UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba (PR)

LABIC
Laboratório de Bioinformática e Inteligência Computacional - UTFPR
http://labic.utfpr.edu.br

Computação Evolucionária

Prof. Heitor Silvério Lopes
hslopes@utfpr.edu.br
http://silverio.net.br/heitor

class4 (2020)

Fundamentos teóricos de AG

1. AGs paralelos: motivação, tipos e topologias
2. AGs híbridos, iteração entre aprendizado e evolução
3. Algoritmos meméticos
4. Ajuste dos parâmetros de controle

1. Algoritmos genéticos paralelos

- **"In a world where serial algorithms are usually made parallel through countless tricks and contortions, it is no small irony that genetic algorithms (highly parallel algorithms) are made serial through equally unnatural tricks and turns"**

GOLDBERG, D.E. (1989, p. 208)

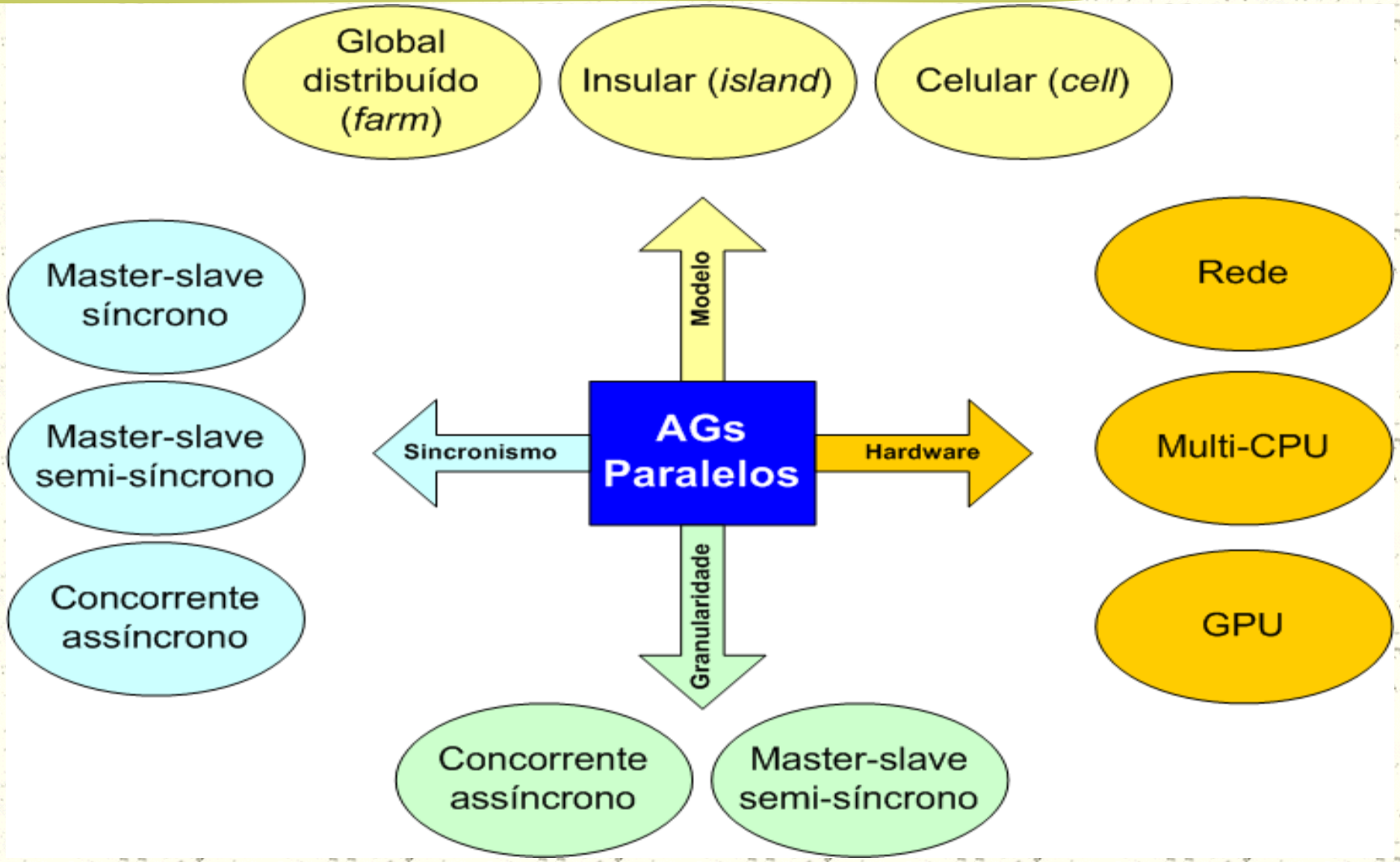
1. Tipos de paralelismo em AGs

- Paralelismo implícito:
  - AGs mantém λ indivíduos a cada geração, mas processam λ^3 schematas (trechos de *strings*).
  - De maneira simplificada é a capacidade do algoritmo de processar muitas informações simultaneamente.
- Paralelismo explícito:
  - É a exploração da característica populacional de AGs (e dos algoritmos baseados em populações), onde as operações sobre os indivíduos podem ser feitas em paralelo.

1. Motivações para uso de AG paralelos

- Dimensionalidade:
  - O espaço de busca cresce exponencialmente. Isto é, para *k* variáveis de *b* bits, num.soluções= $2^{(k*b)}$
  - A população é apenas uma amostra do espaço de busca
- Parâmetros de controle
  - AGs simples não têm utilidade prática. AGs eficientes podem ter um grande número de parâmetros de controle
  - O efeito da interação entre os parâmetros geralmente é pouco conhecido
  - Não há metodologia consagrada específica para ajuste dos parâmetros
- Inspiração natural
  - Na natureza, a evolução dos indivíduos e das populações é "paralela" e não "sequencial"
- Função de *fitness*
  - Para muitas aplicações tem um alto custo computacional

1. Classificação de AGs paralelos



1. Classificação de AGs paralelos

- Quanto ao hardware depende dos recursos disponíveis:
  - Rede:
    - Cada população é gerenciada por um processador
