rECGA

Rafael Luiz Klaser
Set/2012

Contextualização

EDAs:

- partem de uma modelagem binária da solução;
- buscam uma estimação de distribuição (dos bits da solução).

Problemas de Otimização:

- Funções reais de variáveis reais.

rECGA: Uma solução para aplicar EDA em funções de variáveis reais.

Contextualização

CGA:

- Uma nova população não é gerada por crossover e sim a partir de uma distribuição de probabilidade (estimada a partir da própria população selecionada)

Contextualização

EGCA:

- building blocks (Linkeage Learning)

- agregar partes da solução em blocos, sendo a probabilidade destes blocos não mais a probabilidade individual de cada bit e sim das possíveis combinações destes bits.

- Complexidade Combinada: Complexidade + Compressão (Entropia)

rECGA

Aplicar o ECGA para problemas de variávies reais

Apresentação da ideia proposta em:

C.-H. Chen, W.-N. Liu, and Y.-P. Chen. Adaptive discretization for probabilistic model building genetic algorithms. In GECCO **'06: Proceedings of the 8th annual conference on Genetic and evolutionary computation**, pages 1103–1110, New York, NY, USA, 2006. ACM Press.

rECGA

Algoritmo básico de um EDA

- Gera população
- Avalia
- Seleciona
- Distribuição de probabilidade
- Loop

(e, em algum ponto alguma condição de parada)

rECGA

Alterações no rECGA

- Gera população
- Avalia
- **Melhora** (Simplex [1] a cada N Gerações)
- Seleciona
- **Codifica** (SoD)
- **Distribuição de probabilidade** (ECGA)
- Loop

(e, em algum ponto alguma condição de parada)

[1] J. Nelder and R. Mead. A simplex method for function minimization. **Computer Journal** , 7:308–315, 1965.

rECGA

Melhora

- Método Nelder-Mead (Simplex)
- GSL[1]: *gsl_mulmin_fminimizer_nmsimplex*

Motivação

- Como as variáveis não são binárias como no EDA clássico, a busca por uma solução no espaço real discretizado (puramente pelo modelo de distribuição) convergiria de forma inadequada.

1 **GNU Scientific Library** - www.gnu.org/software/gsl

rECGA

Melhora

- A cada N gerações, aplica a otimização clássica em P % melhores indivíduos.

Objetivos

- Velocidade da convergência;
- Mantém o split rate para a busca global, a otimização clássica para a busca local;
- *Fine graining*.

rECGA

Codifica

- A ideia do rECGA é "bin"arizar o espaço de busca, ou seja criar um histograma;
- Mesmo princípio utilizado em abordagens anteriores como o FHH (*Fixed-Height Histogram*) e o FWH (*Fixed-Width Histogram*) [2]

- Proposta: SoD (*Split-On-Demand*)

[2] S. Tsutsui, M. Pelikan, and D. E. Goldberg. Evolutionary algorithm using marginal histogram models in continuous domain. **IlliGAL Report No. 2001019**, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL, 2001.

rECGA

SoD (*Split-On-Demand*)

- discretizar o domínio real baseada em uma taxa de divisão (*split rate*) sendo gradativamente alterada (*decrease rate*) ao longo das gerações;

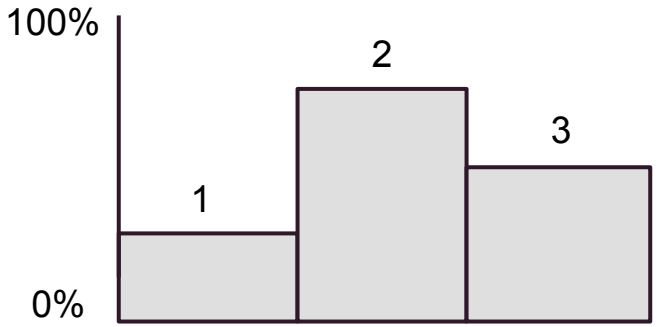
- Ex:

- *split rate* inicial = 50%
- *decrease* factor = 98%
- *rate* próxima geração = 0.5 * 0.98 = 49%

