Computação Evolucionária

Apresentação:

Este resumo tem como objetivo guiar os estudantes no estudo da Computação Evolucionária, ramo da Computação Natural, que é uma área da ciência que busca, através de técnicas inspiradas na Natureza, o desenvolvimento de sistemas inteligentes que imitam aspectos do comportamento humano, tais como: aprendizado, percepção, raciocínio, evolução e adaptação.

Na elaboração deste documento apresentamos aos leitores, de início, a contextualização do assunto estudado, bem como, as áreas de aplicação dos algoritmos de Computação Evolucionária. Após esta apresentação definiremos a estrutura básica de um algoritmo de Computação Evolucionária, caracterizando cada técnica com o seu respectivo idealizador e, por fim, detalharemos cada técnica de uma maneira concisa, tendo em vista, que o objetivo principal do documento é prover as informações de forma direta e de fácil aprendizado.

Contextualização:

A Computação Evolucionária é uma área que abrange métodos computacionais inspirados na Teoria da Evolução das Espécies, do naturalista inglês Charles Robert Darwin essencialmente no conceito de Seleção Natural.

Nos sistemas inspirados por este paradigma, as possíveis soluções para o problema são representadas sobre a forma de indivíduos de uma população. Estes indivíduos evoluem de acordo com regras de seleção e operadores genéticos.

Áreas de Interesse:

Algoritmos de Computação Evolucionária (CE) têm sido aplicados com sucesso a problemas de busca e otimização das diversas áreas de aplicação e com diversos níveis de dificuldade. No entanto, o interesse principal dos algoritmos evolucionários é a resolução de problemas complexos, pois não seria interessante optar por uma solução inteligente, tendo em vista o custo computacional ser maior do que um algoritmo de Computação Clássica.

Algoritmos evolucionários são capazes de manter um nível adequado de diversidade na população, além de aprender sobre a estrutura do problema ao qual são aplicados, capturando interações importantes e mantendo-as ao longo do processo evolutivo. Para capturar estas informações sobre o problema e assim guiar o processo evolutivo, cada algoritmo adota um mecanismo de aprendizado; a indução de classificadores e o aprendizado de regras simbólicas como sendo os mecanismos adotados por algoritmos de CE. Entretanto, o custo computacional do processo de aprendizado escolhido é um fator limitante de alguns algoritmos, já que esta etapa é repetida a cada nova geração. Por outro lado, outros algoritmos são mais baratos computacionalmente, mas apresentam menor capacidade de prospecção do espaço de busca, exigindo a adoção de estratégias híbridas. Estes fatores evidenciam que ainda existe espaço para novas propostas em Computação Evolucionária.

Estrutura Básica:

As três premissas básicas de funcionamento de um algoritmo evolucionário são: criação de uma população de soluções, criação de uma função de avaliação e criação dos operadores de seleção, recombinação e mutação. A seguir detalharemos as ideias do algoritmo, indicando como cada um influi na resolução do problema proposto:

a) População de soluções

É necessário que haja uma população inicial de soluções possível para o problema proposto. Essa população pode ser gerada aleatoriamente ou através de alguma técnica. Cada indivíduo dessa população terá registrado em si os parâmetros que descrevem a sua respectiva solução para o problema.

b) Função de avaliação

A função de avaliação terá o trabalho de julgar a aptidão de cada indivíduo da população. Ela não precisará deter o conhecimento de como encontrar a solução do problema. Somente precisará julgar a qualidade da solução que está sendo apresentada por aquele indivíduo. A função é definida através da codificação das soluções para o problema para que se possa avaliar se o indivíduo está ou não apto.

c) Operadores seleção, recombinação e mutação.

É necessário gerar novas populações para que se encontre o mais apto. Essas são chamadas de gerações e são obtidas através da aplicação de três operadores: seleção, recombinação e mutação. Na seleção escolhem-se os indivíduos mais aptos para gerarem descendentes. Essa reprodução pode ocorrer de duas formas: um indivíduo gera a descendência ou um par de indivíduos geram a descendência. O primeiro caso simula uma reprodução assexuada e o segundo uma reprodução sexuada. É importante destacar que os descendentes serão diferentes de seus antecedentes. Na recombinação ocorre a troca de material genético entre o par de antecedentes definindo assim a carga genética dos descendentes. Dessa forma, cada descendente desse par herdará uma parte do material genético de seus antecedentes escolhidos de forma aleatória. E na mutação ocorrem mudanças aleatórias no material genético do indivíduo. Dessa forma o indivíduo estará mais apto dentro daquela população. A simulação da passagem de gerações ocorre na repetição deste ciclo, ou seja, a cada iteração.

Técnicas Estudadas:

Neste resumo serão apresentadas seis técnicas:

- Algoritmos Genéticos – proposto por Jonh Holland em 1975, definindo como “Algoritmos genéticos são modelos computacionais que imitam o mecanismo da Evolução natural para resolver problemas de otimização.”

- Programação Genética - proposto por Arthur Samuel na década de 50, refletindo sobre a questão "Como os computadores podem aprender a resolver problemas sem estarem explicitamente programados? Em outras palavras, como os computadores podem ser criados para fazer o que precisava ser feito,sem ser dito exatamente como fazê-lo?

- Estratégias Evolucionárias - proposta por Rechenberg (1965), Schwefel (1965) e Peter Bienert (1964) na Universidade Técnica de Berlim por volta de 1964, o objetivo da nova técnica era tratar problemas de otimização em mecânica de fluidos, e em seguida passaram a tratar problemas de otimização de funções de forma mais genérica, enfocando o caso das funções reais
- Evolução Diferencial - 1995 – Foi apresentado em 1995 por Rainer Storn e Kenneth Price seus primeiros resultados iniciais no ICSI. Participaram no ano seguinte do concurso de otimização evolucionária, onde terminaram em terceiro lugar, perdendo para métodos não tão versáteis e finalmente em 1997, foi escrito um artigo sobre o trabalho ganhando respeito na comunidade científica internacional.