    - A comunicação entre populações é feita pela rede
    - Só é viável se houver alto processamento local, devido ao alto custo da comunicação entre processos
  - Multi-CPU/multi-*core*:
    - Baseia-se em processadores de múltiplos *cores*
    - Cada *core* gerencia uma população
  - GPU:
    - Tira proveito do grande número de *cores* disponíveis para avaliar paralelamente a população

1. Classificação de AGs paralelos

- Quanto ao modelo depende do suporte de *hardware* disponível, podendo ser:
  - Global distribuído (*farm model*):
    - Processo *master*: mantém a população, faz a seleção e aplica os operadores genéticos
    - Processos *slaves*: avalia indivíduos (somente a função de *fitness*)
  - Insular (*island model*):
    - Cada processo (processador) controla uma população que evolui independentemente, permitindo migrações periódicas
  - Celular (*cell model*):
    - Cada processo controla um indivíduo que interage somente com seus adjacentes

1. Classificação de AGs paralelos

- Quanto ao sincronismo entre processos (Grefenstette, 1981), podem ser:
  - *Master-slave* síncrono:
    - O processo *master* controla tudo, enquanto os *slaves* somente avaliam os indivíduos.
    - Cada ciclo de evolução (geração) depende do término de cada *slave*.
  - *Master-slave* semi-síncrono:
    - Idem o anterior, porém o *master* aloca tarefas aos *slaves* à medida que ficam desocupados.
  - Concorrente assíncrono:
    - Populações evoluem independentemente e compartilham uma "memória" comum (*pool* de indivíduos, todos ou alguns).

1. Classificação de AGs paralelos

- Quanto à granularidade, considera-se a relação entre o esforço de computação e o de comunicação entre os processos, podem ser:
  - Granulação grosseira (*coarse grain*):
    - Populações evoluem independentemente e há trocas periódicas de indivíduos entre populações
    - Ocorre alto processamento local e baixa comunicação entre processos
  - Granulação fina (*fine grain*):
    - Os indivíduos interagem somente com os indivíduos controlados por processos adjacentes, de acordo com uma topologia definida
    - Ocorre baixo processamento local e alta comunicação entre processos

