Computação Evolucionária

Alunos: Nabila de Paula e Silva
Paulo Henrique Cardoso de Souza
Werikcyano Lima Guimaraes

Computação Evolucionária

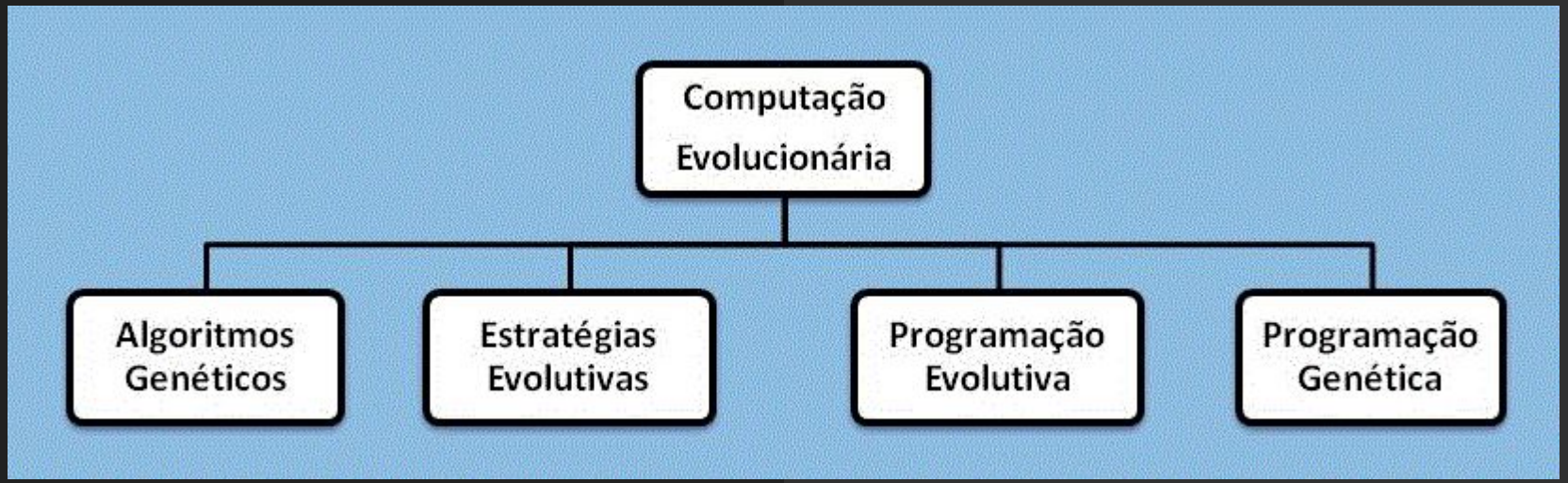
- Campo da computação inspirado no processo de evolução natural.
- Aplica os conceitos da evolução darwiniana na solução automatizada de problemas.
- Primeira aparição em artigos de FriedBerg em 1958 e Bremermann.
- Em 1965 estabelecem as 3 formas dos Algoritmos Evolutivos: Programação Evolucionária (EP), Algoritmos Genéticos (GA) e as Estratégias Evolutivas (EEs).

- Em 1990 começaram os esforços para que essas linhas de desenvolvimento pudessem interagir entre si. O resultado foi um campo de pesquisa, conhecido como Computação Evolucionária.

Evolução	Computação Evolucionária
Environment	*Problem*
Individual	*Candidate solution*
Fitness	*Quality*

Tabela 1. A metáfora básica da computação evolutiva que liga a evolução natural à resolução de problemas”

Computação Evolucionária



Algoritmos Evolucionários

- Aumento na aptidão da população
- Maximização da função de qualidade.

- **SELEÇÃO**:
  - Aumenta qualidade média das soluções.
  - Indivíduos com melhores resultados “semeiam” a próxima geração.
  - Utiliza-se de operadores de variação
- **OPERADORES DE VARIAÇÃO:**
  - Mutações e recombinações
  - Diversidade dentro da população.

- Componentes dos algoritmos evolucionários
  - Representação;
  - Função de avaliação;
  - População;
  - Mecanismo de seleção de pais;
  - Operadores de variação;
  - Mecanismo de seleção de sobrevivente .

Algoritmos Evolucionários

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Figura 1. Pseudocódigo de um algoritmo evolucionário.