- Algoritmos Culturais - propostos por Robert Reynolds em 1994, como um complemento a metáfora utilizada na computação evolutiva, a qual se concentra nos aspectos genéticos da evolução. Definindo “A evolução cultural habilita as sociedades a evoluir ou a se adaptar ao seu ambiente em taxas que excedem as da evolução biológica baseada somente na herança genética”.

- Coevolução – proposto por Jonh Holland em 1975, também criador dos Algoritmos Genéticos e consagrado por Hillis (1990), no seu trabalho de ordenação de redes, onde o resultado da ordenação da rede utilizando o princípio coevolutivo superou muito a resolução do mesmo problema através do algoritmo genético

Algoritmos Genéticos

Componentes de um Algoritmo Genético:

Problema a ser resolvido

É o problema que o algoritmo estará proposto a resolver, algoritmos genéticos são particularmente aplicados em problemas complexos de otimização: problemas com diversos parâmetros ou características que precisam ser combinadas em busca da melhor solução; problemas com muitas restrições ou condições que não podem ser representadas matematicamente; e problemas com grandes espaços de busca.

Representação das Soluções de Problema

A representação das possíveis soluções do espaço de busca de um problema define a estrutura do cromossoma a ser manipulado pelo algoritmo. A representação do cromossoma depende do tipo de problema e do que, essencialmente, se deseja manipular geneticamente. Os principais tipos de representação são:

Representação	**Problemas**
Binária Numéricos	Inteiros
Números Reais	Numéricos
Permutação de Símbolos	Baseados em Ordem
Símbolos Repetidos	Grupamento

Decodificação do Cromossoma

A decodificação do cromossoma consiste basicamente na construção da solução real do problema a partir do cromossoma. O processo de decodificação constrói a solução para que esta seja avaliada pelo problema. A vantagem da representação binária é a fácil transformação para inteiro ou real.
Por exemplo, seja o problema de encontrar o valor máximo da função $f(x) = x^2$, x inteiro [0,63]. Podemos representar as soluções do problema através de um cromossoma de 6 bits.

C1 0 0 1 0 0 1 representa x=9
C2 0 0 0 1 0 0 representa x=4

Avaliação

A avaliação é o elo entre o AG e o mundo externo. A avaliação é feita através de uma função que melhor representa o problema e tem por objetivo fornecer uma medida de aptidão de cada indivíduo na população corrente, que irá dirigir o processo de busca. A função de avaliação é para um GA o que o meio ambiente é para seres humanos.
Funções de avaliação são específicas de cada problema. No exemplo, a função matemática $f(x) = x^2$ mede aptidão de cada indivíduo. Na Tabela abaixo, C1 é um indivíduo mais apto que C2.

Cromossoma	**X**	**f(x)**
C1 0 0 1 0 0 1	9	81
C2 0 0 0 1 0 0	4	16

Seleção

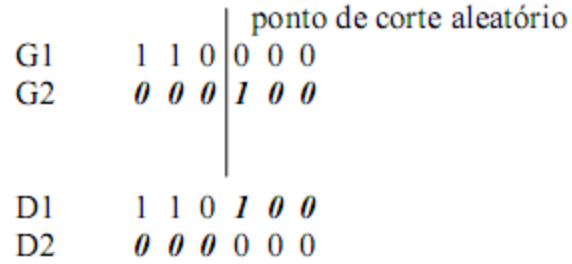
O processo de seleção em algoritmos genéticos seleciona indivíduos para a reprodução. A seleção é baseada na aptidão dos indivíduos: indivíduos mais aptos têm maior probabilidade de serem escolhidos para reprodução. Assim, se f_i é a avaliação do indivíduo i na população corrente, a probabilidade p_i do indivíduo i ser selecionado é proporcional a

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_i}$$

Onde N é o número de indivíduos na população. A seleção em Algoritmos Genéticos é tipicamente implementada por uma roleta onde cada indivíduo é representado por uma fatia proporcional a sua aptidão relativa.

Operadores Genéticos

Indivíduos selecionados (e reproduzidos na população seguinte) são recombinados através do operador de crossover (com uma probabilidade p_c). O operador de crossover é considerado a característica fundamental dos GAs. Pares de genitores são escolhidos aleatoriamente da população, baseado na aptidão, e novos indivíduos são criados a partir da troca do material genético.
Os descendentes serão diferentes de seus pais, mas com características genéticas de ambos os genitores. Por exemplo:



Na sua forma mais simples, o crossover de um ponto de corte (one-point crossover) corta os dois genitores em uma posição aleatoriamente escolhida, criando dois possíveis descendentes:
D1 é um cromossoma mais apto que seus genitores, todavia D2 é um indivíduo medíocre (baixa avaliação em $f(x) = x^2$).
Os cromossomas criados a partir do operador de crossover são então submetidos a operação de mutação (com uma probabilidade p_m). Mutação é um operador exploratório que tem por objetivo aumentar a diversidade na população.

Inicialização da População

A inicialização da população determina o processo de criação dos indivíduos para o primeiro ciclo do algoritmo. Tipicamente, a população inicial é formada a partir de indivíduos aleatoriamente criados. Populações iniciais aleatórias podem ser semeadas com bons cromossomas para uma evolução mais rápida, quando se conhece, a priori, o valor de boas "sementes".
Uma técnica eficiente para se encontrar por GA boas soluções em um problema, consiste em executar evoluções (rodadas) sucessivas, semeando-se a população inicial da evolução seguinte com as melhores soluções encontradas na anterior.