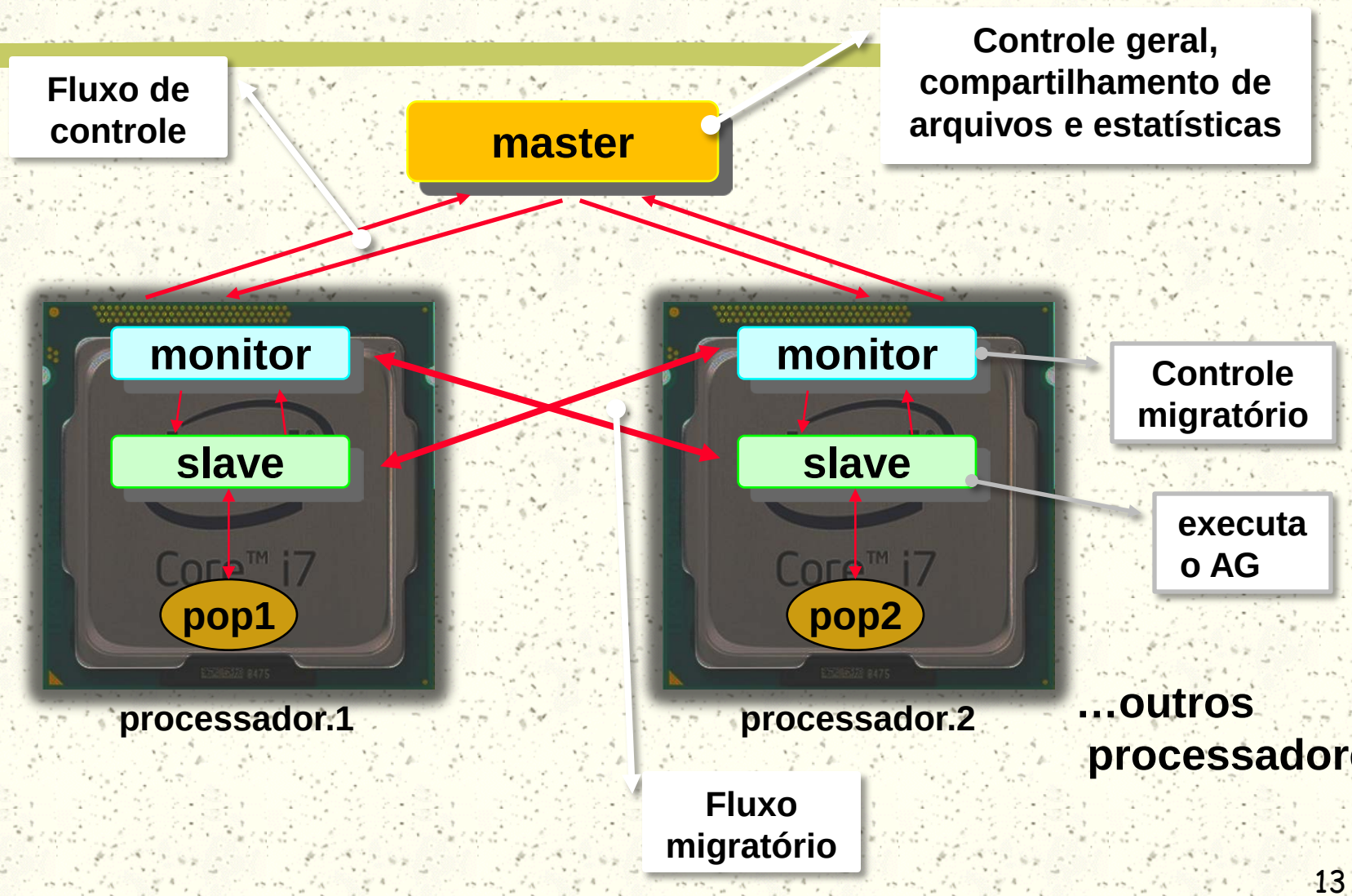
1. AGP insular

- É o método de paralelismo de AGs mais comumente encontrado na literatura
- Questões importantes para AGP insular:
  - Definição da política migratória:
    - Esquema de seleção dos imigrantes
    - Taxa de imigração e emigração
    - Qual população pode permitir imigração e emigração
    - Esquema de substituição de emigrantes
  - Controle dos emigrantes:
    - Efeito de conquista
    - Efeito de extinção

