Algoritmos Genéticos

aula 2 - exemplos

Caixeiro viajante

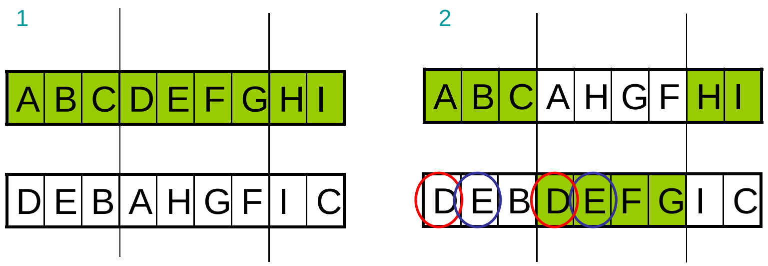
Caixeiro viajante

- representação intuitiva de indivíduo: o próprio caminho

D	E	B	A	H	G	F	I	C

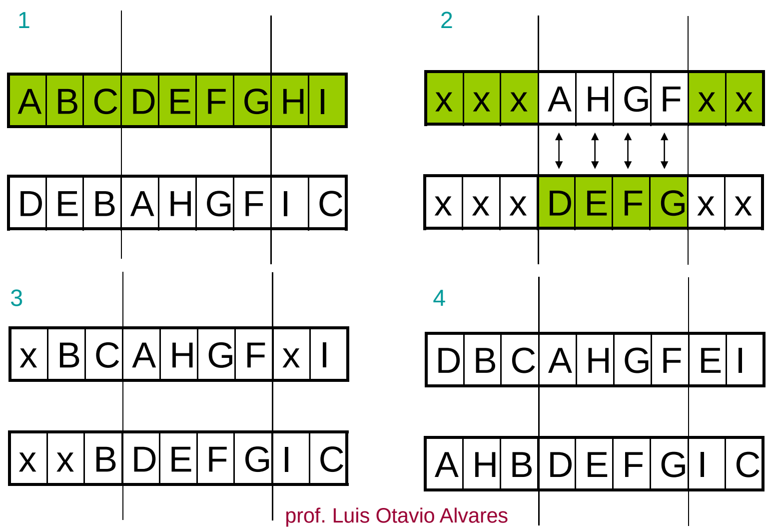
- problema: crossover

Exemplo: recombinação de 2 pontos

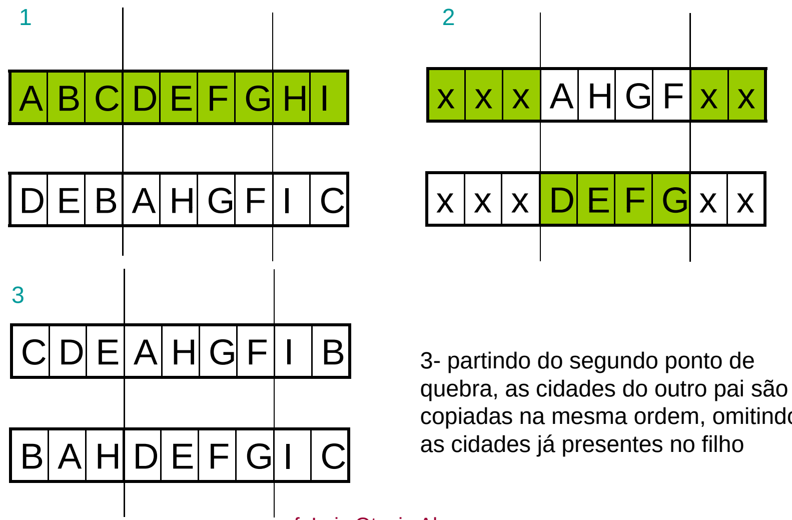


Cidades repetidas - inválido!!!

