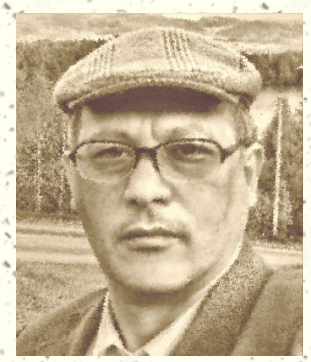
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR

Programa de Pós-Graduação em Engenharia e Informática – CPGEI
Laboratório de Bioinformática e Inteligência Computacional - Curitiba (PR)

Computação Evolucionária



Prof. Heitor Silvério Lopes
hslopes@utfpr.edu.br

CE2018-4

Roteiro para modelagem de problemas

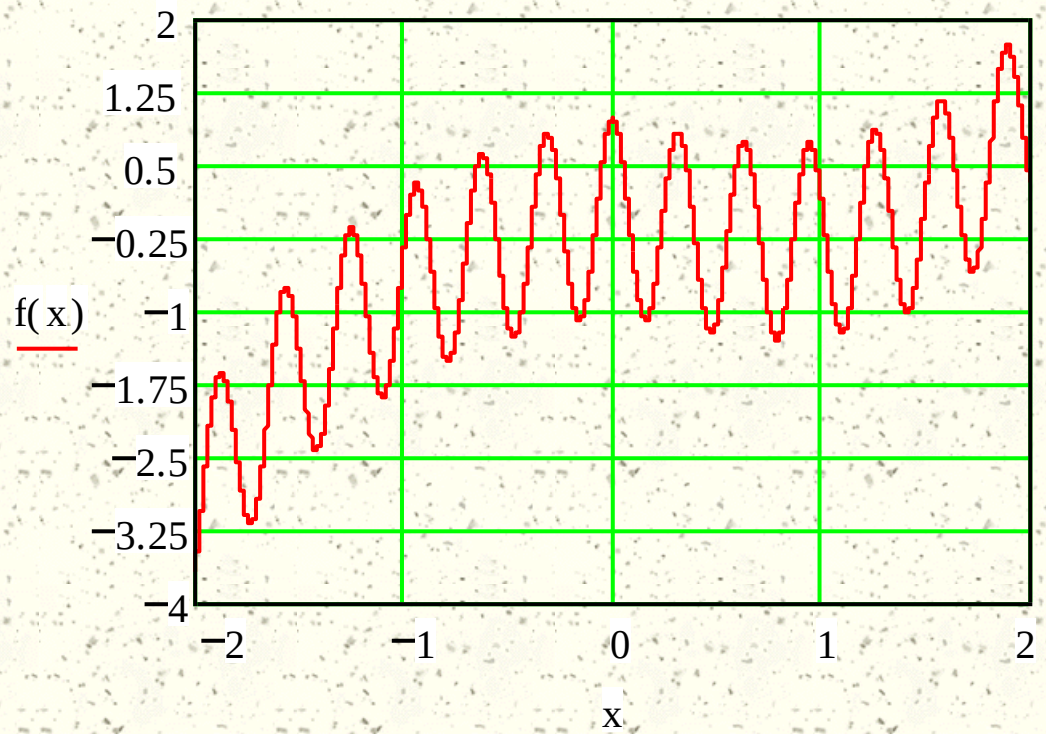
1. Definir a natureza do problema:
   1. Maximização, minimização ou multiobjetivos;
   2. Combinatorial ou não
2. Codificação:
   1. Identificar o conjunto de variáveis e o possível relacionamento entre elas
   2. Definir o formato: binário natural, Gray, inteiro, real ou outro
   3. Definir a precisão necessária de cada variável
   4. Definir o tamanho de cada gene e do cromossomo
3. Decidir se uma busca exaustiva é mais indicada
4. Definir a função objetivo
5. Restrições:
   1. Identificar as restrições aplicáveis às variáveis
   2. Definir como tratar as restrições
   3. Determinar a função e o coeficiente de penalidade
6. Definir a função de *fitness* e sua normalização
7. Definir operadores genéticos especiais que incorporem conhecimento
8. Definir uma estratégia para ajuste dos parâmetros

Exemplo: Maximização de uma função algébrica

$$x^* \mid f(x^*) \geq f(x_i) \forall x_i \subset U$$

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

$$U = [-2, +2]$$



- Esta função tem 12 máximos locais e 1 global no intervalo.