1. AGP no software GALLOPS

- ***Onepop:*** uma única população (serial)
- ***Manypops:***
  - Permite a simulação sequencial de um AGP insular
  - A conectividade e a política migratória das ilhas é definida em um arquivo de parâmetros
- GALOPPS-PVM
  - É uma versão de AGP insular que utiliza PVM -*Parallel Virtual Machine* para controlar processos em diferentes máquinas
  - Há 3 tipos de processos: *master*, *monitores* e *slaves*

1. AGP no GALLOPS-PVM



2. AGs híbridos – Por quê ?

- Inspiração:
  - As idéias (humanas) criativas são induzidas pelo conhecimento.
- Ags usam uma abordagem "Caixa-Preta" e não exploram explicitamente o conhecimento do domínio para melhorar sua busca:
  - Isto é bom pois torna AG uma técnica independente do problema e do contexto
  - Isto é mau pois coloca AG em desvantagem com outros métodos que exploram conhecimento específico do domínio do problema

- Caminho natural: AGs híbridos:
  - Uso conjunto de outras técnicas (p. ex de busca local)
  - Uso de operadores "**inteligentes**"

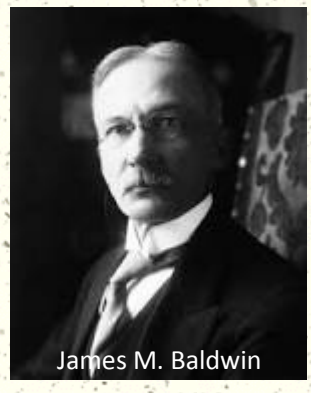
2. AG's híbridos

- Operadores genéticos especializados:
  - Incorporam conhecimento do problema
- P. exemplo? busca local:
  - Busca exaustiva (2-opt, 3-opt, etc)
  - Gradiente descendente
  - *Fast local search / neighborhood search / chaotic search*, etc..
- Utilização conjunta com outras técnicas:
  - *Simulated annealing, Tabu search,* etc.
  - Programação matemática
  - Outras técnicas de computação evolucionária: PSO, ABC...
  - Acoplamento método fraco-forte

2. Interação entre aprendizado e evolução
O efeito Baldwin

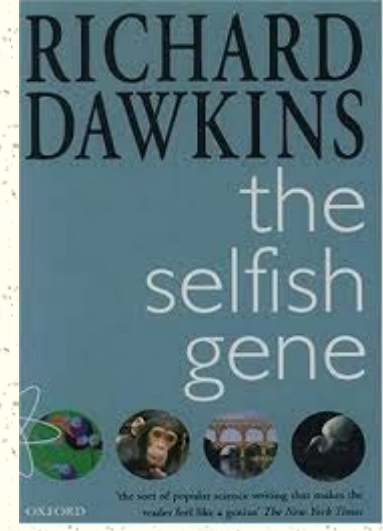
- Interações / analogia:
  - Aprendizado: processo evolutivo durante a vida de um organismo.
  - Evolução: processo adaptativo durante a história da da espécie na Terra
- Hipótese de Lamarck:
  - "Habilidades adquiridas durante a vida de um organismo podem ser transmitidas geneticamente aos descendentes deste organismo".
- Efeito Baldwin (Ontogenética):
  - O aprendizado causa "plasticidade fenotípica", que por sua vez, pode favorecer a evolução da espécie.
- Motivação para <u>hibridizar</u> os AGs