PMX - partially mapped crossover



OX - ordered crossover



3- partindo do segundo ponto de quebra, as cidades do outro pai são copiadas na mesma ordem, omitindo as cidades já presentes no filho

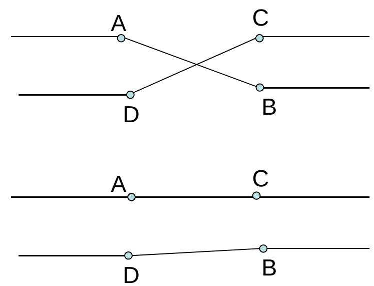
exemplos de mutação

- inversão: de duas cidades adjacentes
- inserção: seleciona uma cidade e a coloca aleatoriamente em algum ponto
- deslocamento: seleciona um sub-percurso e o coloca aleatoriamente em algum ponto
- troca recíproca: troca de posição entre duas cidades

2opt method

AB e CD fazem parte do percurso

se $\overline{AB} + \overline{CD} > \overline{AC} + \overline{BD}$ então faço a troca para AC e BD



Caixeiro viajante

Exemplos de programa

http://www.mac.cie.uva.es/~arratia/cursos/UVA/GeneticTSP/JAVASimultn/TSP.html

http://www.dna-evolutions.com/dnaappletsample.html

Trabalho de Jorge Meinhardt

- *“O presente trabalho pretende mostrar a possibilidade de uso de uma ferramenta de inteligência artificial utilizando a abordagem de algoritmo genético para auxílio ao projeto altimétrico de rodovias.”*

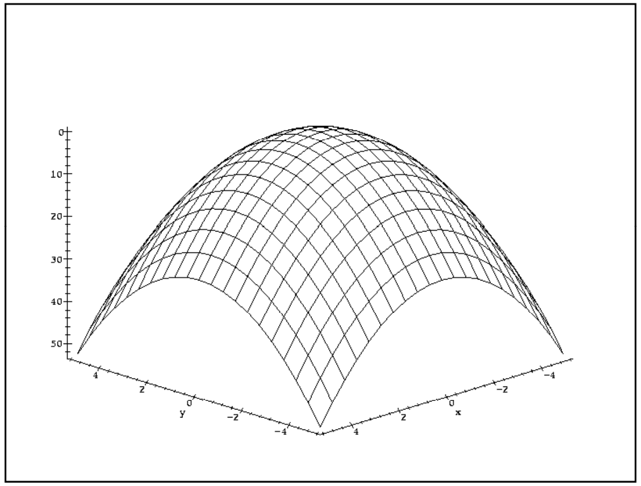
Outros exemplos

- Robocup: p/ melhorar o drible, para melhorar a condução da bola, etc...
- ChicuxBot – Genetic Algorithm Configured Behavior Network Multi-Agent for Quake II
- tese de André Schneider: controle de robôs

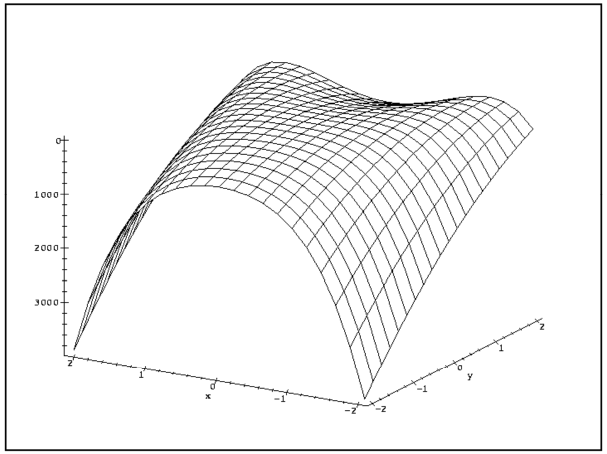
De Jong e Otimização de Funções

- **De Jong**, em sua tese “An analysis of the behavior of a class of genetic adaptive systems”, fez uma investigação cuidadosa da aplicação de algoritmos genéticos à otimização de funções.

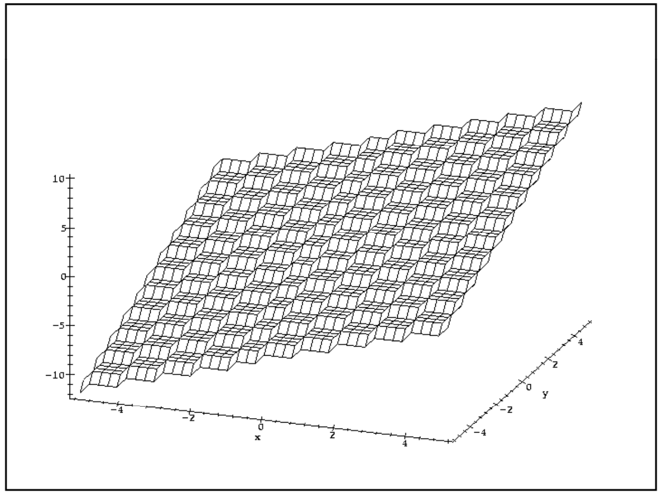
$$f_1(x_i) = \sum_{i=1}^{3} x_i^2, \qquad -5.12 \le x_i \le 5.12$$



