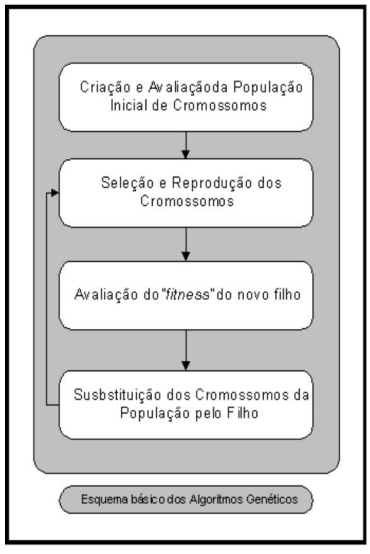
Algoritmos Genéticos

- Simulam o processo de evolução encontrado na natureza. São eficientes para problemas que busquem soluções ótimas ou aproximadamente ótimas. Para isto, manipulam população de soluções potenciais prum problema de busca ou otimização.
- Cada solução é codificada em uma cadeia de *bits*, sendo esta associada a um *fitness*, uma medida de qualidade. Quanto maior o *fitness*, maior as chances de se reproduzir e sobreviver à próxima geração.
- Durante o processo evolutivo, para cada indivíduo da população é dada uma medida de adaptação, isto é, uma que deve refletir sua habilidade de adaptação ao ambiente. Os mais adaptados são mantidos, enquanto outros são descartados.
- Os mantidos podem sofrer mutação e cruzamento, para garantir variabilidade de uma geração à outra. A reprodução pode ser assexuada (indivíduo reinserido diretamente.)
- O cruzamento (recombinação) é simulada por um mecanismo de crossover, que troca pedaços entre as cadeias de bits dos pais.
- A mutação provoca alterações esporádicas e randômicas nas cadeias de bits. Assim, reintroduz na população material perdido.



- A reprodução é repetida até que uma solução satisfatória seja encontrada.
- O resultado encontrado costuma ser o ideal ou aproximado. O critério costuma ser o número máximo de gerações, ou um tempo limite de processamento. Outro critério é quando a população estagnar, em que não se observa melhoria após algumas gerações.
- O processo de seleção e recombinação realizam, na verdade, uma espécie de busca local nas proximidades dos melhores indivíduos da população.

OPERAÇÃO DE SELEÇÃO

- Um indivíduo é selecionado probabilisticamente, baseado em seu grau de adaptação. Os que forem selecionados, são copiados sem alterações para a próxima geração (reprodução assexuada.)
- Quanto mais adaptado, mais chances há de passar à próxima geração.
- Na prática, é como se houvesse uma roleta: quanto maior o *fitness*, maior a área de um indivíduo na roleta e, portanto, maiores as chances da roleta parar em uma destas áreas e de selecionar o indivíduo (método chamado amostragem universal estocástica)
- Por conta disto, a seleção pode conter várias cópias de um mesmo indivíduo, e outros podem desaparecer completamente.

OPERAÇÃO DE CROSSOVER

- Permite o surgimento de novos indivíduos.
- Dois pais são selecionadas probabilisticamente, a partir de seus graus de adaptação.
- Como na operação anterior, mais adaptados têm mais chances de serem selecionados.
- A operação produz dois filhos, cada um com uma parte do material genético dos pais.
- Quanto melhor a eleição do mecanismo de seleção adequado, mais rigoroso é o tratamento matemático.
- É o operador predominante. Assim, é aplicado com probabilidade dada pela taxa de *crossover Pc*, que deve ser maior que a de mutação.
- Para operar o crossover, seleciona-se, entre os bits, um ponto entre 0 e tam-2, e os bits são trocados. Ao lado, exemplo com ponto = 3.

- Pode-se ter também multipontos, ou então uniforme, onde os bits a serem trocados são determinados a partir de um parâmetro global.
- Pode haver também um operador PMX (Partially-Matched Crossover.) O operador copia uma substring de um dos pais diretamente na mesma posição do filho. As

posições restantes são completadas com os valores originais da substring trocada que ainda não tenham sido utilizados, na mesma ordem da origem.