Jean B. Lamarck



James M. Baldwin

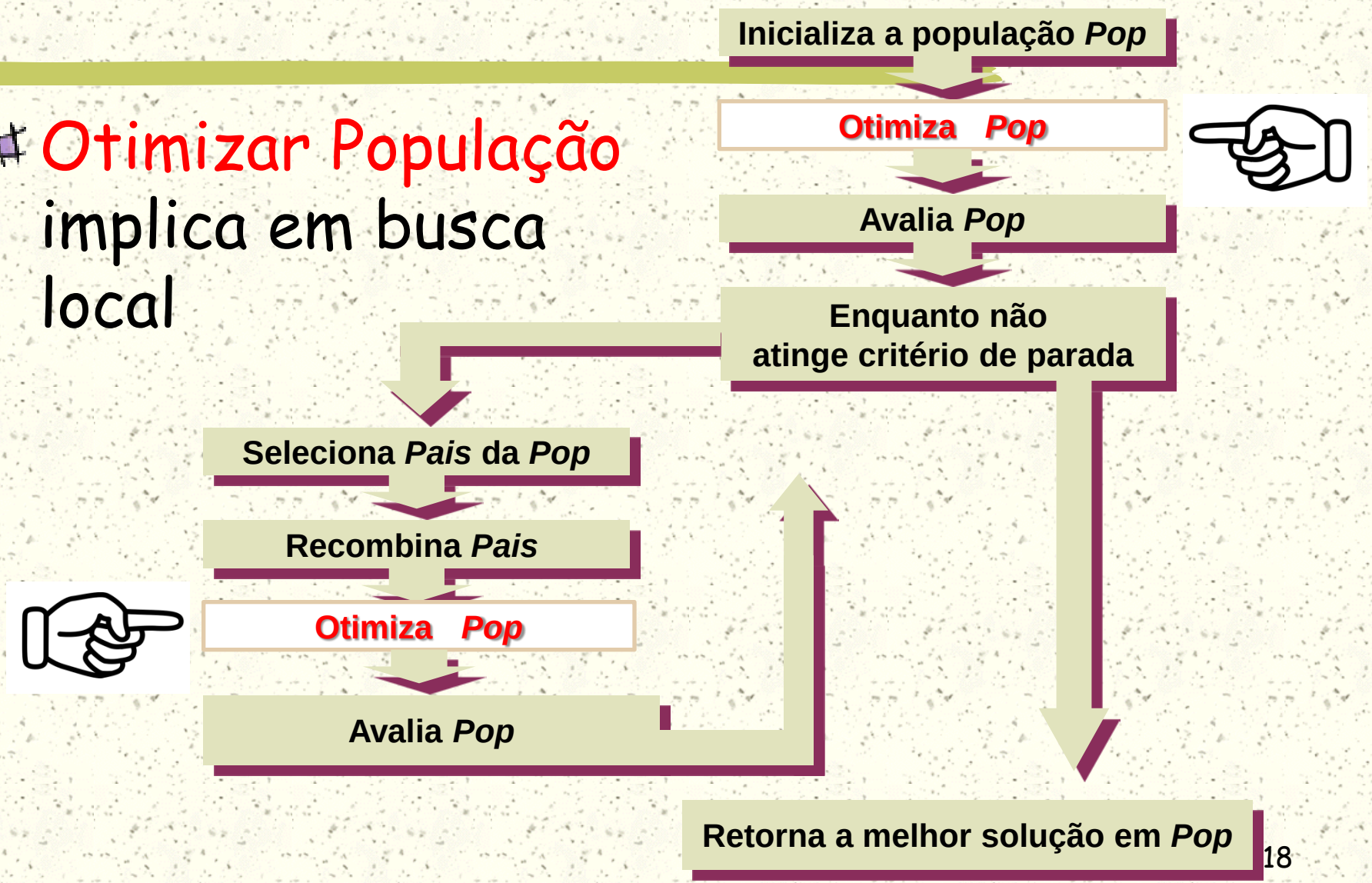
3. Algoritmos meméticos



- 'Meme': palavra introduzida por Richard Dawkins no livro The Selfish Gene (1976).
  - Unidade de imitação, análoga do gene, na evolução cultural.
- Os AM's levam em consideração a transmissão cultural da informação
- Diferenças AG x AM:
  - Em AG: indivíduo, em AM: agente
  - Em AM, um fenótipo pode ser gerado por vários genótipos. Em AG geralmente estão mais vinculados a só um tipo de decodificação (AMs oferecem mais liberdade).
  - Em AM a busca local é fundamental para a evolução e permite a modificação do genótipo