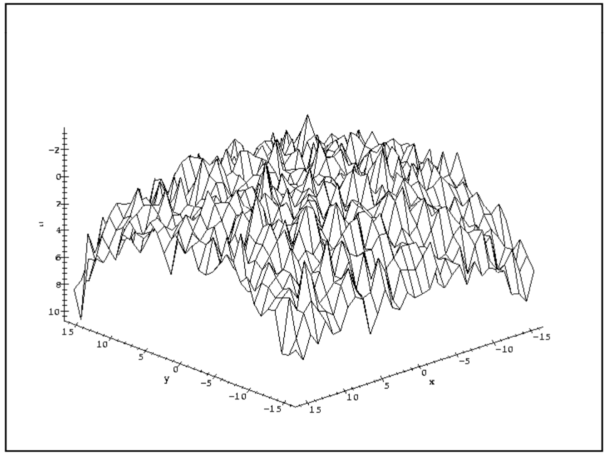
$$f_2(x_i) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2, \qquad -2.048 \le x_i \le 2.048$$



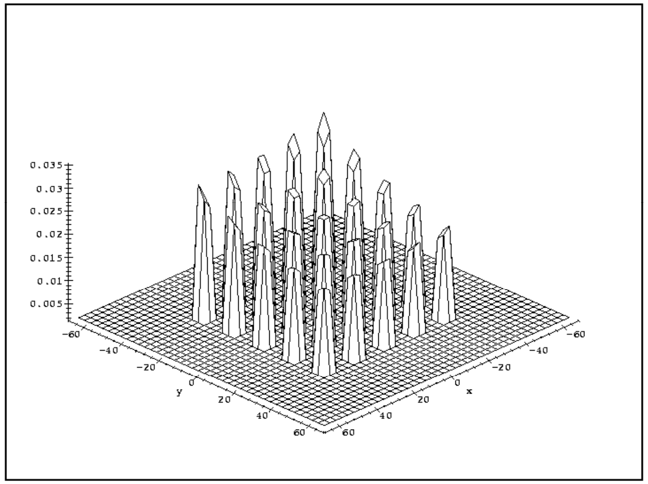
$$f_3(x_i) = \sum_{i=1}^{5} floor(x_i), \qquad -5.12 \le x_i \le 5.12$$



$$f_4(x_i) = \sum_{i=1}^{30} i x_i^4 + Gauss(0,1), \qquad -1.28 \le x_i \le 1.28$$



$$f_5(x_i) = 0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}, \qquad -65.356 \leq x_i \leq 65.356$$



An Evolutionary Behavior Tool for Reactive Multi-agent Systems

Andre Zanki Cordenonsi

andrezc@unifra.br

Centro Universitário Franciscano
Área de Ciências Exatas
Rua dos Andradas 1614
Centro - Santa Maria – RS
CEP 97010 - 032

Luis Otávio Alvares

alvares@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500
Bairro Agronomia - Porto Alegre - RS -Brasil
CEP 91501-970 Caixa Postal: 15064

1. Introduction (…)

Why to use an Evolutionary Reactive Multi-agent Systems ?

- adaptation to dynamically environments
- adaptation to unknown environments

Objective of this paper: shows the specification and implementation of an Evolutionary Reactive Multi-agent Systems where the behavior of the agents can be modified during the simulation

1. Introduction (…)

Previous work: the Simula Tool! [Frozza]

A graphical tool to teach the multi-agent technology developed in the Instituto de Informática – UFRGS by Rejane Frozza and Luis Otávio Alvares

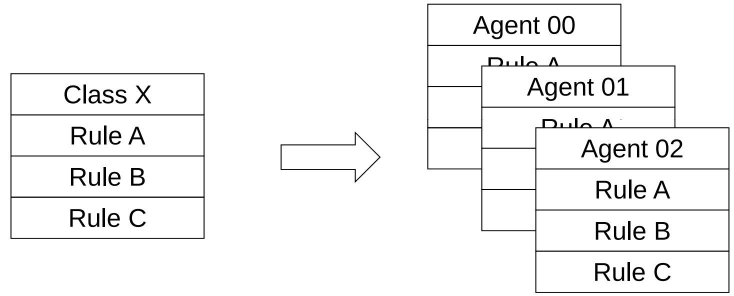
A tool to simulate reactive multi-agent systems, using a graphical interface to build the agents behavior and to shows the simulation.

