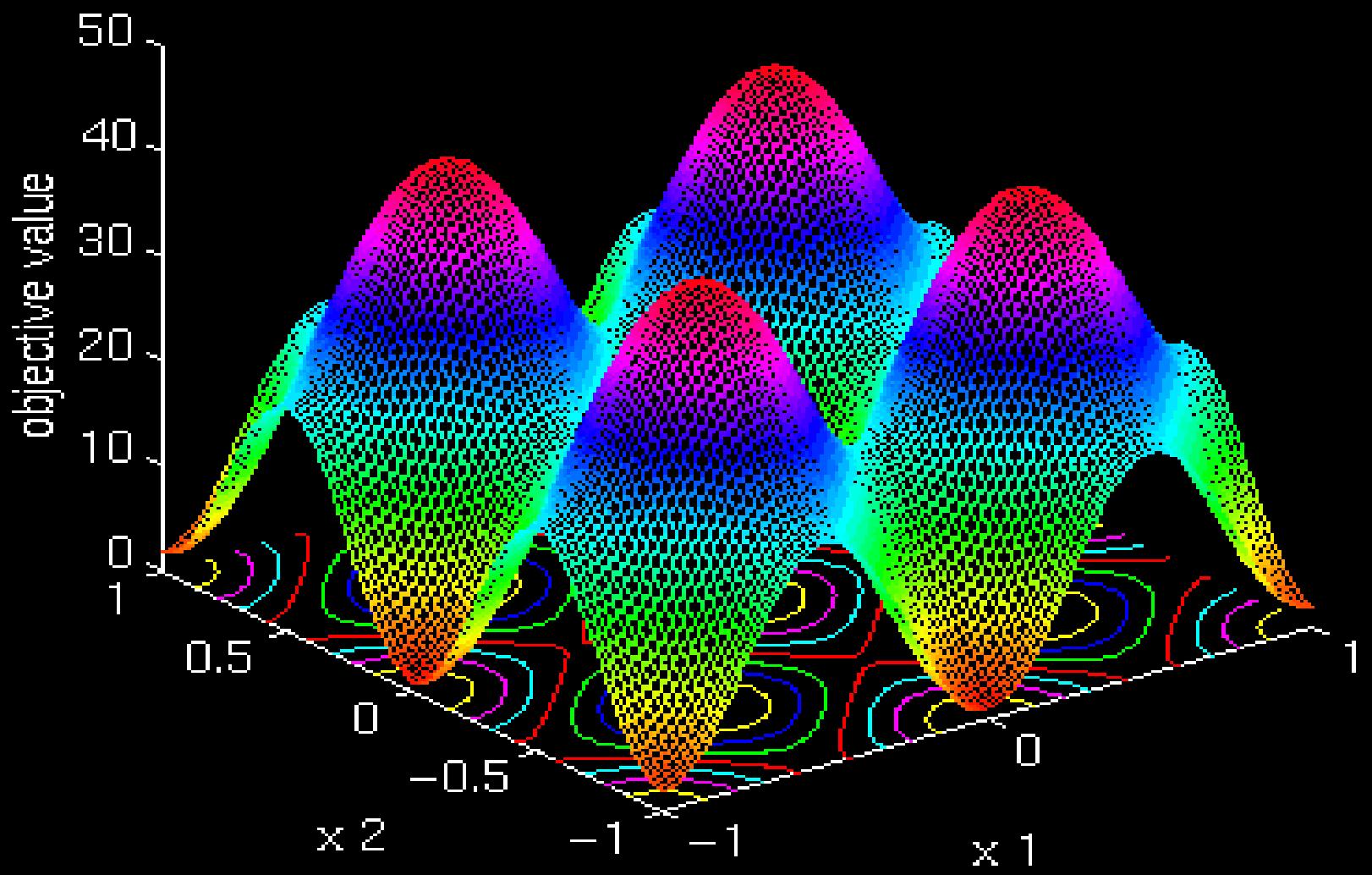
GRIEWANGKs function 8