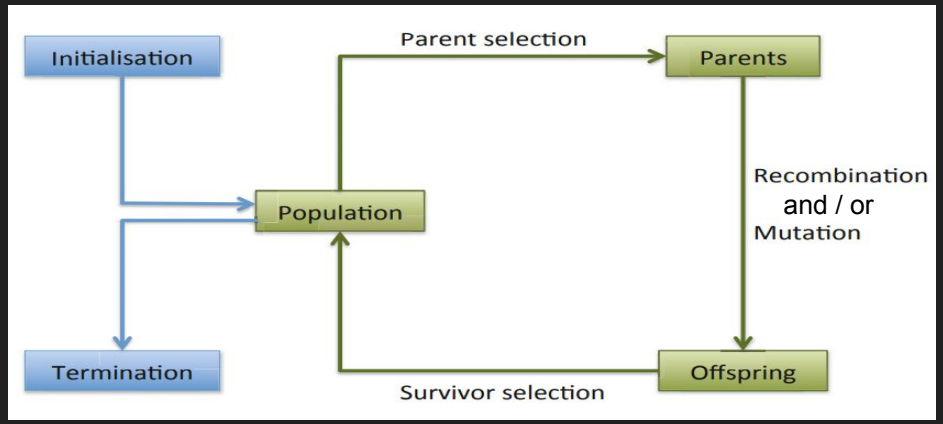


Figura 2. Esquema geral de um algoritmo evolutivo em forma de fluxograma

Algoritmos Genéticos (AG)

- Início nas décadas de 1950 e 1960.
- 1975 - “Adaptation in Natural and Artificial Systems”, John Holland.
- 1980 - David E. Goldberg.
- Simulação dos processos naturais.

- **População**: Conjunto de soluções de um problema.
- **Indivíduos**: Solução do problema.
- **Avaliação de Aptidão**: Avalia a qualidade da solução para problema.
- **Seleção**: Seleciona os indivíduos mais aptos para dar origem à próxima geração.

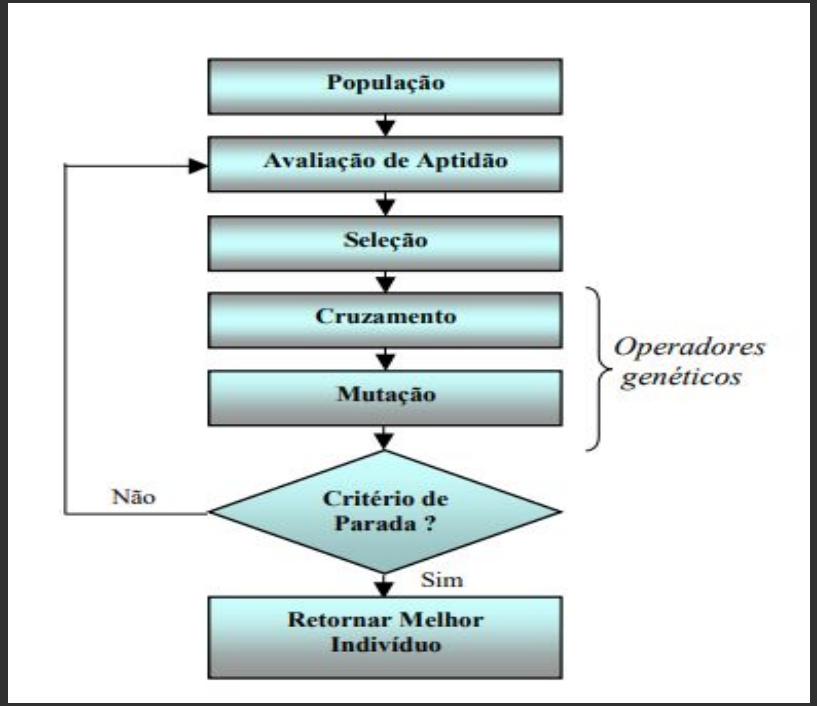
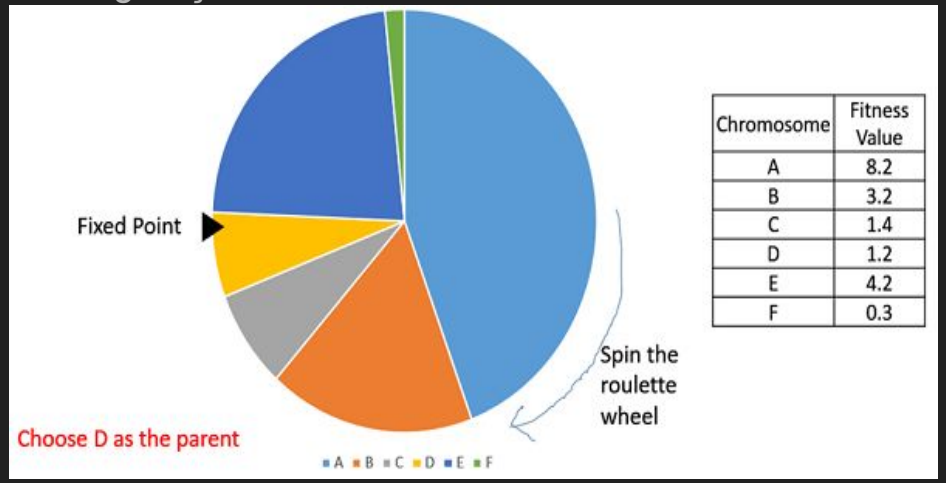


Figura 3. Esquema da estrutura de um algoritmo genético.