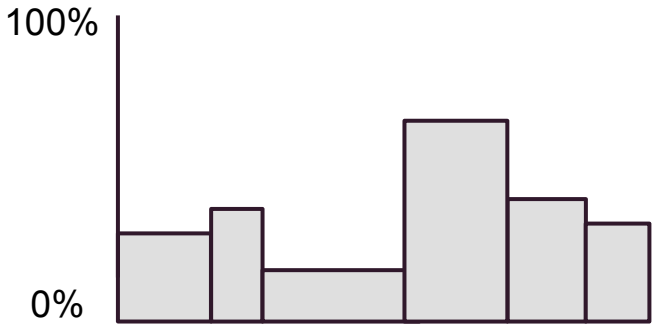
rECGA

SoD - Discretização do domínio real e percentagem da população em cada grupo:

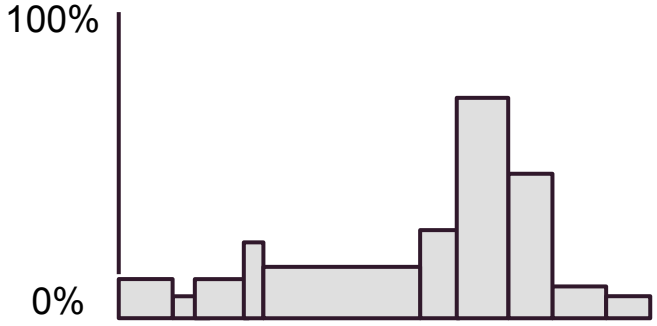
As figuras não estão em escala, nem representam uma execução real, apenas ilustrativo do efeito do SoD.



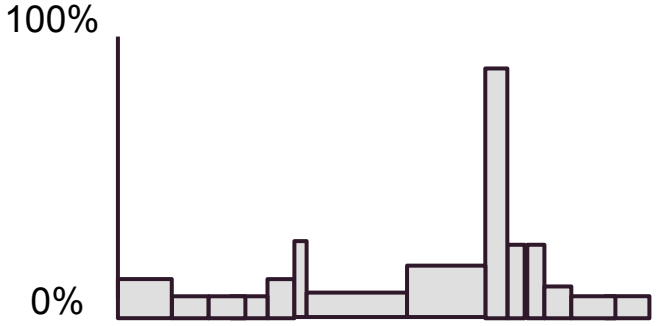
Split Rate 50% Geracao 1



Split Rate 40% Geracao 5



Split Rate 30% Geracao 10

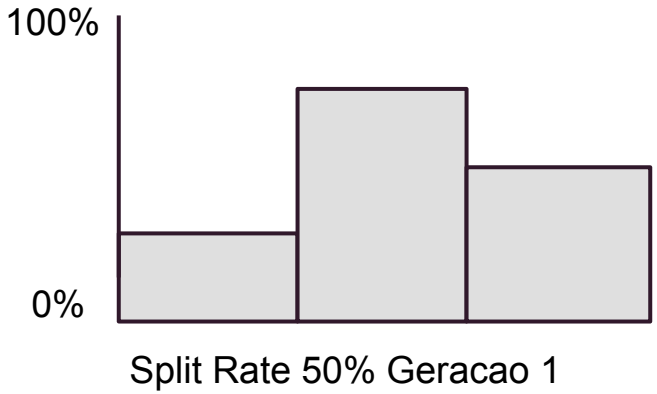


Split Rate 20% Geracao 15

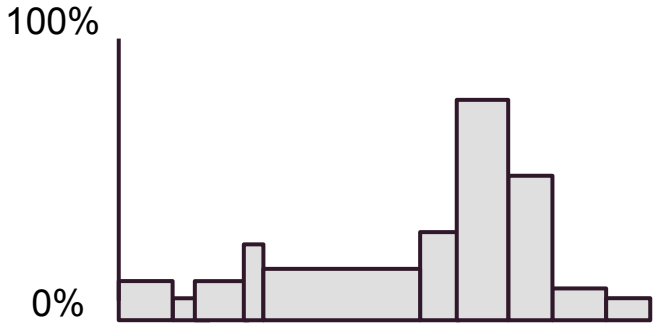
rECGA

SoD - Discretização do domínio real e percentagem da população em cada grupo:

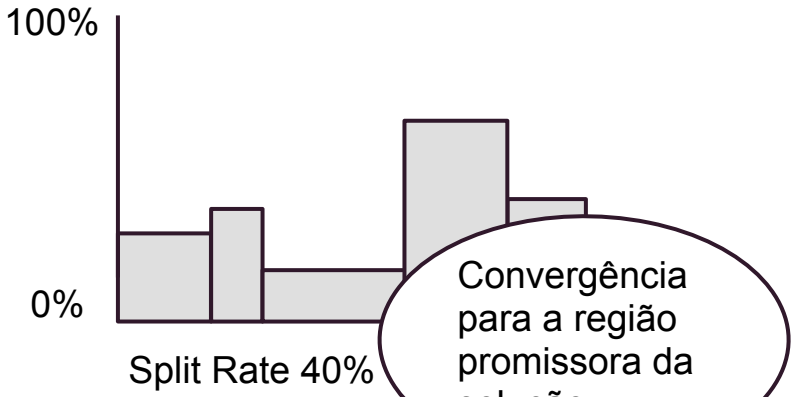
As figuras não estão em escala, nem representam uma execução real, apenas ilustrativo do efeito do SoD.



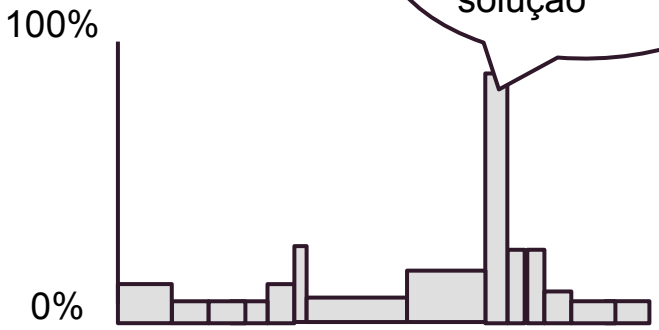
Split Rate 50% Geracao 1



Split Rate 30% Geracao 10



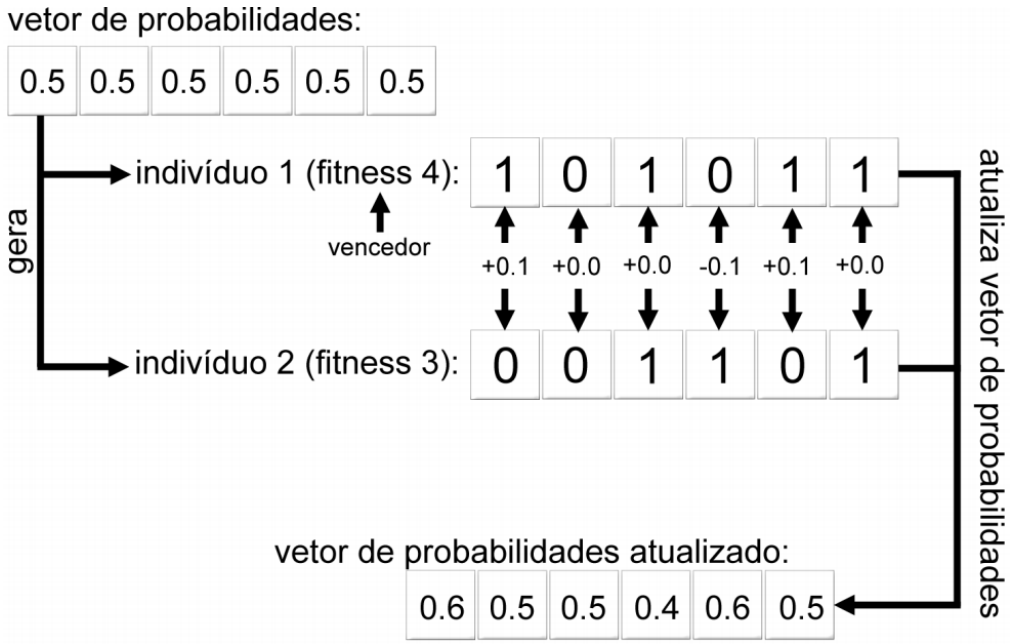
Split Rate 40%



Split Rate 20% Geracao 15

rECGA

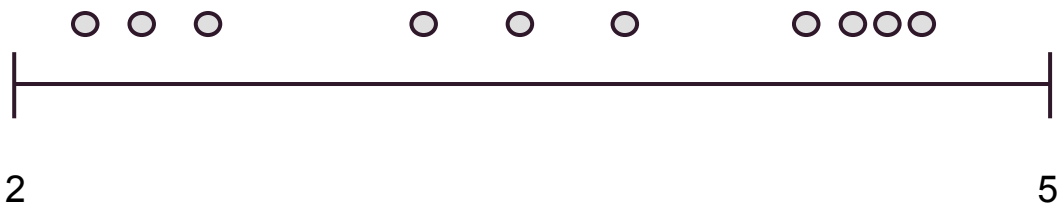
- A representação não precisa ser binária e sim discreta;
- Pois, relembrando o CGA, no final o que interessa é o vetor de probabilidades (discreto).