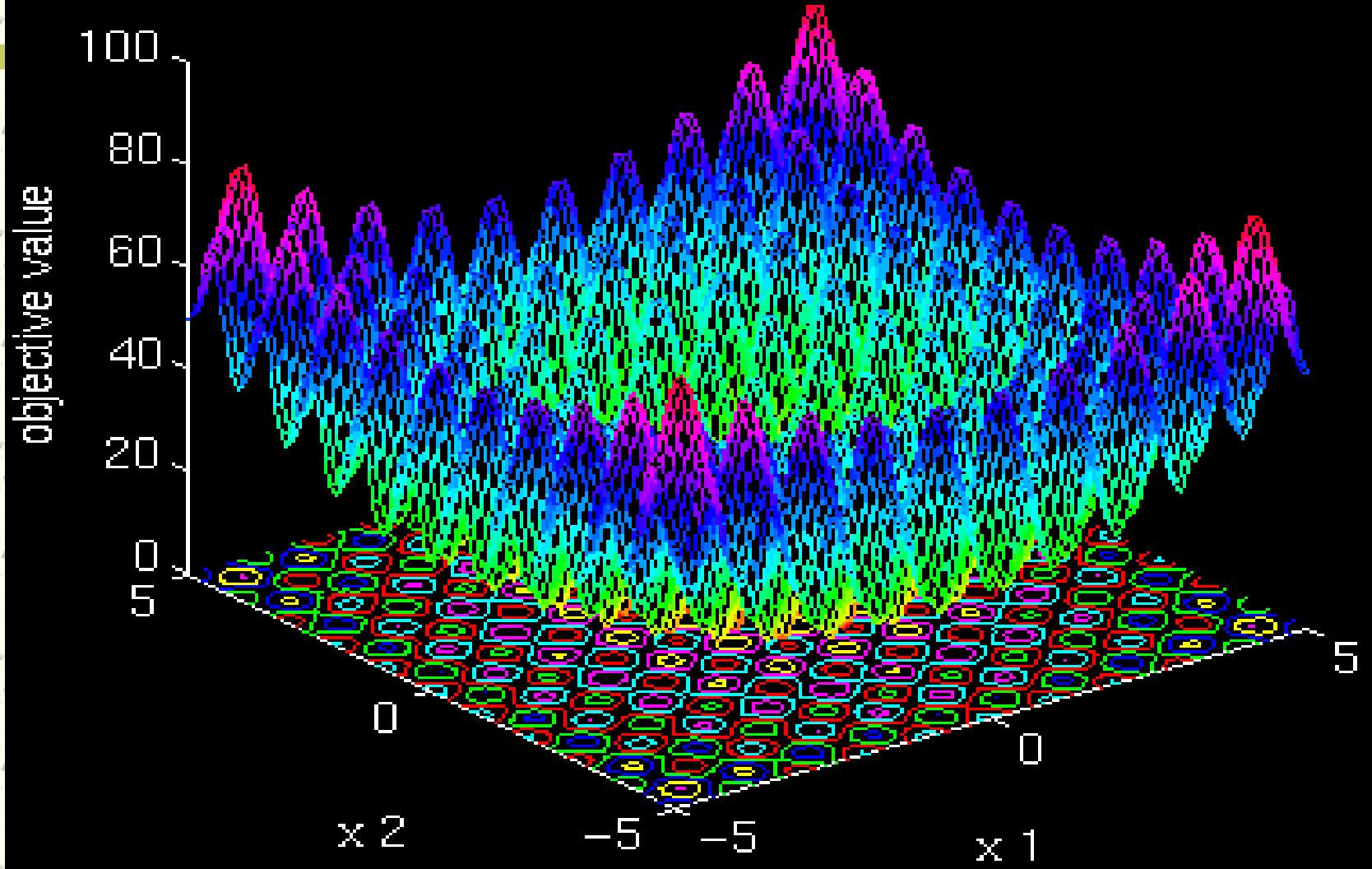
GRIEWANGKs function 8



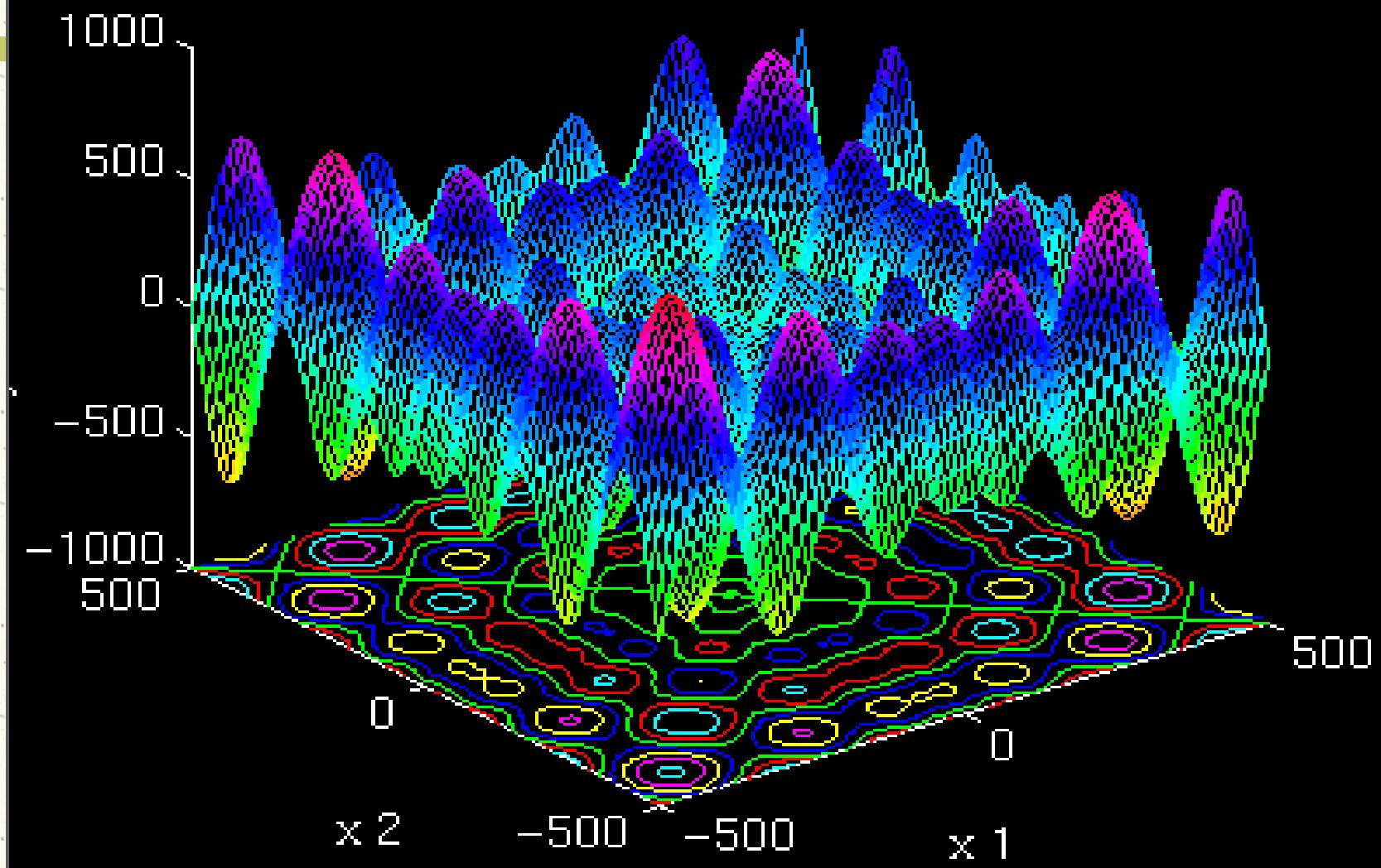