Parâmetros e Critérios de Parada

Em um algoritmo genético vários parâmetros controlam o processo evolucionário:

• Tamanho da População - número de pontos do espaço de busca sendo considerados em paralelo a cada ciclo.
• Taxa de Crossover - probabilidade (p_c) de um indivíduo ser recombinado com outro.
• Taxa de Mutação - probabilidade (p_m) do conteúdo de uma posição/gene do cromossoma ser alterado.
• Número de Gerações - total de ciclos de evolução de um GA.
• Total de Indivíduos - total de tentativas em um experimento (tamanho da população x número de gerações)

Os dois últimos parâmetros são em geral empregados como critério de parada de um algoritmo genético.

Pseudo código:

```
begin
        // começar em um tempo inicial
        t := 0;
        // inicializar uma população (usualmente aleatória) de indivíduos
        initPopulation P(t);
        // avaliar a adequação de todos os indivíduos na população inicial
        evaluate P(t);
        // testar o critério de término (tempo, adequação, etc.)
        while not done do
                // incrementar o contador de tempo
                t := t + 1;
                // selecionar sub-população para a produção de descendência
                P' := selectParents P(t);
                // recombinar os genes da sub-população selecionada
                recombine P'(t);
                // perturbar a população estocasticamente
                mutate P'(t);
                // avaliar a nova adequação
                evaluate P'(t);
                // selecionar os sobreviventes da geração corrente
                P = survive P,P'(t);
        od
    end AE
```

Aplicações:

- Otimização combinatória
  - Oito rainhas
  - Caixeiro viajante
- Otimização com várias dimensões
- Otimização Multiobjetivo

Programação Genética

Componentes de um Algoritmo de Programação Genética:

Problema a ser resolvido

Um algoritmo da PG está preocupado em encontrar soluções que sejam elas próprias um programa passível de execução. Para Eiben PG pode ser vista como "programação de computadores através da seleção natural"ou "a evolução natural de programas de computadores"

Representação

Em PG os indivíduos são representados por árvores. Estas árvores representam expressões que possuem uma sintaxe predefinida. Dependendo do problema considerado, esta sintaxe pode representar uma expressão aritmética, fórmulas da lógica de 1a ordem, ou código-fonte de alguma linguagem de programação.
Sob uma ótica estritamente técnica, pode-se dizer que PG é simplesmente uma variação de um AG que usa uma estrutura de dados diferente: os indivíduos são árvores.

- Uma fórmula aritmética: $2x^{2+}\ 4x + 10$
- Uma fórmula lógica: $(p \vee true) \rightarrow (p \wedge q) \vee (\neg q)$
- Um programa escrito em uma linguagem de programação de alto nível,

Inicialização

O método mais comum de inicialização em PG é denominado de ramped half-and-half. Nessa estratégia define-se a profundidade máxima Dmax das árvores e então cada indivíduo da população inicial é criado usando os conjuntos de terminais (T) e não terminais ou funções (F) com igual probabilidade. Os métodos mais comuns para geração dos ramos das árvores são:
Método completo: cada ramo da árvore tem profundidade Dmax. O conteúdo dos nós de profundidade d são escolhidos de F se d < Dmax ou de T se d = Dmax.
Método de crescimento: os ramos podem ter profundidades diferentes,limitadas a Dmax.

Recombinação

Os operadores de recombinação criam descendentes através da troca de material genético entre os pais selecionados. Tecnicamente, em algoritmos de PG é um operador binário que cria duas novas árvores filhas através de duas árvores pais.
O tipo de recombinação mais comum em PG, o crossover de subárvores, seleciona um nó nas árvores pai de forma aleatória e troca as subárvores relacionadas entre os indivíduos. Assim com ocorre com o operador de mutação, as árvores lhas geradas podem ter uma profundidade diferente dos ancestrais. O operador de recombinação deve considerar dois parâmetros:

- A probabilidade de realizar a recombinação entre dois indivíduos.
- A probabilidade de selecionar um dos nós em cada árvore pai para servir como ponto de corte.

Mutação

Em princípio, o objetivo da mutação em PG seria o mesmo dos outros ramos de CE, ou seja, criar um indivíduo novo a partir de um existente através de uma pequena alteração randômica neste indivíduo, produzindo diversidade.
A forma mais comum de alcançar este efeito em PG é através da substituição de uma subárvore da

solução atual, escolhida ao acaso, por uma outra subárvore, gerada aleatoriamente. Essa geração de subárvore ocorre de modo semelhante a geração de um indivíduo da população inicial.
A mutação em PG deve considerar dois parâmetros:

- A probabilidade de realizar uma mutação no indivíduo considerado.
- A probabilidade de selecionar um dos nós da árvore representativa do indivíduo para ser substituído.

Seleção

- Seleção dos Pais

A PG geralmente usa seleção proporcional à aptidão, contudo, devido ao tamanho tópico de uma população ser na ordem de centenas, quiçá milhares de indivíduos, um método chamado over-selection geralmente é usado para populações acima de 1000 indivíduos.
Nesta metodologia a população é ordenada de acordo com a aptidão e então separada em dois grupos. O primeiro grupo conterá x% indivíduos e o segundo grupo (100 - x%). Na etapa de seleção dos pais, 80% dos selecionados devem ser provenientes do primeiro grupo e os 20% restantes do segundo grupo. Estes percentuais foram obtidos empiricamente e dependem do tamanho da população.