rECGA

SoD em ação

Split-Rate na geração atual = 50%



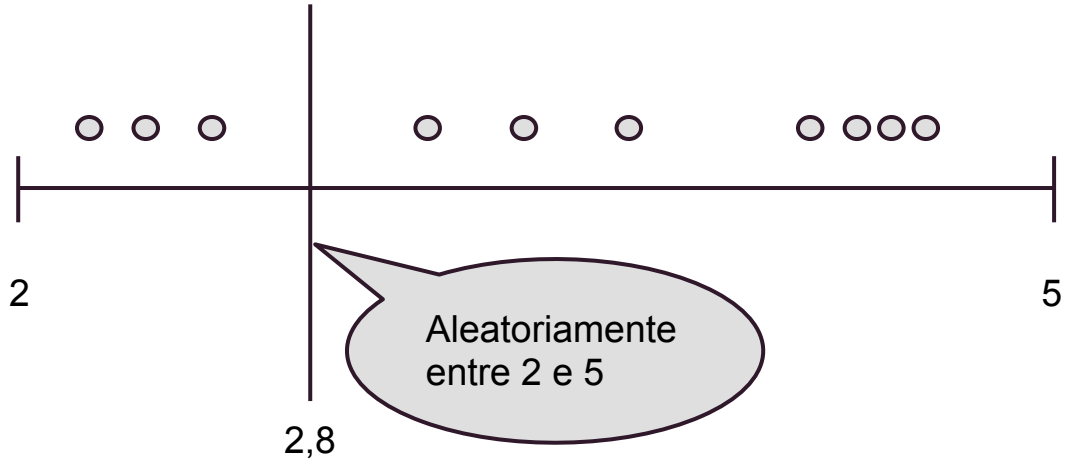
População inicial (10) distribuída no intervalo real [2; 5]

PS. Implementação: utiliza a distribuição uniforme para gerar a população

rECGA

SoD em ação

Split-Rate na geração atual = 50%



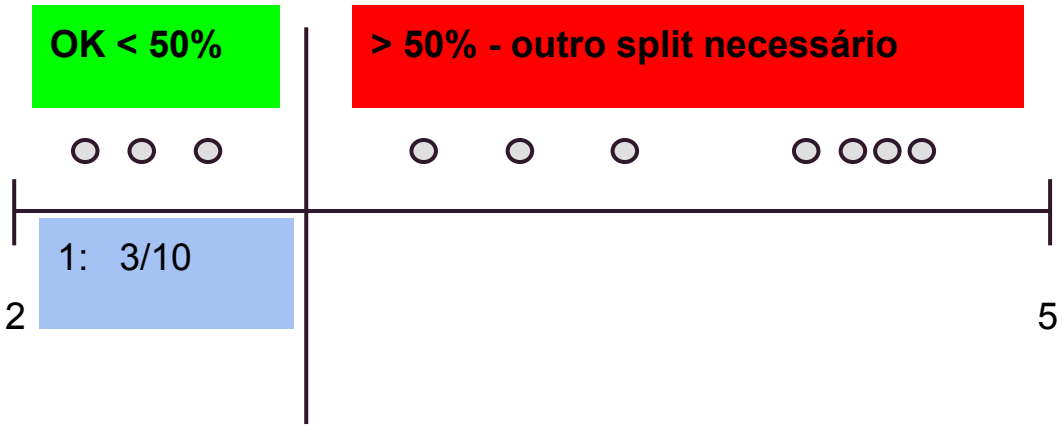
População inicial (10) distribuída no intervalo real [2; 5]