Algoritmos Genéticos

- Seleção
  - Método de Seleção por Roleta
    - Valor percentual proporcional ao seu índice de aptidão.
    - Indivíduos com alta aptidão possuem mais chances de influenciarem a próxima geração.



Chromosome	Fitness Value
A	8.2
B	3.2
C	1.4
D	1.2
E	4.2
F	0.3

Figura 4. Exemplo da seleção por roleta

Algoritmos Genéticos

- ## Operadores Genéticos

- Transformam as gerações da população
- Aplicado nos indivíduos selecionados na etapa de seleção.

Cruzamento ou Crossover

- Ocorre em maior frequência.
- Baseado na reprodução dos genes de uma célula.

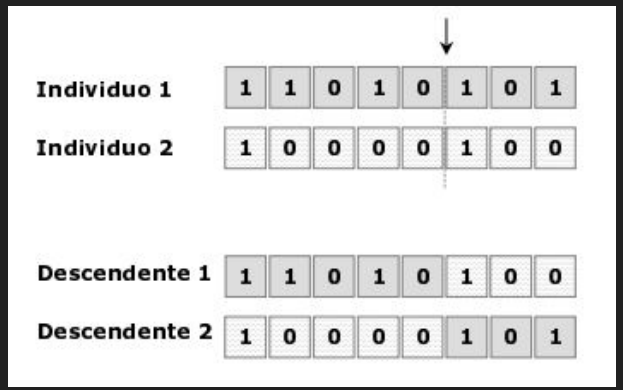


Figura 5. Exemplo de cruzamento

Mutação

- Ocorre com apenas um indivíduo que possui um ou mais de seus genes modificados.
- Menos recorrente.

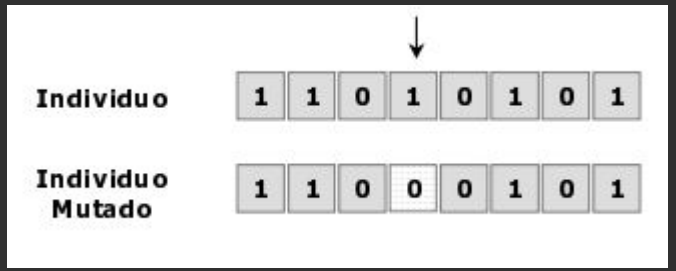


Figura 6. Exemplo de mutação

- ## Geração

- A cada nova iteração do algoritmo é gerado um novo conjunto de soluções.

Estratégia Evolutiva (EE)

- 1960, Alemanha - Teve sua primeira versão a partir de mutações a partir de um pai e um descendente por geração, sendo utilizado pela primeira vez em cálculos mecânicos.

Algoritmos de EE

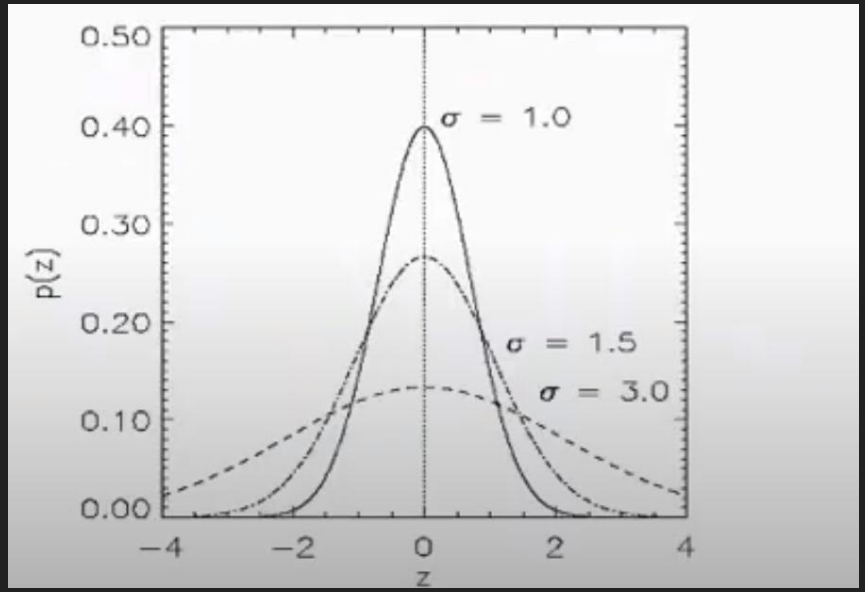
- Um indivíduo é representado pelo vetor:

$$v = \{x, \sigma\}$$

- Muito utilizado para se obter descendências, submetendo dois indivíduos a cruzamentos e mutações.
- Permitem o autoajuste dos seus parâmetros.

- Exemplos de algoritmos de EE:
  - **EE(1+1):** genitor gera um descendente e os dois são confrontados para se obter uma melhor resposta.
  - **EE(λ+μ):** μ indivíduos produzem λ descendentes gerando então uma população aleatória de (λ + μ), onde são escolhidos μ indivíduos.
  - **EE(λ, μ):** μ produz λ individuos, sendo λ > μ e apartir disso, μ descendentes são selecionados.

Estratégia Evolutiva (EE)



Mutação

$$x^{n+1} = x^n + z$$

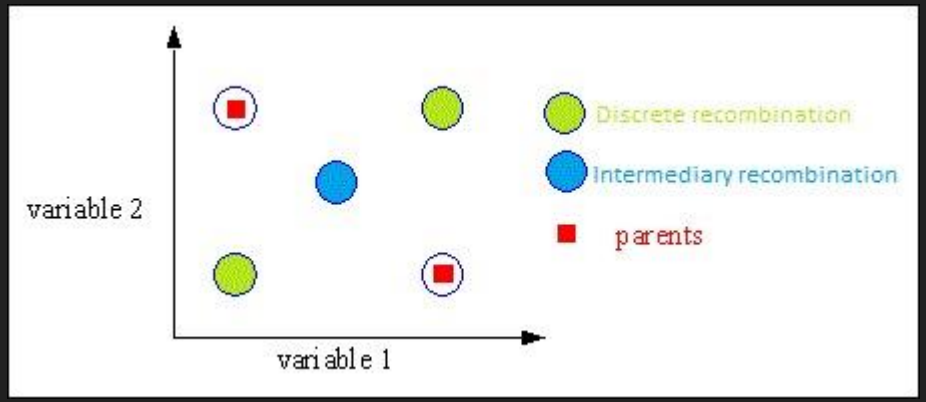
Regra do ⅕

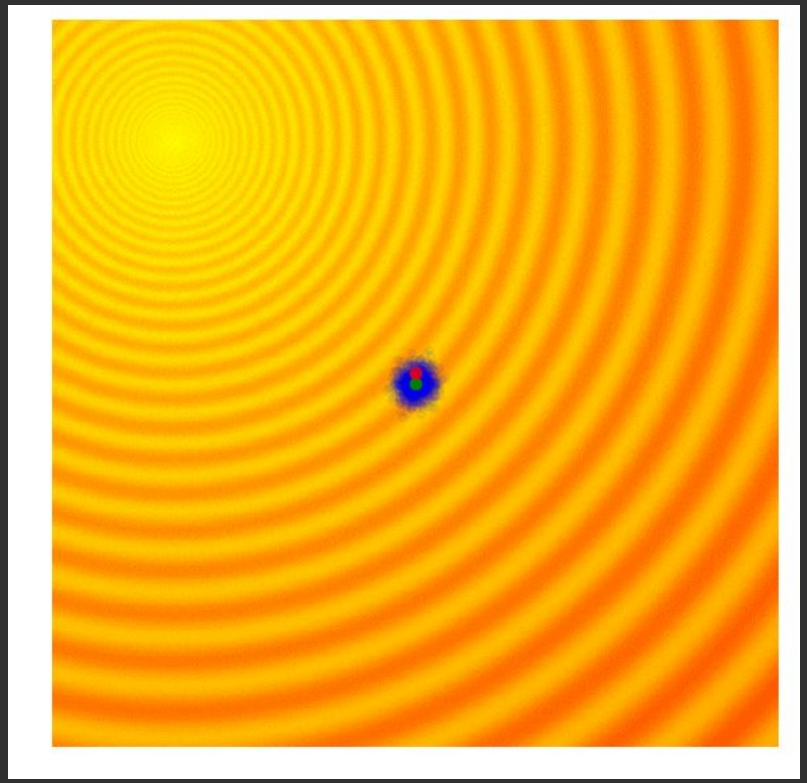
Se mais de ⅕ das mutações obtiverem sucesso σ irá diminuir concentrando as próximas mutações próximas das soluções atuais. Caso contrário, σ aumenta.