- Seleção de Sobreviventes

Nos algoritmos típicos de PG os pais são substituídos pelos descendentes, ou seja, o tempo de vida de um indivíduo é de uma geração e não há mecanismos de elitismo. Evidentemente, não há nenhuma restrição técnica para fixar essa estratégia, apenas é uma convenção histórica na área de PG.

Engorda

Uma das observações feitas sobre os operadores de recombinação e mutação é que os descendentes gerados podem aumentar o tamanho da árvore representativa da solução. Este efeito de crescimento das soluções durante da execução de um algoritmo em PG é muito comum e é conhecido como engorda (bloat). Há muitos estudos para o entendimento do processo e sobre medidas para contornar o problema.
Uma das correntes de pesquisa relaciona o crescimento excessivo dos indivíduos com o fenômeno de overfitting de uma RNA. Ou seja, as soluções do PG estariam se acostumando excessivamente com os pares de entrada-saída do problema que são utilizados na função de avaliação.

Aplicações

Pode-se dizer que as aplicações reais de PG variam desde a geração de pequenas sub-rotinas, até problemas em tempo real(sistemas de controle para robótica), passando por aplicações para análise de séries temporais e mineração de dados. Todas as aplicações de aproximação de funções através de Redes Neurais Artificiais (RNA) são candidatas óbvias para serem resolvidas através de PG. Também pode-se incluir aqui sistemas baseados em regras, como provas automáticas de teoremas.

Estratégias Evolucionárias

Problema a ser resolvido

Tratamento de problemas relacionados a otimização em mecânica de fluidos, e em seguida passaram a tratar problemas de otimização de funções de forma mais genérica, enfocando o caso das funções reais.
Os primeiros algoritmos de Estratégia Evolucionária (EE´s) operavam com um único indivíduo na população, sujeito à mutação e seleção. Uma ideia importante introduzida nos algoritmos mais recentes é a adaptação online (auto-adaptação) dos parâmetros da estratégia durante o processo evolutivo, através da introdução dos mesmos na representação genética dos indivíduos.

Representação de indivíduos

Assim como nos algoritmos genéticos, a representação dos EE´s é feita através de gens que formam um cromossomo que basicamente consistem em três partes:

- Variáveis Objeto (X1, X2,..., Xn) – é o conjunto formado por valores reais distribuídos aleatoriamente que representa uma solução candidata , ou seja um ponto no espaço de busca.

Parâmetros da Estratégia - referem-se aos parâmetros que controlam o processo evolutivo de busca, como taxas de mutação, desvios padrões das mutações, probabilidades de recombinação, etc. Os parâmetros mais utilizados são:

- Passo da Mutação: (σ1,σ2, ..., σns) – é o conjunto que representa os desvios padrões a serem utilizados no operador de mutação.
- Ângulos de Rotação (α1, α2, ..., α nα) - É o conjunto que representa os ângulos de rotação.

Inicialização do Algoritmo

A inicialização consiste em atribuir valores aleatórios para cada gene do cromossomo de cada indivíduo, certificando-se de que seu genótipo esteja inicializado dentro das fronteiras de restrição do problema. Os parâmetros de estratégia também são inicializados.

Auto adaptação

A ideia de auto adaptação consiste na evolução dos parâmetros da estratégia em adição à evolução dos atributos da estrutura de dados. Como exemplo de autoadaptação temos:
Dado um indivíduo v = (x,σ,α) composto pelo vetor de atributos x, os conjuntos σ e α de parâmetros da estratégia, o processo de auto adaptação é geralmente implementado efetuando-se primeiro (a recombinação e) mutação (de acordo com alguma função densidade de probabilidade) dos vetores de parâmetros σ e α, resultando em σ’ e α’, e depois utilizando os vetores atualizados de parâmetros da estratégia σ’ e α’ para (recombinar e) mutacionar o vetor de atributos x, resultando em x’.

Operadores

Seleção

Assim como nos algoritmos genéticos, há várias técnicas já abordas sobre a seleção de indivíduos como roleta (Monte Carlos) e torneio.

Recombinação

Além da recombinação discreta (crossover uniforme) são usadas mais duas abordagens:

- Recombinação intermediária global: média aritmética dos parâmetros de cada um dos pais.

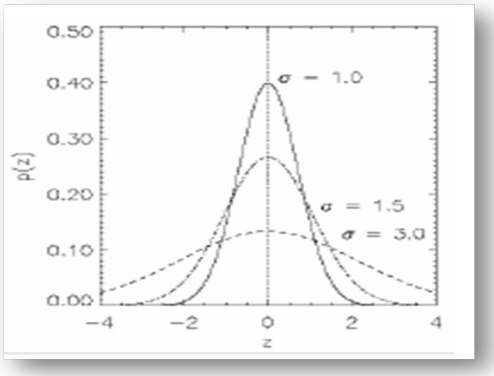
Dados dois indivíduos a = (xa,σa) e b = (xb,σb)
xi' = ½.(xa,i + xb,i), "i ϵ {1,...,l}
σi' = ½.(σa,i + σb,i), "i ϵ {1,...,l}

- Recombinação intermediária local: combinação linear convexa dos vetores correspondentes aos pais

Dados dois indivíduos a = (xa,σa) e b = (xb,σb)
xa' = axa + (1-a)xb,
xb' = (1-a)xa + axb, "i ϵ {1,...,l}, e a ϵ [0,1] é um número aleatório se aplica também a σ.
Os operadores de recombinação podem gerar um ou mais indivíduos (filhos) partindo de dois ou mais pais.