PS. Implementação: utiliza a distribuição uniforme para gerar o corte

rECGA

SoD em ação

Split-Rate na geração atual = 50%



População inicial (10) distribuída no intervalo real [2; 5]

rECGA

SoD em ação

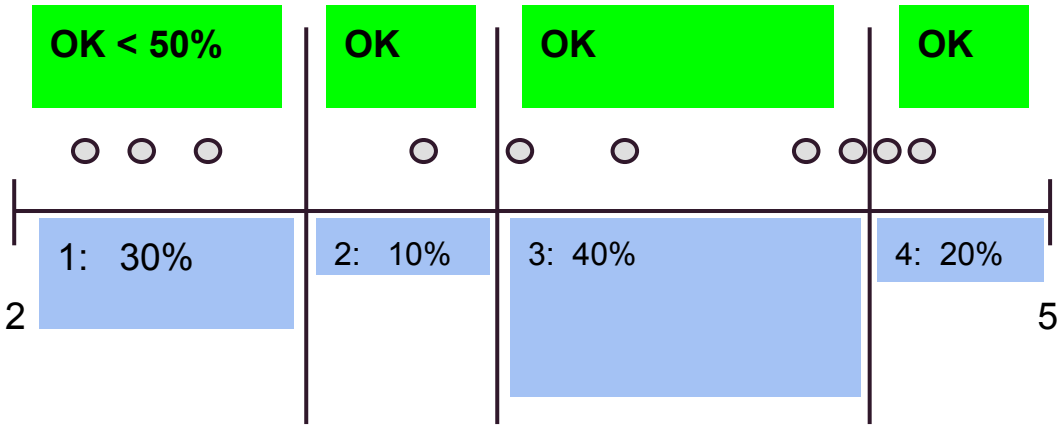
Split-Rate na geração atual = 50%

População inicial (10) distribuída no intervalo real [2; 5]

rECGA

SoD em ação

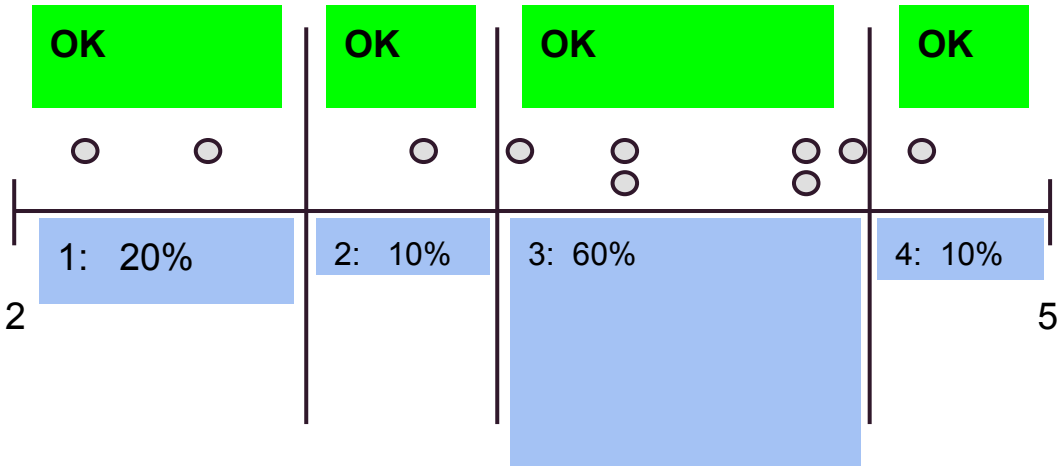
Split-Rate na geração atual = 50%



População inicial (10) distribuída no intervalo real [2; 5]

rECGA

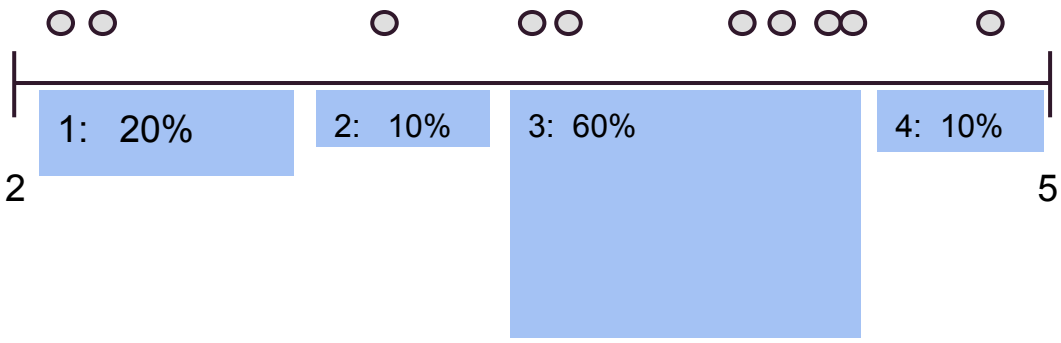
Seleção



Torneio de N: Os melhores são representados mais de uma vez enquanto os piores desaparecem.