RASTRIGINS function 6



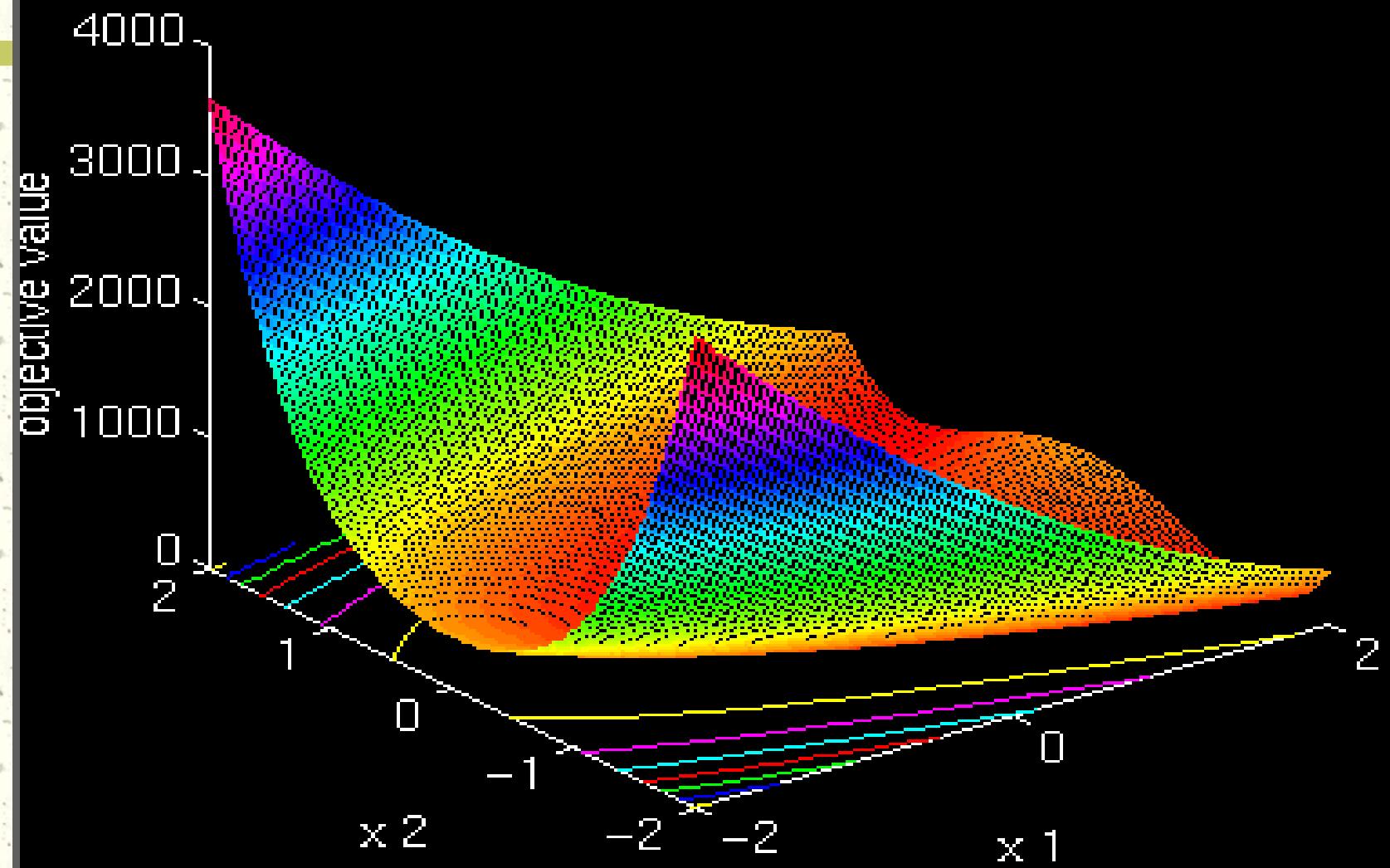