3. Esquema básico de um Algoritmo Memético

- Otimizar População implica em busca local



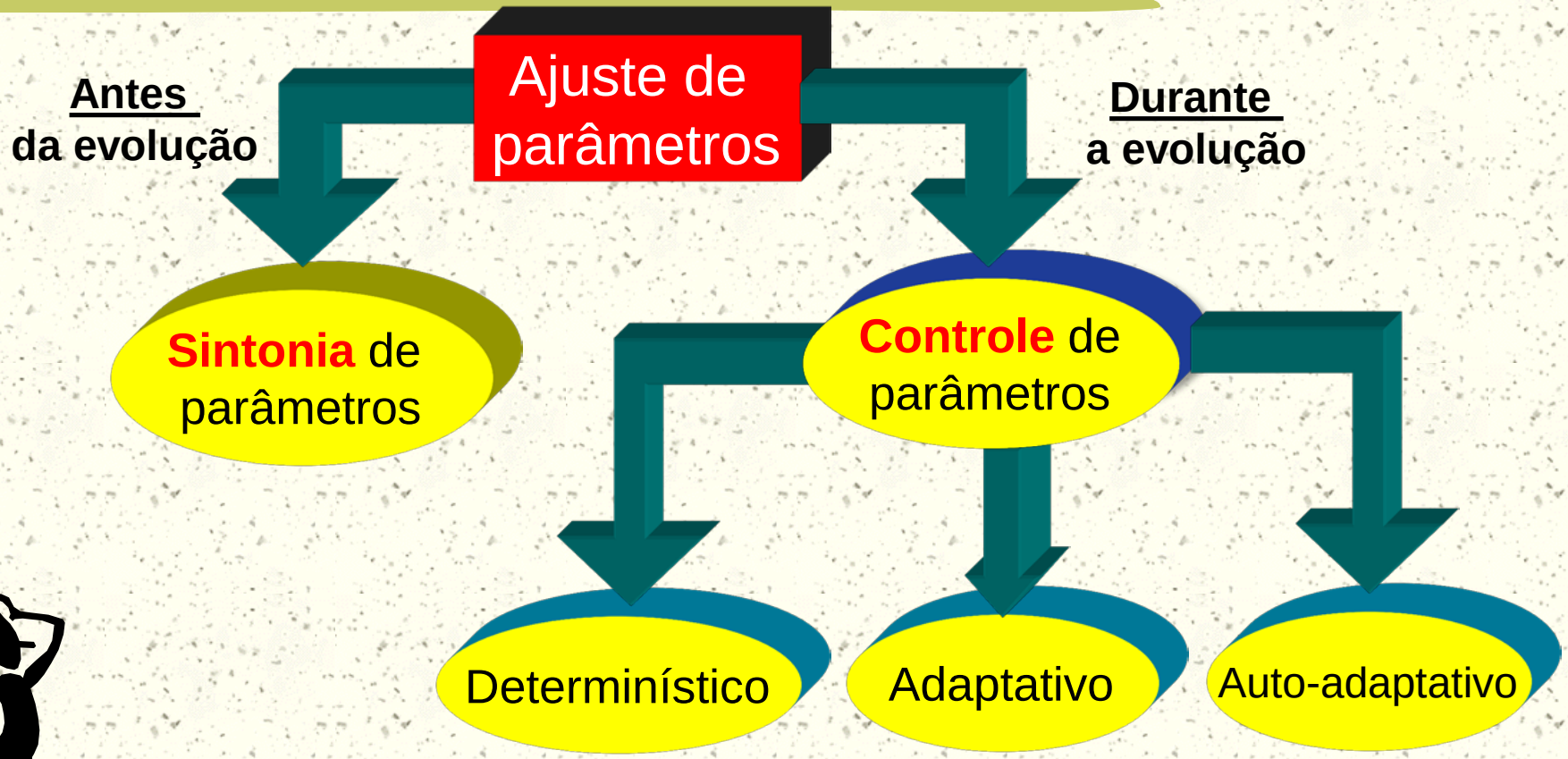
4. Ajuste dos parâmetros de controle

- A eficiência de um AG (e dos outros algoritmos evolucionários) é profundamente afetada pela escolha dos parâmetros de controle, em especial:
  - Tamanho da população
  - Número de gerações
  - Probabilidade de mutação e de *crossover*
  - Método de seleção
  - Estratégias para controle da pressão seletiva

4. Ajuste dos parâmetros de controle

- Considerando que um AG é um processo dinâmico e adaptativo, o valor ideal dos parâmetros também deveria variar ao longo da busca
  - Como encontrar parâmetros adequados para um determinado problema ?
  - Como variar adequadamente os parâmetros ao longo da busca ?