rECGA

Nova população



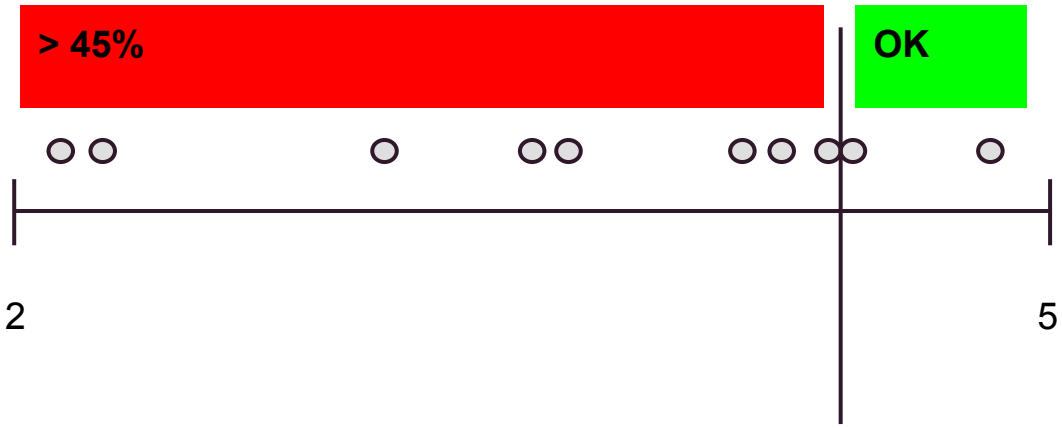
Gerada em cima da nova distribuição

PS. Implementação: 10% (dentre os melhores) recebem um "melhoramento" após a geração da população.

rECGA

SoD próxima geração

Split-Rate na geração atual = 45%



Novos splits são gerados de acordo com as regras (Split Rate), as divisões da geração passada não interessam mais.

rECGA

- Simples assim!?

rECGA

- Simples assim!? Não.

- Até aqui foi apresentado a parte "CGA" do algoritmo, ainda é necessário passar para o ECGA.

- Dúvidas até aqui...?

rECGA

Agora complica... (o artigo não menciona os detalhes, vamos ao código)

> *... In rECGA we use SoD to encode each dimension of the individuals in the current population after tournament selection and do the MPM greedy search as in ECGA. [1]*

rECGA

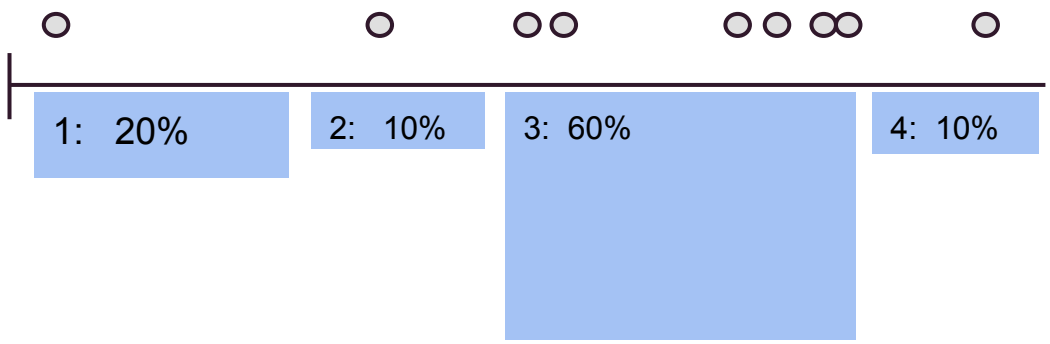
ECGA -> MPM (Marginal Probability Model)

Como calcular a complexidade combinada a partir do hitograma gerado pelo SoD:

- Complexidade do modelo: **log(N+1) * (B-1)**
- Compressão: **N * Σ E(M_I)**
    - N: tamanho da população
    - B: número de bins do histograma
    - E(M_I): Entropia (-$prob_{bin}$ * log($prob_{bin}$))

rECGA

Cálculo da complexidade combinada



Complexidade = log(10+1)*(4-1) = 3,459*3 = **10,378**

Compressão = 10*((-0,2*log(0,2))+(-0,1*log(0,1)+ (-0,6*log(0,6)+(-0,1*log(0,1))=10*1,8657 = **18,657**