Mutação

Para os EE´s as mutações são os operadores mais utilizados e proporcionam a esta técnica uma versatilidade na resolução de problemas de combinatória numérica. A fundamentação deste operador é a modificação dos valores pela adição aleatória de ruído guiado por uma distribuição normal, ou seja, por uma função gaussiana de média zero e desvio padrão σ. Por esse motivo este operador é denominado de mutação gaussiana, como a distribuição demonstrada na figura abaixo.



Este procedimento de mutação está de acordo com a observação biológica de que pequenas variações ocorrem com maior freqüência do que grandes variações, e de que os filhos herdam características dos pais, ou seja, são parecidos com eles.
Como idéia chave e utilização da auto-adaptação temos:

- σ é parte do cromossomo < x1,…,xn, σ >
- σ também é mutacionado dentro de σ'

Com isso temos que o passo da mutação σ está co-evoluindo com a solução x.
Com a auto-adaptação surgem efeitos derivados do processo de mutação, onde temos:

- < x, σ > → < x', σ'>

Como critério para que haja a auto-adaptação é necessário garantir a ordem dos eventos.

- Primeiro é realizada a evolução dos parâmetros do indivíduo: σ →σ'
- Então é feita a evolução das variáveis objeto utilizando os parâmetros já evoluídos anteriormente: x → x' = x + N(0,σ')

Após a evolução das variáveis objeto, é realizada a avaliação em dois processos distintos:

- No primeiro momento é avaliada a qualidade do conjunto de variáveis objeto: x' é bom se f(x') é bom
- No segundo momento é avaliada a qualidade do conjunto de parâmetros: σ' é bom se o x' que este criou é bom

Caso não haja hierarquia com relação à ordem de evolução entre as partes do cromossomo não é garantido que o processo funcione como esperado.

Estratégias

São metodologias desenvolvidas com o objetivo de descrever como se dará o mecanismo de evolução da população. Há vários tipos de estratégias, onde se destacam os seguintes:

Dois membros: (1 + 1)-EE

Nesta estratégia a população é composta por um único indivíduo, e apenas o operador genético de mutação é utilizado para geração de diversidade. A representação do indivíduo da população é dada por um par de vetores reais, ou seja, $v = (x,\sigma)$ onde cada valor do vetor x representa um parâmetro a ser otimizado e cada valor do vetor σ representa o desvio padrão a ser utilizado na atualização de x. As mutações são realizadas atualizando-se x da seguinte forma (Mutação Gaussiana):

$xt+1= xt + N(0,\sigma)$

Este procedimento de mutação está de acordo com a observação biológica de que pequenas variações ocorrem com maior freqüência do que grandes variações, e de que os filhos herdam características dos pais, ou seja, são parecidos com eles.
O filho (indivíduo mutacionado) é aceito na nova geração se e somente se ele possuir um fitness melhor do que o pai (e for factível). Apesar de existir um único indivíduo na população, este procedimento é denominado de estratégia evolutiva de dois membros, pois o filho compete com o pai na sobrevivência para a próxima geração.

Multimembros: (μ + 1)-EE

A partir desta estratégia foi introduzido o conceito de população no algoritmo, Rechenberg propôs as que $\mu > 1$ pais podem participar na geração de um único filho. Este operador é denominado de operador de recombinação discreto e é equivalente ao crossover uniforme em algoritmos genéticos e a mutação é idêntica a dois membros (1+1)-EE.
O operador de seleção remove o indivíduo menos apto (com menor fitness) dentre os pais e o filho gerado.

Multimembros: (μ + ʎ)-EE

A estratégia (μ+ʎ)-EE foi proposta inicialmente, onde μ pais produzem ʎ filhos e a população μ+ʎ é posteriormente reduzida para μ indivíduos. A seleção opera no conjunto união de pais e filhos. Assim, os pais sobrevivem até que filhos com fitness superiores a eles sejam produzidos.
A partir desta nova abordagem os parâmetros de controle σ deixam de ser regidos pela regra de 1/5 e passam a ser evoluídos juntamente com o conjunto objeto e são atualizados através do operador de mutação gaussiana. Com isso os desvios padrões sofrem auto-adaptação e evoluem com a solução.

Multimembros: (μ , ʎ)-EE

A partir da utilização da estratégia (μ+ʎ)-EE foram detectados alguns problemas, como:

- Em problemas com superfície de fitness dinâmicas (que variam ao longo do tempo), a estratégia (μ+ʎ)-EE pode ficar presa em um ótimo que não é mais um ótimo da superfície de fitness atual.
- O mesmo pode ocorrer na presença de ruído

Pseudo codigo (1+1) – EE:

1. Escolha um único vetor pai que contém m parâmetros X = (x1, x2, ..., xm). Cada parâmetro é
2. escolhido por um processo randômico e satisfaz as restrições do problema.
3. Crie um novo descendente por meio de mutação. Para obter a mutação nesse método, adicione um
4. vetor randômico do tamanho de X com distribuição normal (média zero e variância σ):
5. $X' = X + N(0, \sigma)$
6. Se for o caso, aplique a regra de sucesso 1/5.
7. Compare as soluções para X e X'. Escolha o melhor membro para a próxima geração.
8. Repita os passos 2 e 3 até encontrar uma solução satisfatória, ou até que tenha se esgotado o tempo computacional (ou número de gerações).

Aplicações:

O uso de EE está bastante associado a problemas de combinação numérica e esta característica tem sido fator motivador para a utilização desta técnica para resolução de problemas envolvendo design e modelagem de protótipos automotivos e de aerodinâmica.