RASTRIGINS function 6



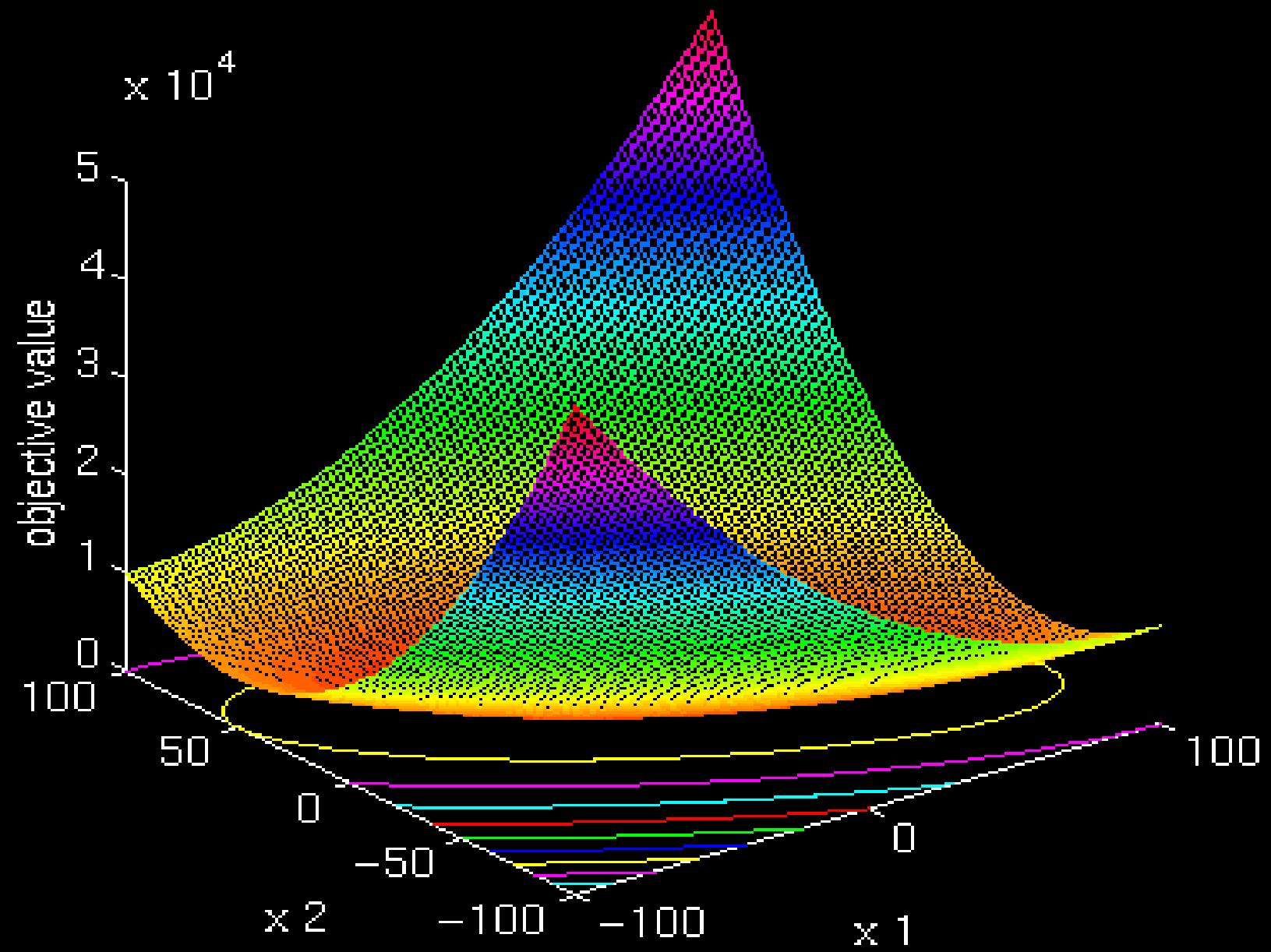
SCHWEFELs function 7