Recombinação

$$z_i = \begin{cases} (x_i + y_i)/2 & \text{intermediary recombination} \\ x_i \text{ or } y_i \text{ chosen randomly} & \text{discrete recombination} \end{cases}$$

Estratégia Evolutiva (EE)





Verde: Média das distribuições para caga geração.
Azul: Indivíduos da geração.
Vermelho: Melhor indivíduo.

Programação Evolucionária (PE)

- 1960 - Lawrence J. Fogel, EUA.
- Usar a evolução simulada como um processo de aprendizagem, tendo como objetivo gerar inteligência artificial.
- Muito parecida com a programação genética, com diferença na estrutura do programa.
- É garantido que todos os indivíduos irão produzir novos descendentes, porém apenas os k-melhores sobrevivem, sendo $k < N$, onde N é o tamanho da população (pais e filhos).
- Possibilidade de diminuição significativa da diversidade dos indivíduos.

- Indivíduo: cada um é representado por uma máquina de estados infinitos.
- Avaliação: uma vez que os indivíduos são escolhidos, eles passam pela análise da função *payoff*, que compara a saída da máquina com a saída esperada para solução do problema.
- Reprodução: feita apenas por operadores de mutação.
- Seleção: para a nova geração, os descendentes competem entre si e com os μ pais, sobrevivendo somente os indivíduos com maior fitness. Somente λ indivíduos irão sobreviver, de um total de $\lambda + \mu$ indivíduos candidatos.

Programação Evolucionária (PE)

- Funcionamento:

Pseudocódigo (Programação evolucionária):

- Gera aleatoriamente uma população inicial de M indivíduos segundo uma distribuição uniforme.
- Computa a função de fitness para os M indivíduos.
- Faça até a condição de parada (recurso computacional ou satisfação no resultado):
    - Para i de 1 até M:
        - Selecione o Pai i e o utilize para produzir um Filho
            - Faça uma cópia idêntica do pai
            - Então, de forma probabilística, altere (no caso, apenas por mutação) essa cópia
        - Calcule o fitness do filho
        - Mantenha o Filho em uma população separada
    - Unifique as populações:
        - Ordene os 2M indivíduos pelo fitness
        - Mantenha apenas os M indivíduos mais aptos.

Programação Genética (PG)

- 1980 - Desenvolvimento dos primeiros algoritmos LISP, baseados em uma árvore de decisão.
- 1988 - John Koza pioneiro do algoritmo genético para a evolução de um programa.
- Comum a utilização de árvores e nós. que facilitam a implementação.

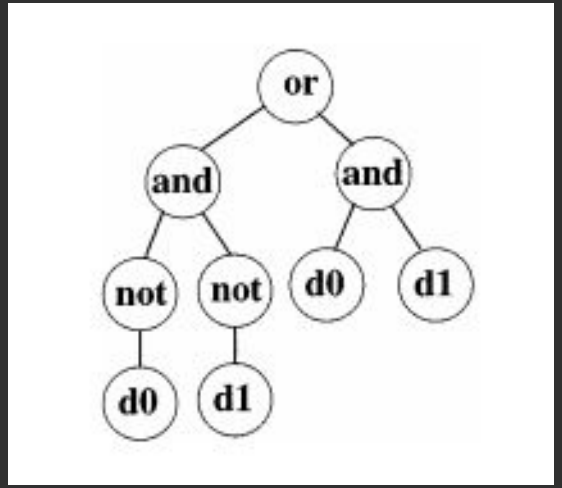


Figura 7. Estrutura em árvore de uma PG

Programação Genética (PG)

- Evolução de programas de computador usando analogias com muitos dos mecanismos utilizados pela evolução biológica natural.
- os indivíduos da população não são seqüências de bits, mas sim programas de computador armazenados na forma de árvores sintáticas.
- Os programas é que são os candidatos à solução do problema proposto.
- A recombinação se dá pela troca de subárvores entre dois indivíduos candidatos à solução