Evolução Diferencial

Problema a ser resolvido

Otimização estocástica de funções não-lineares e não-diferenciável no espaço contínuo baseada em populações de indivíduos, para tentar resolver problemas de ajuste polinomial de Chebychev. Tal técnica se mostrou não só capaz de resolver os problemas de ajuste polinomial de Chebychev, como também para várias outras funções de testes, mesmo que possuam restrições do espaço de busca.

Representação individual

De acordo com os outros algoritmos evolucionários, cada posição de um indivíduo da população representa uma possível solução no espaço de busca. Isso permite a Evolução Diferencial (ED) obter uma boa representação de todo o espaço de busca apenas com as posições iniciais dos indivíduos. Cada indivíduo é codificado em valores reais em forma de um vetor de posição, onde cada componente desse vetor representa uma das variáveis da solução.
O uso de números do tipo ponto flutuantes na representação das componentes dos valores codificados no ED traz vantagens como uma maior precisão dos resultados, sem a perda devido à representação e uma melhor capacidade de representação do domínio do problema. Essas vantagens fazem com que a representação ponto flutuante seja uma abordagem bastante atrativa.

Inicialização

Um dos pontos mais sensíveis do algoritmo de Evolução Diferencial é a inicialização dos indivíduos pelo espaço de busca. Para o ED funcionar satisfatoriamente, os indivíduos devem ser bem distribuídos pelo espaço de busca, podendo ser utilizado da distribuição aleatória e uniforme. Caso a inicialização seja realizada de forma equivocada como, por exemplo, definida muito longe dos limites do espaço de busca, esta pode comprometer a qualidade da solução encontrada. Devido a isso, o algoritmo nem sempre irá convergir para o melhor ponto da função fitness.

Operadores

No caso dos algoritmos de Evolução Diferencial os operadores de mutação e cruzamento são utilizados, o que basicamente diferencia ED das outras técnicas é o operador de mutação que é aplicado primeiro em um vetor experimental para depois realizar o cruzamento e na produção da sua aleatoriedade que não é regido por um número aleatório de distribuição uniforme.

Mutação

Este operador produz um vetor experimental $u_i(t)$ a partir de outros três indivíduos da população. O primeiro passo consiste na escolha do indivíduo destino $x_{i1}(t)$ como sendo um indivíduo da população diferente do atual. Os critérios dessa escolha podem ser a aleatoriedade, o melhor indivíduo (que possui o melhor valor de fitness) ou o pior indivíduo (que possui o pior valor de fitness), etc.
Uma vez que o indivíduo destino tenha sido escolhido, escolhem-se mais dois outros indivíduos diferentes $x_{i2}(t)$ e $x_{i3}(t)$ na população. Esses serão exclusivamente utilizados para o cálculo do vetor experimental, que é realizado como sendo a diferença vetorial desses indivíduos escolhidos (x_{i2} e x_{i3}), amplificada pelo fator de escala (β) sendo somada ao vetor da posição do indivíduo destino (x_{i1}), como pode ser visto na expressão abaixo.

$$u_i(t) = x_{i1}(t) + \beta\,(x_{i2}(t) - x_{i3}(t))$$

Cruzamento

O operador de cruzamento proporciona a diversidade no ED, que é realizada através da combinação entre o vetor experimental $u_i(t)$ e o vetor pai $x_i(t)$, tendo como resultado um vetor filho $x'_i(t)$. A recombinação é feita através da seguinte equação:

$$x'_{ij} = \begin{cases} u_{ij}(t), & Se\ j \in J \\ x_{ij}(t), & Caso\ contrário \end{cases}$$

Onde x_{ij} é referente a j-ésima componente do vetor $x_i(t)$, e J é o conjunto de pontos de cruzamentos, ou seja, os índices referentes às posições do vetor que sofrerão perturbação. Para que a componente seja trocada, um número aleatório é sorteado e ele é comparado ao parâmetro de probabilidade de recombinação (P_r). Caso o número seja menor que o parâmetro de recombinação, a componente do vetor experimental será utilizada. Caso contrário, a própria componente do pai sem modificações será utilizada.
Para a escolha dessas componentes, existem duas técnicas comumente utilizadas, são elas:
• Cruzamento Binomial: Onde os pontos de cruzamento são selecionados aleatoriamente dentro da dimensão do problema. Ou seja, cada uma das dimensões gera um número aleatório e compara o número ao parâmetro de recombinação (P_r).
• Cruzamento Exponencial: Os pontos de cruzamento são selecionados sequencialmente dentro das dimensões do problema, em forma de uma lista circular. Ou seja, para cada dimensão é sorteado um número aleatório, que caso seja menor que o parâmetro de recombinação (P_r) então todas as componentes a partir daquela em diante serão trocadas, até que o número sorteado seja maior que o parâmetro de recombinação.

Seleção

O operador de seleção é o operador mais simples da técnica, sendo utilizado apenas para escolha dos indivíduos que farão parte da próxima geração. A escolha é realizada através da avaliação do fitness do vetor escolhido com o vetor experimental:
• Caso o vetor experimental possuir fitness melhor que o do vetor escolhido, então o vetor escolhido será substituído pelo vetor experimental na próxima geração.
• Caso o vetor escolhido possua fitness melhor que o vetor experimental, então o vetor escolhido permanece na população para a próxima geração.