Complexidade Combinada = 29,035

rECGA

Relembrando o SoD

Não parece fazer sentido combinar os *bins*

rECGA

- o rECGA é para funções multivariadas

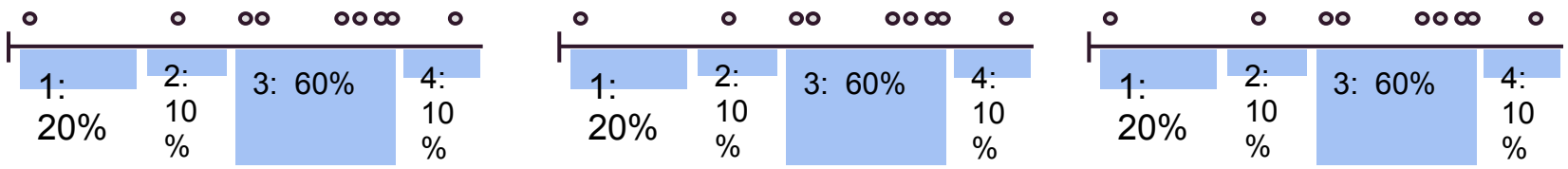
rECGA

Funções multivariadas

- Para cada dimensão da função é efetuado um SoD independente;
- É calculado a complexidade combinada de cada dimensão;
- O MPM ocorre entre as dimensões.

rECGA

Exemplo: min $y = f(x_1, x_2, x_3)$



X_1 X_2 X_3

O objetivo do MPM é buscar a relação entre as variáveis (x_1 pode ser relacionado com x_3 ou são independentes?)

rECGA

Busca Gulosa

x_1

$CC_{x1} = 18,45$

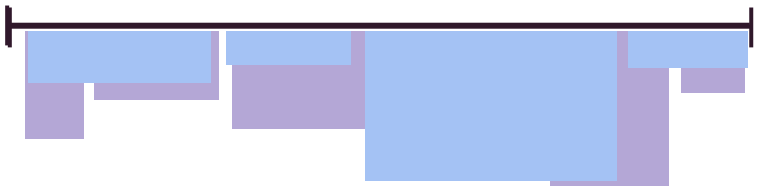
x_3

$CC_{x3} = 31,07$

rECGA

Testa a redução da complexidade combinada

X_1 - X_3



$CC_{x1-x3} = 47,98$

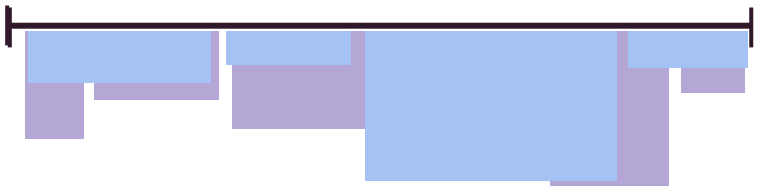
$CC_{x1-x3} < CC_{x1} + CC_{x2}$?

$CC_{x1} + CC_{x2} = 18,45 + 31,07 = 49,52$ **Agrupa!**

rECGA

Testa a redução da complexidade combinada

X_1 - X_3



$CC_{x1-x3} = 47,98$

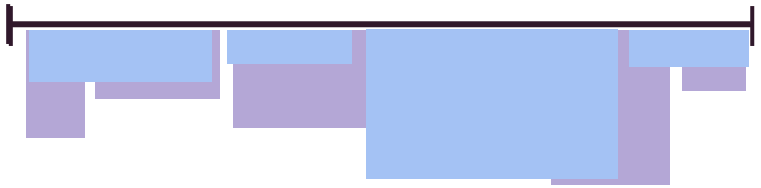
$CC_{x1-x3} < CC_{x1} + CC_{x2}$?

$CC_{x1} + CC_{x2} = 18,45 + 31,07 = 49,52$ **Agrupa!**

rECGA

Merge?