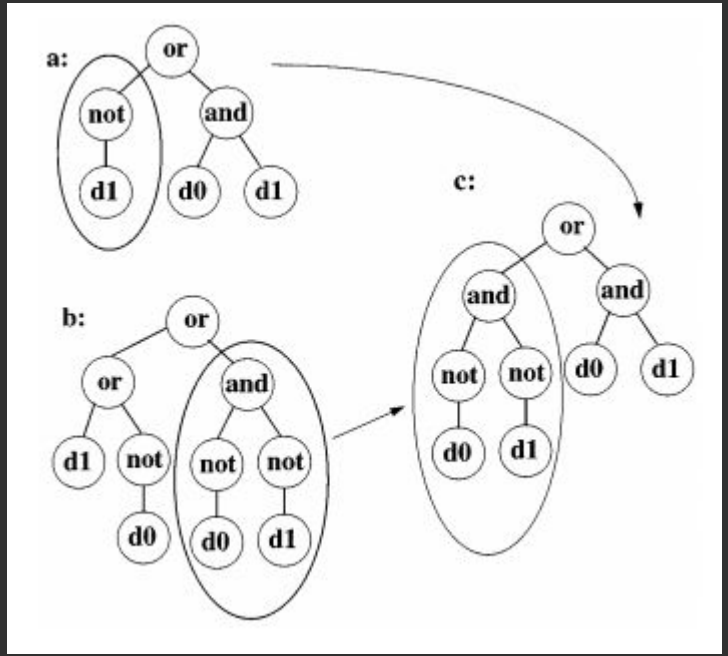
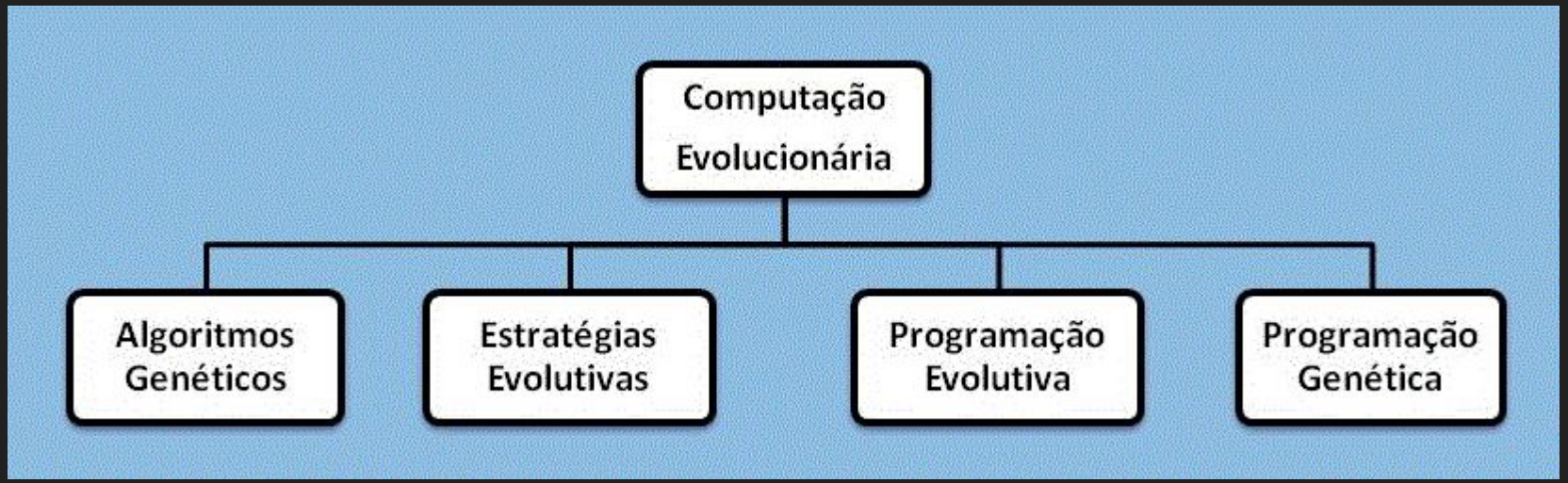


Figura 8. Exemplo do funcionamento da recombinação para geração de novos descendentes na PG.

Comparações entre AG, EE, PE e PG



Comparações entre AG, EE e PE

- Semelhanças:
  - São sistemas baseados em computação evolutiva.
  - Todas mantêm uma população de indivíduos/soluções potenciais.
  - Realizam o processo de seleção tendo como base a adaptação de um indivíduo.
  - Utilizam operadores genéticos.
  - PE e EE utilizam somente do operador de mutação.

Comparações entre AG, EE e PE

- Diferenças:
  - AG possuem como objetivo a formalização matemática e a explicação rigorosa de processos de adaptação em sistemas neurais e desenvolvimento de sistemas artificiais.
  - EE foram inicialmente propostas com o objetivo de resolver problemas de otimização de parâmetros, sendo discretos ou contínuos. Usam operadores de mutação, gerando grande contribuição nas análises e sínteses.
  - PE apesar das semelhanças com EE, diferem em pequenas coisas em relação aos procedimentos de seleção e codificação de indivíduos.
  - PG os candidatos à solução do problema são programas. Originalmente é escolhido, para cada indivíduo, ou mutação ou recombinação.