Parâmetros

- Número de indivíduos na população
  – Controla a quantidade de indivíduos na população
  – Estudos empíricos mostram que um bom valor é 10x o número de dimensões
- Fator de escala (β)
  – Controla a amplificação da variação diferencial – Controla a amplificação da variação diferencial
  – Valores pequenos demais comprometem a convergência
  – Valores grandes demais facilitam a exploração porém o algoritmo pode ficar “saltando” do ponto ótimo
- Probabilidade de recombinação (p_r)
  – Influencia diretamente na diversidade, controlando a quantidade de elementos que irão mudar
  – Valores altos aumentam a diversidade e exploração
  – Valores baixos aumentam a robustez da busca

Critérios de Parada

- Número máximo de gerações for atingido.
- Não for observado nenhuma melhoria no melhor indivíduo em um número consecutivo de gerações.
- Não for observado nenhuma alteração na população em um número de gerações.
- Uma solução aceitável for encontrada.
- O delta da função objetivo for aproximadamente zero.

Pseudo código:

```
Inicializa o contador de gerações, t = 0;
Inicializa os parâmetros de controle β e p_r;
Cria e inicializa uma população nx-dimensional;
repete
        para cada indivíduo da população faça
        Calcular o fitness f(x (t)) Calcular o fitness f(x_i(t))
        Cria um vetor experimental pelo operador de mutação
        Cria um filho x_i(t) aplicando o operador de cruzamento
                se f(x'_i(t)) é melhor que f(xi(t)) então
                        Adiciona x'_i(t) para a próxima população;
                caso contrário
                        Adiciona x_i(t) para a próxima população;
                fim
        fim
até a condição de parada ser verdadeira;
Retornar o indivíduo com o melhor fitness como sendo a solução
```

Aplicações:

Problemas de otimização de funções não-lineares, de espaços contínuos, ambientes estáticos multidimensionais (mais de uma dimensão)
– Múltiplos objetivos
– Treinamento de redes MLP
– Problemas reais de otimização:

- Processamento de imagens
- Identificação de objetos

Algoritmos Culturais

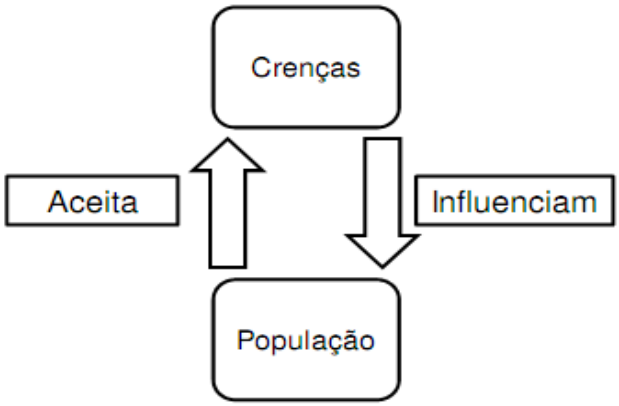
Problema a ser resolvido

Os algoritmos Culturais (AC) não são, em tese, uma técnica nova, trata-se de uma abordagem em que o conhecimento sobre o domínio for utilizado para guiar o processo de busca, melhorando assim a performance dos Algoritmos Evolucionários
O objetivo principal dos AC´s é reduzir o espaço de busca, podando as regiões indesejáveis do espaço de soluções e promovendo as desejáveis.

Composição básica

Um Algoritmo Cultural (AC) é um sistema de dupla herança onde dois espaços de busca são mantidos:

- Espaço da população (componente genética)
    - Representa os indivíduos de uma população
- Espaço de crenças (componente cultural)
    - Modela a informação cultural de uma população
- Ambos os espaços evoluem paralelamente com um influenciando o outro.



Espaço de Crenças

Repositório de conhecimento - repositório onde as crenças dos indivíduos (memes) serão armazenadas; onde um meme é uma unidade de informação transmitida entre unidade de informação transmitida entre indivíduos em uma população.
Os memes em um espaço de crenças são generalizações das experiências individuais em uma população. Essas generalizações são acumuladas e moldadas ao longo das gerações.
Protocolos de Comunicação - Especificam as operações que controlam a influência dos indivíduos no espaço de crenças bem como o papel do espaço de crenças no processo de evolução da população, possibilitando que haja evolução cultural e por fim guiará o processo de busca para regiões mais promissoras.

- Componentes do conhecimento
    - O espaço de crenças possui componentes de conhecimento para representar os padrões comportamentais dos indivíduos da comportamentais dos indivíduos da população.
    - Os tipos e a estrutura de dados usada para representar o conhecimento depende do problema a ser atacado. Para problemas de otimização, o conhecimento pode ser representado de forma vetorial.

Geralmente o espaço de crenças possui duas componentes de conhecimento:

- Situacional - Armazena as melhores soluções encontradas em cada geração, ou seja, o indivíduo que obteve o melhor fitness.

- Normativo - Guia os ajustes mutacionais dos indivíduos no espaço da população. Armazena um conjunto de intervalos para cada dimensão do problema, representado pelo vetor abaixo:

$$X_j(t) = (I_j(t), L_j(t), U_j(t))$$

j = dimensão
t = geração
I = Intervalo
L = fitness do índivíduo no limite inferior na dimensão j
U = fitness no índivíduo no limite superior na dimensão j

Função de Aceitação

Determina quais indivíduos da população serão selecionados para moldar o espaço de crenças. Utiliza métodos estáticos se baseiam apenas no valor de fitness para selecionar um determinado percentual dos indivíduos. Quaisquer métodos de seleção utilizados em Algoritmos Evolucionários podem ser utilizados tais como elitismo, roleta , bem como, métodos dinâmicos e adaptativos.