3. Simula ++

- Reactive Multi-agent Systems
- Didactic tool
- Graphical User Interface

3. Simula ++ (…)

Class of agents and the Behavior



Initial State

3. Simula ++ (…)

Each Agent has

- Set of Independent Elements
- Chromosome (set of rules)

3. Simula ++ (…)

Independent Elements (equal for all agents of the same class)

- Initial Energy
- Energy Amount
- Maximum Life Time
- Life Time
- Sexual Maturity Time
- New Generation Time

3. Simula ++ (…)

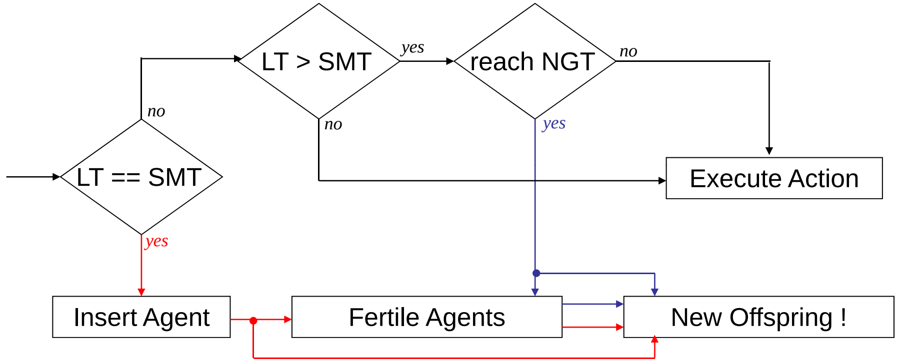
Chromosome: the rule characteristics

- declarative
- precondition
- action
- priority

RULE A
priority = 7
if (not perceive agent A) then
random_move();

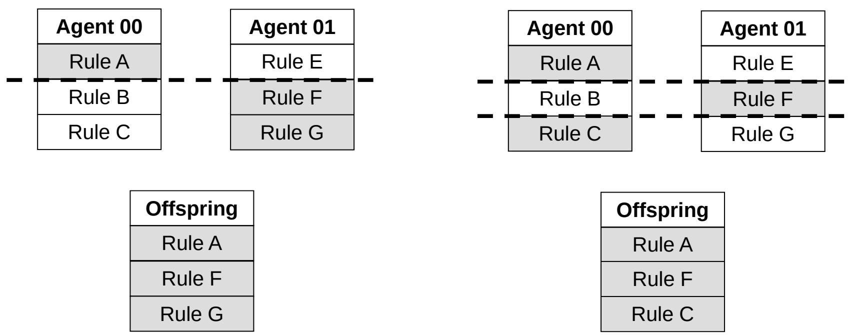
3. Simula ++ (…)

The new Generation Algorithm



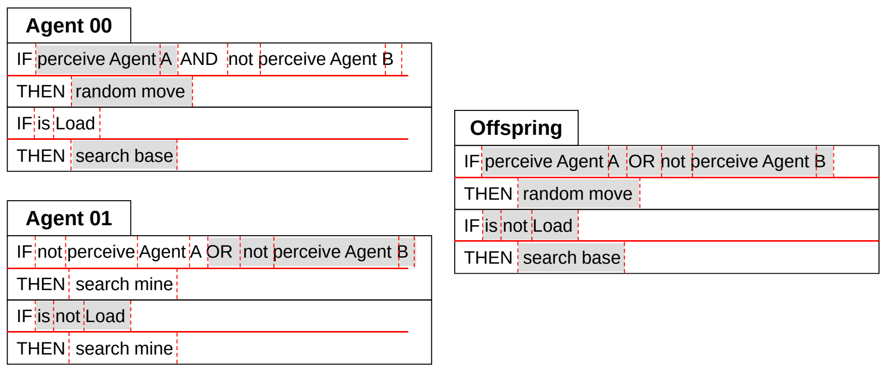
4. Genetic Operators

One-Point External Crossover   Two-Point External Crossover



4. Genetic Operators (…)

Internal Crossover



4. Genetic Operators (…)

Mutation

- between 1% and 2%
- change pieces of a rule (respecting the semantic value)