Exemplos de Aplicações

- Projetos arquitetônicos
  - Construção civil
  - APLICABILIDADE DOS ALGORITMOS GENÉTICOS NA ENGENHARIA CIVIL
  - Artigo
- Simulações
- Sistema de controle
  - Melhores soluções

EASEA

Framework para demonstração de um algoritmo evolucionário.

Foi utilizado o problema “One Max”, onde os seguintes resultados foram obtidos:

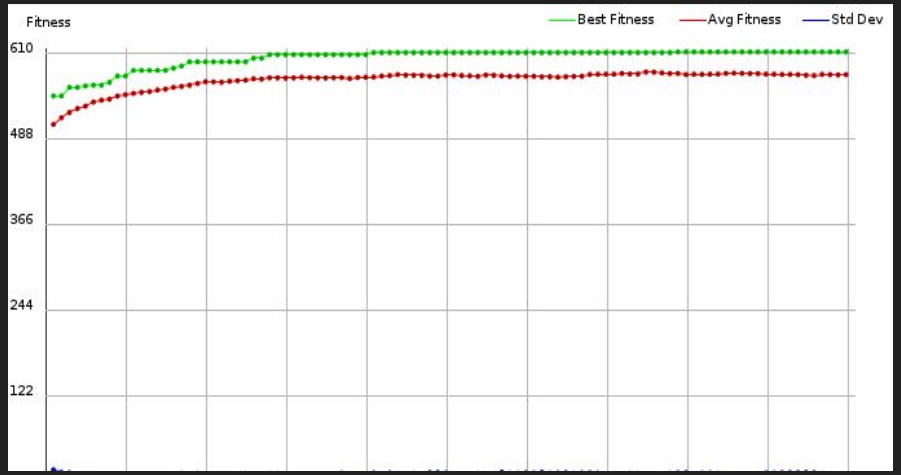


Figura 9. Plotagem dos dados

```
     65          0.170s            6600            6600 6.040000000e+02 5.7e+02 1.1e+01 5.5e+02
     66          0.172s            6700            6700 6.040000000e+02 5.7e+02 1.1e+01 5.4e+02
     67          0.174s            6800            6800 6.040000000e+02 5.7e+02 1.2e+01 5.3e+02
     68          0.177s            6900            6900 6.040000000e+02 5.7e+02 1.1e+01 5.5e+02
     69          0.179s            7000            7000 6.040000000e+02 5.7e+02 1.0e+01 5.5e+02
     70          0.181s            7100            7100 6.040000000e+02 5.7e+02 1.0e+01 5.4e+02
     71          0.183s            7200            7200 6.040000000e+02 5.7e+02 1.0e+01 5.5e+02
     72          0.186s            7300            7300 6.040000000e+02 5.7e+02 1.1e+01 5.5e+02
     73          0.188s            7400            7400 6.040000000e+02 5.7e+02 1.1e+01 5.5e+02
     74          0.190s            7500            7500 6.040000000e+02 5.8e+02 9.7e+00 5.5e+02
     75          0.192s            7600            7600 6.040000000e+02 5.8e+02 9.4e+00 5.5e+02
     76          0.195s            7700            7700 6.040000000e+02 5.8e+02 9.6e+00 5.6e+02
     77          0.197s            7800            7800 6.040000000e+02 5.7e+02 1.1e+01 5.5e+02
     78          0.199s            7900            7900 6.050000000e+02 5.7e+02 1.2e+01 5.5e+02
     79          0.201s            8000            8000 6.050000000e+02 5.7e+02 1.2e+01 5.5e+02
     80          0.203s            8100            8100 6.050000000e+02 5.7e+02 1.0e+01 5.5e+02
     81          0.206s            8200            8200 6.050000000e+02 5.7e+02 1.1e+01 5.4e+02
     82          0.208s            8300            8300 6.050000000e+02 5.7e+02 1.1e+01 5.5e+02
     83          0.210s            8400            8400 6.050000000e+02 5.7e+02 1.1e+01 5.5e+02
     84          0.213s            8500            8500 6.050000000e+02 5.7e+02 1.1e+01 5.5e+02
     85          0.215s            8600            8600 6.050000000e+02 5.8e+02 9.5e+00 5.5e+02
     86          0.217s            8700            8700 6.050000000e+02 5.7e+02 9.4e+00 5.5e+02
     87          0.219s            8800            8800 6.050000000e+02 5.7e+02 9.6e+00 5.5e+02
     88          0.221s            8900            8900 6.050000000e+02 5.7e+02 1.1e+01 5.5e+02
     89          0.223s            9000            9000 6.050000000e+02 5.7e+02 1.1e+01 5.4e+02
     90          0.225s            9100            9100 6.050000000e+02 5.7e+02 1.1e+01 5.5e+02
     91          0.228s            9200            9200 6.050000000e+02 5.7e+02 1.2e+01 5.5e+02
     92          0.230s            9300            9300 6.050000000e+02 5.7e+02 1.2e+01 5.5e+02
     93          0.232s            9400            9400 6.050000000e+02 5.7e+02 1.2e+01 5.5e+02
     94          0.234s            9500            9500 6.050000000e+02 5.7e+02 1.2e+01 5.5e+02
     95          0.236s            9600            9600 6.050000000e+02 5.7e+02 1.2e+01 5.4e+02
     96          0.238s            9700            9700 6.050000000e+02 5.7e+02 1.1e+01 5.4e+02
     97          0.240s            9800            9800 6.050000000e+02 5.7e+02 1.0e+01 5.5e+02
     98          0.243s            9900            9900 6.050000000e+02 5.7e+02 1.0e+01 5.4e+02
     99          0.245s           10000           10000 6.050000000e+02 5.7e+02 1.0e+01 5.4e+02
EASEA LOG [INFO]: Seed: 1616368206
EASEA LOG [INFO]: Best fitness: 605
EASEA LOG [INFO]: Elapsed time: 0.245264
```