Ajuste do espaço de crenças

Componente normativa

Para cada indivíduo selecionado:
x_{min},j (t+1) - o menor valor na dimensão j ou o valor da dimensão j do indivíduo o valor da dimensão j do indivíduo cujo fitness seja menor que L_j
x_{max},j (t+1) - o maior valor na dimensão j ou o valor da dimensão j do indivíduo cujo fitness seja menor que U_j

Componente situacional

Para cada indivíduo selecionado:
L_j (t+1) - o valor do fitness do indivíduo com menor valor na dimensão j ou menor valor na dimensão j ou o valor do fitness que seja menor que L_j (t)
U_j(t+1) - o valor do fitness do indivíduo com maior valor na dimensão j ou o valor do fitness que seja menor que U_j (t)

Função de Influência

Usada para determinar o tamanho do passo de mutação e a direção em que essa mudança será realizada.
Componente situacional para determinar a direção da mudança.
Componente normativa para determinar a direção e a situacional para determinar o tamanho do passo.

Pseudo Código:

```
Início
        t=0                                 ;primeira geração
        inicializar população P(t)          ;população inicial aleatória
        Inicializar Espaço de Crença EP(t)
        avaliar população P(t)              ;calcula f(i) para cada indivíduo
        enquanto (não condição_fim) faça
                Comunicação (P(t), EP(t));  ;aceitação das crenças
                Atualização EP(t);          ;uso de operadores culturais
                Comunicação (EP(t), P(t));  ;influência das crenças
                t← t+1                      ;próxima geração
                selecionar P(t) de P(t-1)
                altera P(t)                 ;crossover e mutação
                avaliar P(t)                ;calcula f(i) para cada indivíduo
        fim enquanto
        fim
```

Aplicações:

- Otimização multiobjetivo
- Sistemas híbridos
- Ambientes dinâmicos

Coevolução

Problema a ser resolvido

Resolver problemas ou modelar processos, como realizado por Hillis em resolver problema de ordenação: competitivo e inter-populacional (algoritmos de ordenação e instâncias).

Modelo Computacional

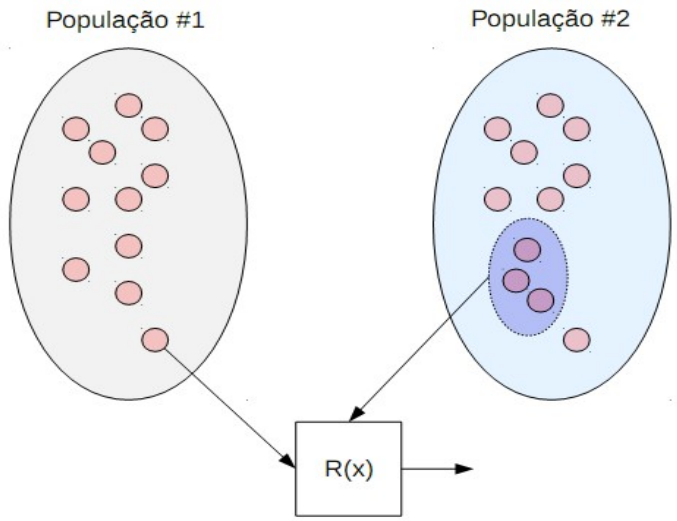
- Múltiplas populações
- Função de Aptidão Relativa
- Influenciada por outras populações + ambiente
- f(P2, E, I) = Quantidade de indivíduos
- superados pelo indivíduo I usando o ambiente E

Tipos de Coevolução

- Competitiva
- Competição Simples
- Amensalismo
- Cooperativa
- Mutualismo
- Comensalismo
- Parasitismo

Coevolução Competitiva (CCE)

- As populações evolvidas são inibidas
- Função de aptidão relativa
- Seleção de amostra para comparação
- Elitismo
- Modelo Genérico para aplicações em outras técnicas como PSO, GA …



Tipos de Seleção

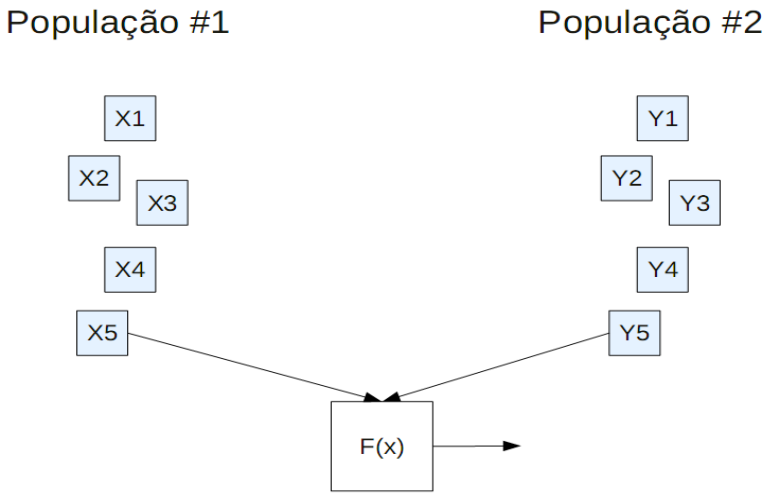
- Todos contra Todos
- Aleatória
- Torneio (Aptidão Relativa)
- Todos contra os melhores (Aptidão)
- Seleção compartilhada

Cálculo da Aptidão Relativa

- Aptidão Simples
- Aptidão Compartilhada
- Aptidão por torneio

Coevolução Cooperativa

- Mutualismo
- Ganho coletivo compartilhado
- Função de aptidão reflete o quão benéfico é o indivíduo para o grupo
- Decomposição da solução em componentes
- Evoluir os componentes isoladamente



Aplicações

- Dificuldade no cálculo do fitness:
- testar todos os casos: custoso ou intratável.
- testar um subconjunto de casos: qual?
- Nenhuma função de fitness conhecida.
- Problemas modularizáveis: dividir-para-conquistar.