X_1 - X_3



Quantidade de *bins* do novo histograma:

- produto cartesiano entre os histogramas
- *No exemplo: 4 * 5 = 20*

Probabilidades nos novos *bins*:

- Contabiliza os individuos de acordo com uma codificação: Σ_i **schema**$_i$ * **index**$_i$

rECGA

Codificação:

- i = 2 (merge de 2 conjuntos)

- $schema_i$ -> codigo do *bin* do alelo

Ex. alelo x_1=3, alelo x_3=4

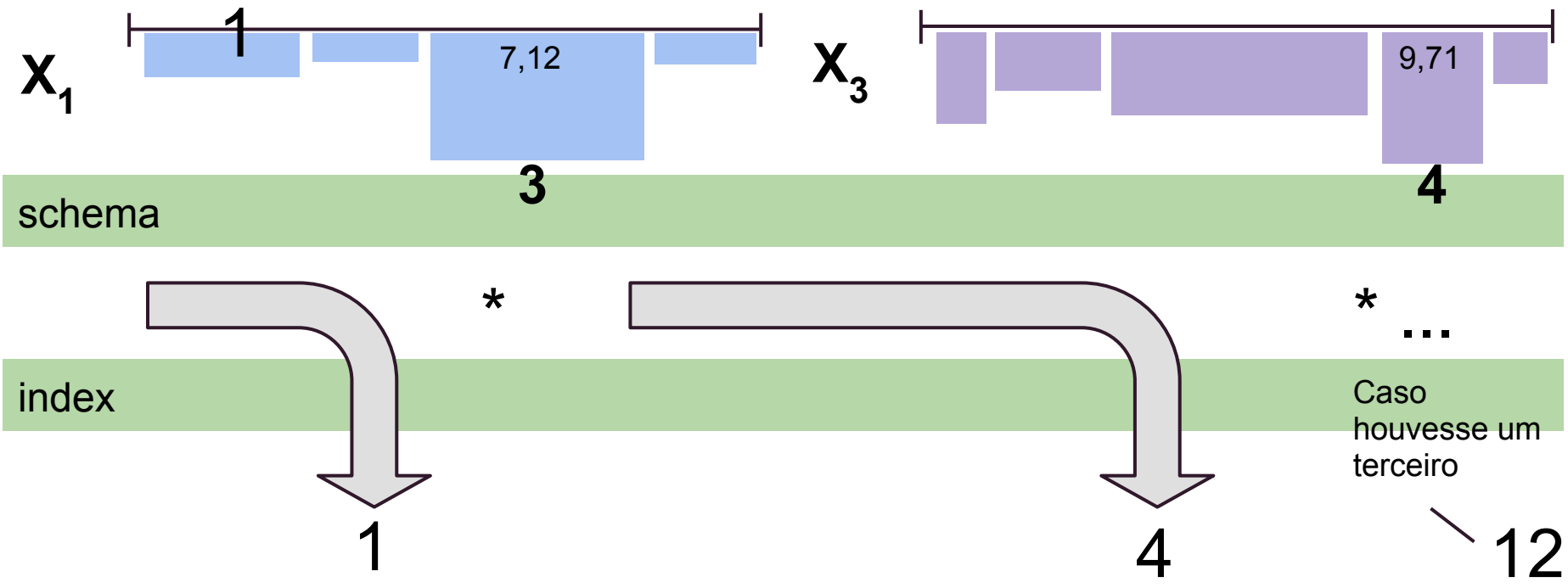
- $index_i$ -> na ordem do alelo (x_1, x_3), produto cartesiano acumulado

Ex. index[0]=1, index[1]=4 (bins do alelo x_1)

rECGA

Decodificação, para i = 2

y = func(7,12; ... ; 9,71; ... ; ...) : individuo



O individuo 1 pertence ao *bin*: (3*1)+(4*4) = 19

rECGA

- Com este *merge,* é calculada então a complexidade combinada dos genes unidos;

- Caso for menor que a soma das complexidades dos genes separados, se passa a considerar unidos;
(utiliza a estratégia gulosa para selecionar a melhor união)

- A nova distribuição (unida) é repassada para cada gene (do grupo) para gerar a nova população.

rECGA

Nova distribuição para os genes unidos:

- para cada novo agrupamento, identifica o menor individuo e o maior individuo e cria-se um intervalo

- os possíveis "gaps" são excluidos unindo-se uma metade para cada intervalo vizinho

-

- a nova distribuição é copiada igualmente para cada gene

rECGA

- Simples assim? Não.

- O algoritmo guarda os melhores, somente gera uma percentagem da nova população - parâmetro: *probability_crossover*