Algoritmo:

```
/*_____________________________________________________________________
onemax.ez // Evolve individuals containing 11111111111111111...
_____________________________________________________________________*/

\User declarations :
#define SIZE 1000
#include <math.h>
float pMutPerGene=0.1;
\end

\User functions:
\end

\User CUDA:
\end

\Before everything else function:
\end

\After everything else function:
\end

\At the beginning of each generation function:
\end

\At the end of each generation function:
\end

\At each generation before reduce function:
\end

\User classes :
GenomeClass {
  int x[SIZE];
}
\end

\GenomeClass::display:
\end

\GenomeClass::initialiser : // "initializer" is also accepted
  for(int i=0; i<SIZE; i++ )
        Genome.x[i] = random(0,2);
\end

\GenomeClass::crossover :
  int nLocus=random(1,SIZE);

  for (int i=nLocus;i<SIZE;i++)
     child.x[i]=parent2.x[i];
```

```
  for (int i=nLocus;i<SIZE;i++)
     child.x[i]=parent2.x[i];
\end

\GenomeClass::mutator : // Must return a value (for historical reasons)
  for (int i=0;i<SIZE;i++)
    if (tossCoin(pMutPerGene)) Genome.x[i]=(Genome.x[i]+1)%2;

\end

\GenomeClass::evaluator : // Returns the score
  float fScore=0;
  for (int i=0;i<SIZE;i++) fScore+=Genome.x[i];
  return fScore;
\end

\User Makefile options:
\end

\Default run parameters :         // Please let the parameters appear in this order
  Number of generations : 100     // NB_GEN
  Time limit: 0                   // In seconds, 0 to deactivate
  Population size : 100           //POP_SIZE
  Offspring size : 100 // 40%
  Mutation probability : 1        // MUT_PROB
  Crossover probability : 1       // XOVER_PROB
  Evaluator goal : maximise       // Maximise
  Selection operator: Tournament 2
  Surviving parents: 1//percentage or absolute
  Surviving offspring: 100%
  Reduce parents operator: Tournament 2
  Reduce offspring operator: Tournament 2
  Final reduce operator: Tournament 2

  Elitism: Strong                 //Weak or Strong
  Elite: 1
  Print stats: true                       //Default: 1
  Generate csv stats file:false
  Generate gnuplot script:false
  Generate R script:false
  Plot stats:true                         //Default: 0

  Remote island model: false
  IP file: ip.txt                 //File containing all the remote island's IP
  Server port : 2929
  Migration probability: 0.3

  Save population: false
  Start from file:false
\end
```

Conclusões

Nos foi possível observar a evolução da história, ainda que recente, da computação evolucionária e também algumas de suas subdivisões como os Algoritmo Genéticos, os Algoritmos de Estratégia Evolutiva, a Programação Evolucionária e a Programação Genética.

Além disso, se tornou notável os principais aspectos de funcionamento de cada categoria de algoritmos, tais como seus operadores genéticos, peculiaridades de desempenho e performance e também as diferenças entre cada tipo de algoritmo, evidenciando os diferentes tipos de uso para cada um deles.