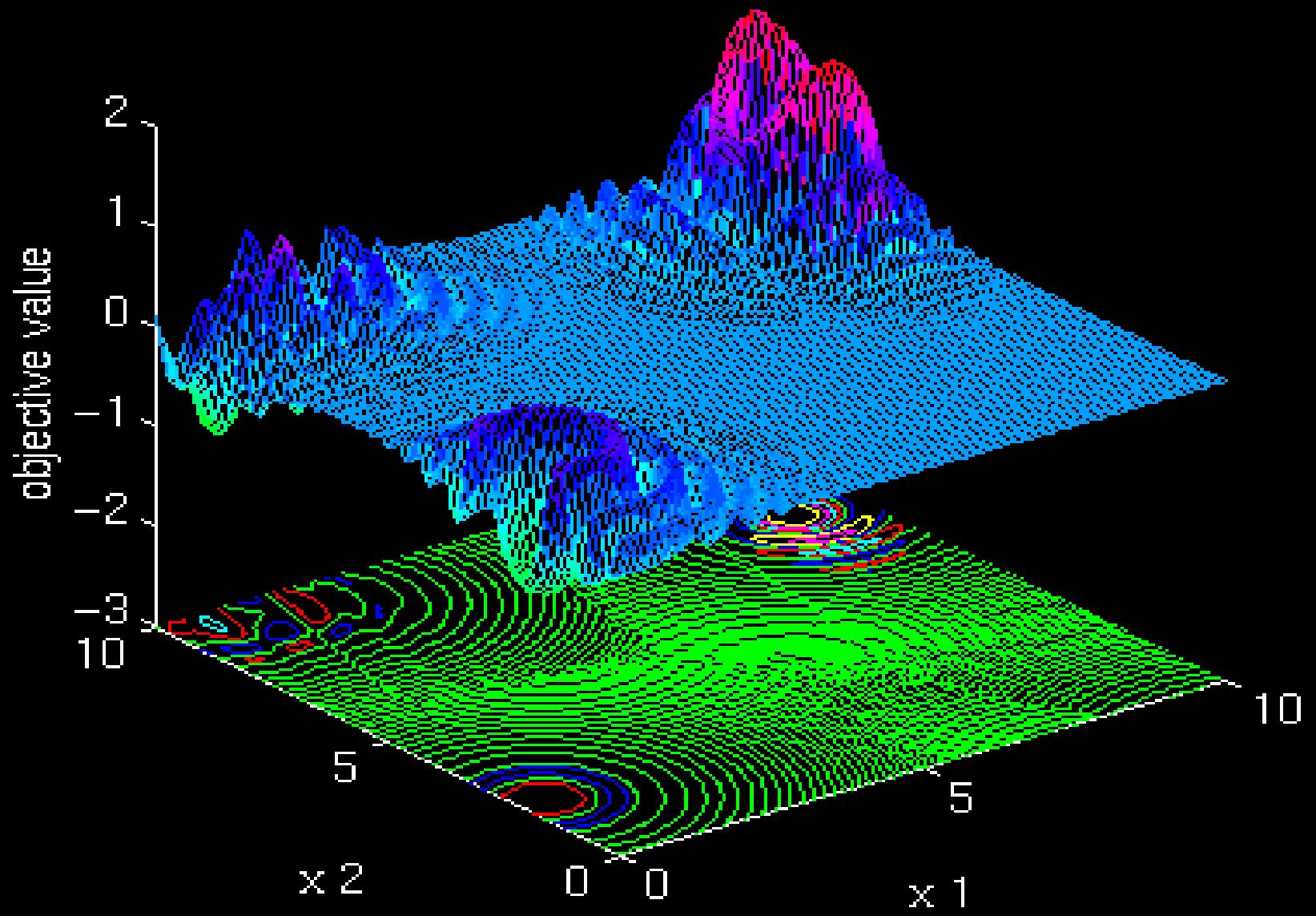
ROSEN BROCK's function 2



Rotated Hyper-Ellipsoid 1b



LANGERMANNs function 11



MICHALEWICZs function 12