Codificação

- *L* de cada indivíduo depende da precisão desejada (neste caso, 4 casas decimais):
  - 0,0001 x (2-(-2))=40000 pontos
  - $32768 = 2^{15} < 40000 < 2^{16} = 65536$
  - um cromossomo com 1 gene de 16 bits, espaço de busca: $2^{16} =$ **65536**
- Mapeamento genótipo->fenótipo
  - $a = [b_{15}b_{14}...b_2b_1b_0]$ $b_i \in \{0,1\}$
  - $d = \Sigma_{(i=0..15)} b_i 2^i$ valor decimal de "a"
  - mapeamento a->d->x
  - Ajuste de escala:

	a	d	x
mínimo	0000000000000000	0	-2,0000
máximo	1111111111111111	65535	2,0000

$$x = -x_{\min} + \left(\frac{x_{\max} - x_{\min}}{2^{16} - 1} \right) . d$$

Função de *fitness*

- É um problema de maximização de lucro, logo:

$$fitness(x) = \begin{cases} f(x) + C_{\min} & \text{quando } f(x) + C_{\min} > 0 \\ 0 & \text{qualquer outro caso} \end{cases}$$

onde $C_{\min}$ é o módulo do pior caso conhecido

- Resultado:
  - $x^* = 1{,}88929$
  - $f(x^*) = 1{,}73752$

Exemplo #1: Fábrica de garrafas

- Você gerencia uma fábrica de garrafas plásticas que tem apenas uma máquina extrusora. Esta máquina pode funcionar até 60 horas por semana, isto é, 6 dias por semana com jornada de 10 horas por dia. A máquina é capaz de produzir dois tipos de garrafas plásticas: tipo "leite" e tipo "suco". Toda a produção semanal de garrafas plásticas é armazenada temporariamente num depósito. No domingo toda a produção é despachada para os compradores e o depósito é esvaziado completamente.

- A linha de produção leva 6 horas para produzir 100 garrafas tipo leite e 5 horas para produzir 100 garrafas tipo suco. Cada garrafa tipo leite ocupa 10 unidades cúbicas de espaço no depósito, enquanto que a garrafa tipo suco ocupa 20 unidades cúbicas. O depósito tem a capacidade máxima de 15000 unidades cúbicas.

- A contribuição no lucro final da empresa por garrafa tipo leite é 5 unidades monetárias e por garrafa tipo suco é 4,50. O departamento de vendas tem contratos de fornecimento capazes de absorver toda a produção possível de garrafas tipo suco, porém tem compradores somente para 800 garrafas tipo leite por semana.

- Você deve estabelecer qual é o plano de produção mais adequado para maximizar o lucro total da empresa, isto é, quantas garrafas tipo Leite e quantas tipo Suco devem ser produzidas semanalmente.

Exercício #1: Carga do avião

- Um avião de carga tem três compartimentos dianteiro, central e traseiro, cada um com as seguintes capacidades volumétricas e de peso:

Compartimento	Capacidade de peso (tonenada)	Capacidade volumétrica (m3)
Dianteiro	10	6800
Central	16	8700
Traseiro	8	5300

- Para ser mantido o equilíbrio em vôo, a distribuição de carga nos compartimentos do avião deve ser equilibrada nos compartimentos de tal maneira que o peso da carga colocada em cada compartimento seja proporcional a sua capacidade de peso.
- Existem 4 cargas a serem embarcadas no próximo vôo, mostradas na tabela a seguir. Cada carga tem peso e volume característico e dá à empresa que transporta um lucro específico. Qualquer proporção destas cargas podem ser aceitas para transporte.