    AND/OR

    Agent A, Agent B, Agent C, …

    perceive_agent, search_agent, escape_from_agent,…

5. Food Foraging Problem

- Collect all minerals from an unknown environment
- The environment is a grid (100 x 100)
- Base is fixed
- All agents known where is the base
- There are three mines, with 100 units of mineral

5. Food Foraging Problem (…)

It was defined six simulation groups:

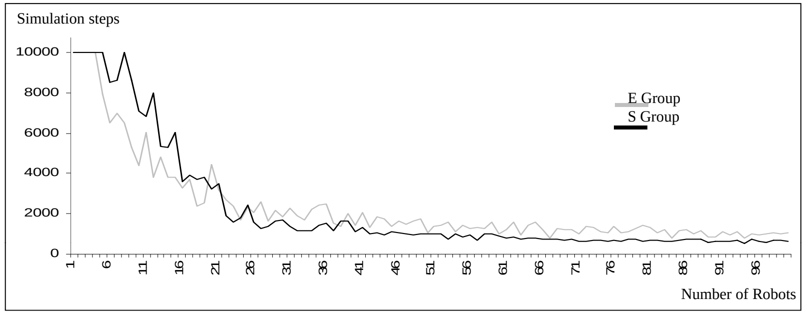
- Petit Poucet I [Drougol]
- Petit Poucet II [Drougol]
- Petit Poucet III [Drougol]
- Dockers [Drougol]
- Evolutionary Group
- Standard Group

5. Food Foraging Problem (…)

Standard Group (evolved by the evolutionary algorithm)

IF NOT perceive (Mine) AND NOT perceive (Robot)
    THEN random move
IF NOT perceive (Mine) AND perceive (Robot)
    THEN follows (Robot)
IF NOT perceive (Mine) AND reach (Robot)
    THEN flee (Robot)
IF perceive (Mine)
    THEN load AND return to base AND leave (Mark Track)
IF perceive (Mark Track)
    THEN follows(Mark Track) AND remove(Mark Track)

5. Food Foraging Problem (...)

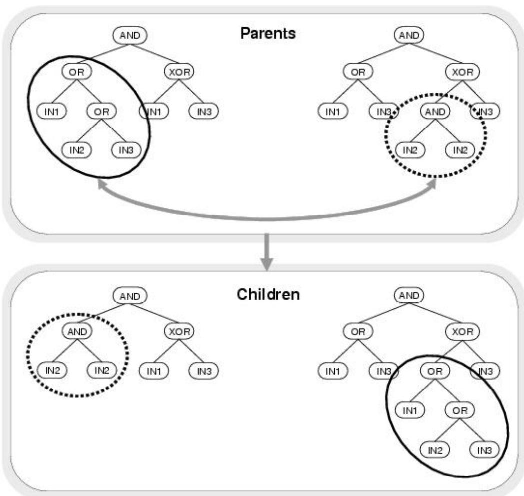


	PP1	*PP2*	*PP3*	*Doc.*	*E*	*S*
Time to collect all minerals	3351	5315	3519	1746	2303	2217
Minimum time to collect all minerals	1113	1607	1075	695	770	540
Number of Robots for the minimum time	64	98	87	84	95	95

Programação Genética

- **Programação genética** é uma técnica automática de *programação* que propicia a *evolução* de *programas de computadores* que resolvem (ou aproximadamente resolvem) problemas
- Na programação genética, os indivíduos da *população* não são seqüências de *bits*, mas sim programas de computador armazenados na forma de *árvores sintáticas*. Tais programas é que são os candidatos à solução do problema proposto.

Programação genética



Referências

Koza, J.R. 1990, *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems*, Stanford University Computer Science

Koza, John R. 1992. *Genetic* Programming: *On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press.

Congressos e workshops específicos sobre o assunto