- Então, se temos os dois pais a seguir, o resultado do filho1 será 5 4 **4 6 3** 6 8 3, e do 2, 1 2 **1 7 2** 7 5 8

Pai1 = 1 2 4 6 3 7 5 8
Pai2 = 5 4 1 7 2 6 8 3

- Repare, no entanto, que, no filho1, ficou repetido os cromossomos 4, 6 e 3. Então, arrumando, teríamos algo como 5 * 4 6 3 * 8 * => 5 **1** 4 6 3 **7** 8 **2**, isto é, recolocamos o 172 na sua origem, mas em ordem diferente: substituímos os algarismos repetidos não da posição recém-colocada, mas da posição em que eles originalmente apareceram.

OPERAÇÃO DE MUTAÇÃO

- Permite criar novos indivíduos, mas deve ser usada com cautela; usá-la demais iria atrapalhar a busca pela solução, por provocar muitas oscilações.
- Pela mutação, garante-se que a busca nunca será zero.
- Seleciona, por adaptação, e depois escolhe, de forma aleatória, um caractere do cromossomo para ser invertido. O novo indivíduo é então inserido na nova população.
- O operador de mutação é aplicado a indivíduos com uma probabilidade dada pela taxa de mutação *Pm*. Esta costuma ser pequena.
- Se a representação for binária, para a mutação basta inverter o bit selecionado; para outros tipos de representações, é preciso pensar.
- Para decidir se haverá mutação ou não, escolhe-se aleatoriamente um *bit* do cromossomo; depois, um número aleatório entre 0 e 100 é gerado; se *Pm* for maior que ele, haverá a mutação. Depois, há duas formas de aplicar a mutação: ou só se inverte mesmo o bit selecionado, ou decide-se, ao azar, se o novo valor será 0 ou 1 (neste último caso, a taxa de mutação de fato é *Pm/2*.)

PARÂMETROS GENÉTICOS

- Certos parâmetros influenciam no comportamento dos algoritmos genéticos. São eles:
- **Tamanho da população:** uma população pequena oferece uma cobertura pequena de espaço de busca, o que dá mau desempenho; de igual forma, populações muito grandes exigem muitos recursos computacionais, muitas vezes desnecessários, pois já se acha a solução desejada em algum dos passos intermediários.
- **Taxa de cruzamento:** mais alta, mais rapidamente novas estruturas cromossômicas serão introduzidas; no entanto, se for mais alta que o necessário, pode fazer com que se percam organismos de alta aptidão. Mas, também, se a taxa for lenta, o algoritmo torna-se também muito lento.
- **Taxa de mutação:** se for muito alta, o algoritmo se torna praticamente aleatório. O ideal é que seja um valor baixo, apenas para que não se esgote o espaço de busca, e de forma que se chegue ao resultado da forma mais rápida possível.
- **Intervalo de geração:** controla a porcentagem da população que será substituída na próxima geração.
- **Probabilidades:** recomenda-se deixar alta a probabilidade de recombinação (>0.7) e baixa a de mutação (<0.02.)

COMO REPRESENTAR

- Geralmente, indivíduos são representados por vetores binários. Cada elemento do vetor denota a presença (1) ou ausência de alguma característica.
- **Genótipo:** conjunto de ausências e presenças de características.
- Outra representação é converter um valor inteiro em binário.
- Exemplo: se há uma função de 3 variáveis, cada uma pode ser representada por 10 bits. Destarte, o cromossomo fica com 30 bits e há 3 genes possíveis *(nota: não saquei...)*
- Menor o alfabeto de representação, melhor. Por isto, geralmente as combinações binárias são as usadas.

- Alta pressão seletiva faz a diversidade cair rápido, fazendo que a população convirja nas primeiras gerações, algo ruim para se encontrar boas soluções. Baixa pressão seletiva, por sua vez, pode tornar o processo de busca extremamente lento.
- Para melhores resultados, o algoritmo pode usar escalonamento para evitar que super-indivíduos dominem a população no início da busca.

CONVERGÊNCIA

- É uma progressão uniforme. Diz-se que um gene é convertido quando 95% da população possui o mesmo valor. A população converge quando todos os genes de cada indivíduo tiver convergido.

FUNÇÃO FITNESS

- A aptidão se determina por meio do cálculo da função objetivo, a qual depende das especificações do projeto. Esta função, analisando um cromossomo, atribui-lhe um valor numérico que indica seu nível de adaptação.
- Esta função deve ser de cálculo rápido, pois roda a todos indivíduos de todas gerações.