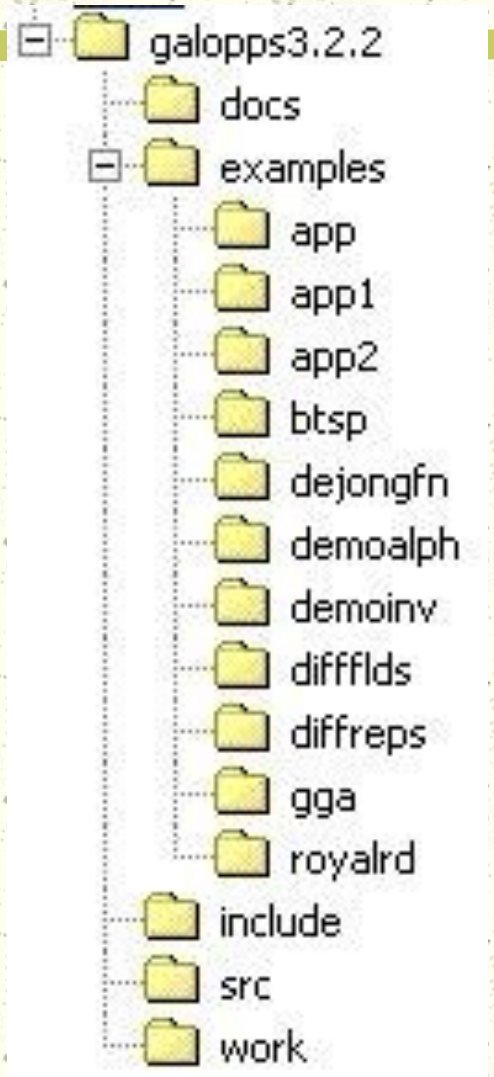
Carga	Peso (tonelada)	Volume (m3/tonelada)	Lucro (R$/tonelada)
1	18	480	310
2	15	650	380
3	23	580	350
4	12	390	285

- O objetivo é determinar o quanto de cada carga (C1, C2, C3, C4) deve ser aceito (se for aceito) e como distribuir a(s) carga(s) ao longo dos compartimentos de modo que o lucro total seja maximizado e ainda seja respeitada a restrição da proporcionalidade dos compartimentos.
- É assumido que: Cada carga pode ser dividida em quantas partes (frações $\geq$ 1kg) for necessário. Cada carga pode ser dividida em dois ou mais compartimentos se for necessário. Quaisquer cargas pode ser colocadas em quaisquer compartimentos.

O *software* GALOPPS

- *Genetic ALgorithm Optimized for Portability and Parallelism System*
- Escrito em ANSI-C, versão atual 3.2.4 - agosto/2002
- Baseado na arquitetura do SGA de Goldberg
- Roda em UNIX e em DOS/Windows
- Simula serialmente paralelismo ou pode-se utilizar uma versão real com PVM
- 5 métodos de seleção, mais elitismo opcional
- Escalonamento: linear, Boltzmann, *sigma truncation*, *window scaling*, *ranking*
- 7 tipos de crossover, 4 de mutação e 2 de inversão
- Medidas de desempenho: *on-line, off-line performance*, mais melhor, média e pior *fitness*
- Vários níveis de relatórios
- Verificação de convergência
- Múltiplas populações