4. Métodos de ajuste dos parâmetros de controle



4. Sintonia de parâmetros

- É a maneira tradicional e normalmente a menos eficiente
  - Estabelecer os valores antes de rodar, rodar o algoritmo e avaliar o resultado
  - Os valores dos parâmetros permanecem fixos durante a rodada
- Problemas:
  - O usuário pode não ter conhecimento para ajustar os parâmetros em faixas de valores adequados
  - Pode existir forte interação entre parâmetros
  - O conjunto ideal de parâmetros pode variar de problema para problema e mesmo de instância para instância do mesmo problema (o conhecimento empírico não serve)
  - Valores de parâmetros inicialmente bons podem se tornar ruins ao longo da evolução
  - Surge, então, um meta-nível de otimização: como otimizar os parâmetros ?

4. Sintonia de parâmetros

- Experimento fatorial:
  - Consiste em experimentar todas as possíveis combinações dos parâmetros sob teste
  - Pode ser computacionalmente muito caro

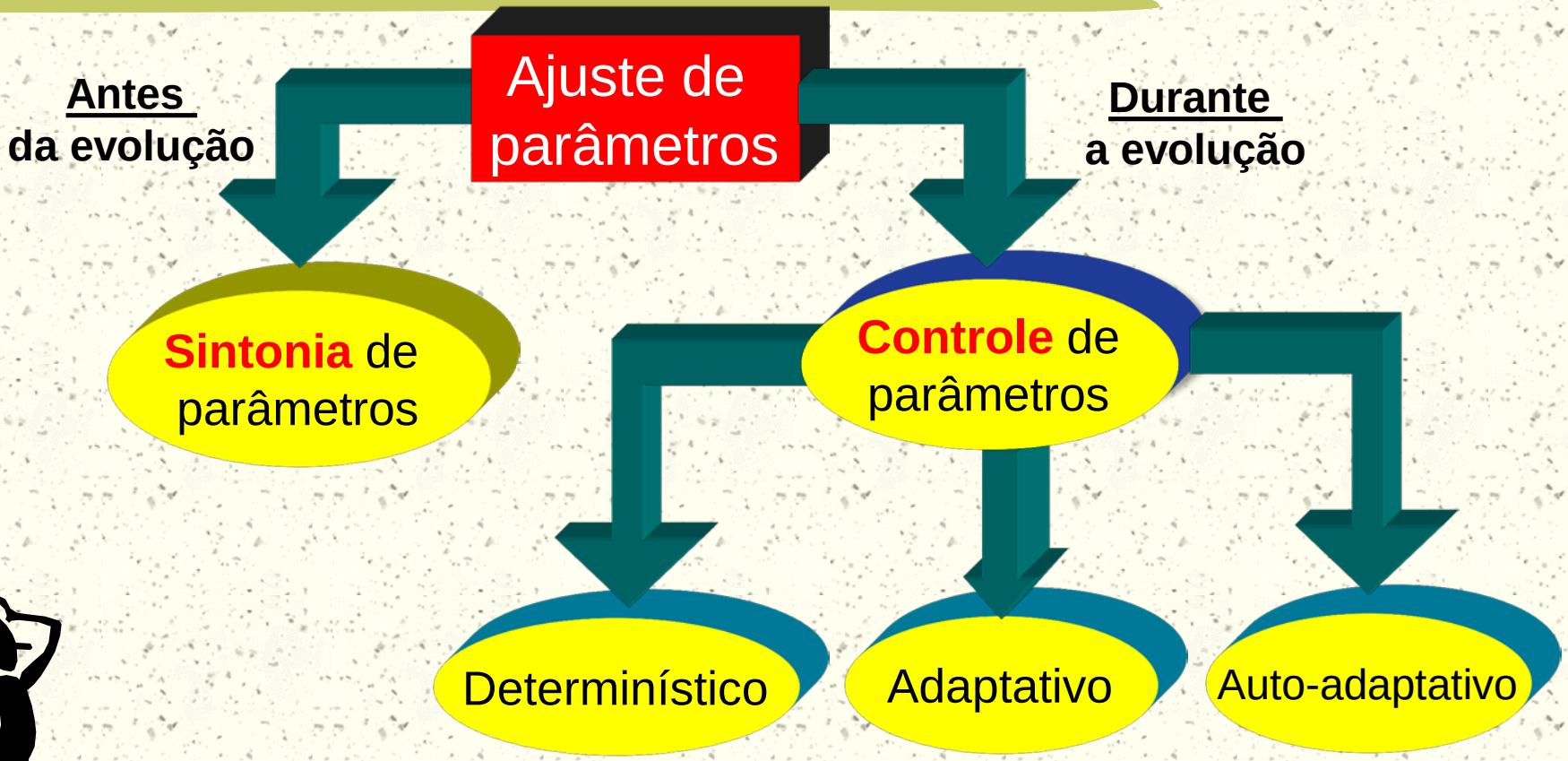
- Meta-nível de ajuste:
  - Consiste em utilizar um AG para otimizar os parâmetros de um outro AG.
    - No nível alto, indivíduos codificam conjuntos de parâmetros
    - No nível baixo indivíduos codificam as variáveis do problema
  - Há uma versão do GALLOPS para isto: SAGA
  - Há poucos estudos sobre esta abordagem de sintonia de parâmetros