EXERCÍCIOS DE EXEMPLO

1) **O valor máximo de $f(x) = -x^2 + 33x$, $x \in [0,31]$ e $x \in Z$.**

Como o maior valor possível é 31, precisaremos de 5 bits de representação, pois 2^5 = 32. Usando uma população de 4 indivíduos, escolhemos alguns valores aleatórios, em geral buscando algo que, de olho, já dê bom resultado, como: 00001 (1 em decimal) e 01001 (9 em decimal) e 10110 (22 em decimal) e 11100 (28 em decimal.)

Feito isto, substituímos os valores escolhidos na função, vemos qual o resultado (que vai ser a função-objetivo, FO) e vemos quanto a soma destes valores dá, e quantos % esta

soma é dos valores totais. Estes percentuais acumulados servem para elaborar a roleta de seleção.

Cadeia	Função-Objetivo	% do total	% acumulado
00001 (1)	32	5,1	5,1
01001 (9)	216	34,3	(5,1+34,3)=39,4
10110 (22)	242	38,4	(39,4+38,4)=77,8
11100 (28)	140	22,2	(77,8+22,2)=100
TOTAL:	630	100	

Como há 4 indivíduos, devemos gerar 4 números aleatórios entre 0 e 100, a fim de selecionar, na roleta, quais indivíduos serão escolhidos para reprodução. Assim, suponha que os números aleatórios escolhidos tenham sido: 42, 31, 70 e 23.

A cadeia 3 (10110) está na faixa de 39,4 a 77,8, logo, 42 seleciona ela.

A cadeia 2 (01001) está na faixa de 5,1 a 39,4, logo, 31 seleciona ela.

A cadeia 3 (10110) está na faixa de 39,4 a 77,8, logo, 70 seleciona ela.

A cadeia 2 (01001) está na faixa de 5,1 a 39,4, logo, 23 seleciona ela.

Agora, seleciona-se aleatoriamente *q* pares de cadeia, sendo este valor *q* baseado no *Pc* (taxa de crossover.) Nos *slides*, XL não dá maiores explicações de como estes se relacionam.

Escolhidos os pares, deve-se gerar um número aleatório *k* entre 0 e tamanhoDaCadeia-2.

Suponha que as cadeias misturadas são a 2 (01001) e a 3 (10110), e k=2.

O corte será realizado assim: cadeia 2 (01|001) e 3 (10|110.) O primeiro filho será (01|110) e o segundo será (10|001), isto é, em decimal, 14 e 17.

Suponha que os outros *q* pares selecionados foram ambos cadeia 3, o que obviamente produzirá 2 filhos iguais, 10110.

Agora, aplique-se o fator de mutação. Se temos ele = 2%, então escolhemos dois números fixos; depois geramos um número aleatório entre 1 e 100 e vemos se este aleatório bate com algum dos fixos. Se bater, o bit é invertido. Isto deve ser feito em cada bit de cada indivíduo.

Como temos 20 bits, usei um *script* para gerar 20 números aleatórios, sendo eles: [93, 37, 17, 49, 3, 38, 83, 35, 21, 92, 27, 52, 57, 80, 99, 97, 92, 82, 92, 58.]

Agora, como para cada bit teremos 2 valores fixos (há 2% de chance de mutar), selecionei 20 pares de dois números fixos aleatórios: [[48, 41], [46, 60], [91, 65], [69, 27], [36, 4], [77, 25], [1, 53], [30, 72], [4, 74], [57, 93], [3, 9], [54, 29], [51, 3], [22, 14], [26, 59], [0, 55], [85, 31], [42, 46], [90, 11], [46, 76]].

Depois, comparei se algum dos dois números fixos batia com o número correspondente dos bits, isto é, se, por exemplo, 48 ou 41 eram iguais à 93, e repeti essa operação para cada par. Infelizmente, com uma mutação tão baixa, nenhum bateu (vou confiar no meu programa que nenhum bateu.)

Assim, sem mutar nenhum bit, a próxima geração de indivíduos são as cadeias:

Cadeia 1: 01110 (14 em decimal)

Cadeia 2: 10001 (17 em decimal)

Cadeia 3: 10110 (22 em decimal)

Cadeia 4: 10110 (22 em decimal)

Assim, remontaríamos as tabelas novamente, e realizaríamos todas as contas de novo, até que se chegasse a algum resultado satisfatório ou a algum critério de parada.