Árvore de pastas do GALLOPS



- **docs**: manual e informações
- **examples**:
  - **app, app1, app2**: exemplos do livro do Goldberg; **btsp**: caixeiro viajante sem ponto de início
  - **dejongfn**: funções F1,F2,F3,F4 de deJong
  - **demoalph**: aplicação utilizando alfabeto não-binário
  - **demoinv**:demonstração do operador de inversão
  - **difffld s**: demonstração do uso de genes de tamanhos diferentes
  - **diffreps**: ilustra diferentes representações em diferentes subpopulações
  - **gga**: problema do empacotamento (*bin packing*)
  - **royalrd**: problema Royal Road de Holland
- **include**: contém os arquivos de *header* (.h)
- **src**: contém os arquivos fonte (.c)
- **work**: diretório de trabalho contendo os arquivos-esqueletos (appxxxxx.c)

Funções importantes do GALLOPS

- **objfunc(critter)**
  - Calcula o *fitness* do indivíduo apontado por critter. Efetivamente o valor do *fitness* é colocado na estrutura: critter->init_fitness
- **app_set_options()**
  - Atribui valores a variáveis globais importantes:
    - stochastic, elitism, user_supplied_initialization, using_inversion
- **app_set_field_sizes()**
  - Chamada se alpha_size < 2 significando que os campos (genes) têm tamanhos diferentes. O usuário deverá declarar o tamanho de cada gene nas variáveis correspondentes, p.ex. field_sizes[0] = 24

Funções importantes do GALLOPS

- **application()**
  - Contém procedimentos dependentes da aplicação a serem executados a cada geração
- **app_init()**
  - Inicialização dependente do problema chamada no início da rodada
- **app_initreport()**
  - Relatório inicial (antes de rodar o AG), dependente da aplicação
- **app_report()**
  - Relatório dependente da aplicação a cada geração
- **app_quiet_report()**
  - Relatório executado quando quiet < 3

Funções importantes do GALLOPS

- **app_generate()**
  - Relatório dependente da aplicação no final de cada geração
- **app_stats(pop)**
  - Estatísticas da população a critério do usuário

Arquivo de parâmetros do AG

```
numberofruns = 1
quiet = 0
ckptfreq = 10
checkptfileprefix =
restartfileprefix =
permproblem = n
alpha_size = 2
numfields = 10
superuniform = n
maxgen = 200
```

```
/* parameters for first run */
popsize = 8
printstrings = y
pcross = .5
pmutation = .0
pinversion = 0.
autocontrol_selection = n
beta =
scaling_window = -1
sigma_trunc = 0.
scalemult = 1.3
crowding_factor = 0
conv_sigma_coeff = .5
convinterval = 5
randomseed = .123
```

Arquivo Makefile

- As alterações devem ser feitas no Makefile do diretório work e no do diretório src

```
OPT = -O3
CC=gcc $(OPT)
```

- Selecionar um dentre os métodos de seleção:

```
SELECT  = suselect.c  -- Stochastic Universal Sampling
#SELECT = rselect.c   -- roulette wheel
#SELECT = srselect.c  -- stochastic remainder
#SELECT = tselect.c   -- tournament selection
#SELECT = rnkslect.c  -- Rank-based selection
```

- Selecionar um dentre os operadores de *crossover*:

```
CROSSOVER = twoptx.c  -- two-point crossover
#CROSSOVER = oneptx.c  -- one-point crossover
#CROSSOVER = uobx.c  -- uniform order-based crossover
#CROSSOVER = ox.c  -- order crossover
#CROSSOVER = cx.c  -- cycle crossover
#CROSSOVER = pmx.c  -- partially matched crossover
#CROSSOVER = unifx.c  -- ??
```

- Selecionar um dentre os operadores de mutação:

```
MUTATION = multimut.c -- ??
#MUTATION = bitmutat.c  -- single-bit mutation
#MUTATION = scramble.c  -- random Sublist Scramble mutation
#MUTATION  = swap.c  -- ??
```

- Colocar as rotinas específicas da sua aplicação aqui:

```
APP = app.c
```

IMPORTANTE !

- LEIA O MANUAL ! RTFM !
  - É muito mal escrito, mas é a única documentação disponível
  - Tem uma versão em pdf na página da disciplina