4. Sintonia de parâmetros

- Na tese de doutorado de K. DeJong (1975):
  - Estudou o desempenho de um AG simples para diversos problemas de otimização de funções algébricas e propôs um conjunto de parâmetros "padrão":
    - População = 50
    - *Crossover*: 1-ponto com p_{cross}=60%
    - Mutação: de bit simples com p_{mut}=1% por bit
    - *Generation gap* = 1
- Na prática, os parâmetros mais usuais para um AG são:
  - População = 100..200
  - Gerações = 200...500
  - Operadores: p_{cross}=70...90% (1 ou 2 pontos), p_{mut}=1/λ
  - Seleção: por torneio com k=3% de *pop*
  - Escalonamento com c=1,3
  - Elitismo: normalmente não, se utilizar, só em conunto com técnicas de controle da pressão seletiva

IMPORTANTE

4. Métodos de ajuste dos parâmetros de controle



4. Controle de parâmetros

- Determinístico:
  - Uma regra (heurística) preestabelecida modifica o valor de um parâmetro sem utilizar nenhuma realimentação do processo evolutivo
  - Normalmente a regra é tempo-dependente e é uma função do número da geração
- Adaptativo:
  - Utiliza algum tipo de realimentação da busca para determinar a direção e intensidade do ajuste nos parâmetros
  - Pode-se utilizar: *on-line/off-line performance*, diversidade genética, número de gerações sem melhoria

4. Controle de parâmetros

- Auto-adaptação:
  - Uma vez que a idéia central em AG (e outros algoritmos evolucionários) é a evolução, é razoável utilizar o mesmo princípio não só para evoluir boas soluções para um problema, mas também evoluir o próprio algoritmo
  - Os parâmetros do AG são codificados no cromossomo e submetidos ao processo evolutivo
  - É uma abordagem muito promissora e, para alguns problemas, foi mostrado que o seu desempenho é, pelo menos, melhor do que o melhor conjunto estático de parâmetros

4. Controle de parâmetros

- Os parâmetros de controle são codificados no cromossomo e evoluem com o algoritmo
- Problemas da auto-adaptação:
  - O quê ?
    - Codificar os parâmetros propriamente ditos
    - Codificar um valor limitado que é somado/subtraído dos parâmetros atuais
  - Como ?
    - Pegar os parâmetros do indivíduo de melhor *fitness*
    - Pegar os parâmetros dos m melhores indivíduos e tirar a média ($2 <= m <= \lambda$)
  - Quando ?
    - Alterar os parâmetros a cada geração
    - Alterar os parâmetros após um número fixo de gerações

4. Controle de parâmetros

- Vantagem da auto-adaptação:
  - O algoritmo se adapta ao problema e ao momento da busca
  - O mesmo algoritmo pode ser utilizado para problemas diferentes
- Desvantagem da auto-adaptação:
  - É necessário embasamento teórico responder "o quê", "como" e "quando"