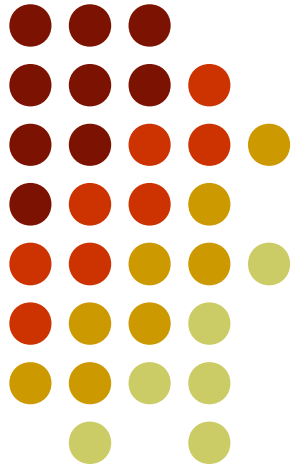
PGCA009 –Inteligência Computacional
Aula 8 Computação Evolutiva

Prof. Angelo Loula

Mestrado em Computação Aplicada (UEFS)



Computação Evolutiva

- Computação Evolutiva é uma das linhas de pesquisa em Inteligência Computacional
- década de 60: Programação Evolutiva por Lawrence J. Fogel, Algoritmos Genéticos por John H Holland (EUA), Estratégias Evolutivas por Ingo Rechenberg and Hans-Paul Schwefel (Alemanha)
  - Desenvolvimento separado e independente por 15 anos só se reunindo na década de 90

Computação Evolutiva

- Computação Evolutiva é um conceito geral de solução de problemas, especialmente problemas de busca e otimização
- Não propõe algoritmos pronto para uso
- Propõe modelos gerais de processos de adaptação inspirados em princípios da biologia evolutiva

Computação Evolutiva

- Evolucionismo
  - Até meados do século XIX: “fixismo”, os seres vivos não sofrem alterações, surgimento de espécies imutáveis.
  - meados do século XVIII: “evolucionismo” ou “transformismo”, o estado natural de todas as coisas (astros siderais, às formas do relevo ou aos seres vivos) é a mudança

Fonte: O Evolucionismo, Felipe Leal e Charbel Niño El-Hani
http://www.cienciaartemagia.ufba.br/darwinnabahia/oevolucionismo.html

Computação Evolutiva

- Evolucionismo
  - A evolução – a modificação das espécies ao longo do tempo – propõe que os seres vivos não são imutáveis: aqueles que são vistos atualmente nem sempre existiram, nem sempre tiveram a mesma forma e nem sempre existirão.
  - Buffon: espécies se transformavam de um modo limitado
  - Lamark: processo evolutivo é uma escalada de complexidade

Fonte: O Evolucionismo, Felipe Leal e Charbel Niño El-Hani
http://www.cienciaartemagia.ufba.br/darwinnabahia/oevolucionismo.html

Computação Evolutiva

- Evolucionismo
  - Darwin: A Origem das Espécies
    - O processo que causa as mudanças evolutivas é a seleção natural
  - Seleção Natural
    - Indivíduos de uma população possuem variações de características
    - Algumas afetam sobrevivência e assim reprodução
    - Características podem ser herdadas
    - Seleção natural aumenta a freqüência populacional de características que melhoram sobrevivência/reprodução

Computação Evolutiva

- Evolucionismo
  - Neodarwinismo (Seleção Natural + Genética)
    - Herança genética de características
    - Variação por mutação e recombinação genéticas
  - Evolução
    - Mudanças geradas por seleção natural
    - Não há um objetivo final, não há planejamento de longo prazo, não vê para onde vai ao final
    - Seleção natural não é aleatória, seleciona características com maior sucesso reprodutivo
    - Variação pode gerar indivíduos melhor ou pior adaptados

Computação Evolutiva

- Terminologia da Evolução e Genética
  - Gene: região do DNA, seqüência de nucleotídeos definindo uma unidade hereditária
  - Cromossomo: unidade compacta de organização de DNA que contém genes, podem ser diplóides ou haplóide
  - Alelo: uma das várias formas de um gene ocupando uma dada posição (locus) em um cromossomo

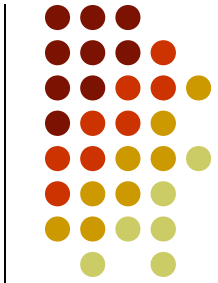
Computação Evolutiva

- Terminologia da Evolução e Genética
  - Genótipo: conjunto completo de genes de um organismo
  - Fenótipo: composição das características observáveis de um organismo
  - Fitness: Sucesso reprodutivo de um indivíduo, sua contribuição para o pool genético da próxima geração, número esperado de filhos sobreviventes
  - Adaptação: estado de estar adaptado ou processo que leva a adaptação, característica que provê vantagem para sobrevivência e reprodução

Computação Evolutiva

- Evolução Natural e Computação Evolutiva
  - Evolução é um processo de otimização.
  - Otimização não implica necessariamente em perfeição
  - Mas evolução pode descobrir soluções muito precisas e adequadas
  - Funcionalidade é otimizada pela seleção e variação
  - Evolução natural se assemelha a um processo de busca e otimização

Computação Evolutiva



- Princípios da Computação Evolutiva
  - Reprodução com herança ‘genética’ de características
  - Variação ‘genética’
  - Seleção favorável aos mais aptos
  - Competição

  - Ao contrário da seleção natural, podemos ‘guiar’ o processo selecionando e variando como desejarmos

Computação Evolutiva

- Terminologia da Computação Evolutiva
  - Representação: codificação dos indivíduos que especificam soluções para o problema, devem ser interpretados para torna-se uma solução candidata
  - Espaço de busca: conjunto de todos indivíduos que podem ser representados
  - População: subconjunto do espaço de busca correspondente aos indivíduos de uma dada geração

Baseado em slides do prof. Fernando von Zuben (Unicamp)

Computação Evolutiva

- Terminologia da Computação Evolutiva
  - Geração: Uma iteração do algoritmo evolutivo
  - Função de adaptação (ou fitness): função que atribui um valor para a qualidade de um indivíduo enquanto solução para o problema
  - Operador de inicialização: procedimento para criar a população inicial
  - Operador de seleção: procedimento de seleção preferencial dos indivíduos melhores adaptados

Baseado em slides do prof. Fernando von Zuben (Unicamp)

Computação Evolutiva

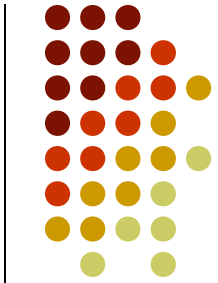
- Terminologia da Computação Evolutiva
  - Operadores genéticos (operadores de busca): procedimentos que simula mutação e recombinação das representações dos indivíduos, gerando variação na população
    - Mutação: altera aleatoriamente partes da representação do indivíduo
    - Recombinação (cross-over): troca partes da representação entre indivíduos

Baseado em slides do prof. Fernando von Zuben (Unicamp)

Computação Evolutiva

- Um Algoritmo Evolutivo (AE)
  1. População inicial de indivíduos, representando por suas características possíveis soluções para um problema
  2. Indivíduos são selecionados segundo sua qualidade para solução do problema
  3. Reprodução dos selecionados com herança e variação de características gerando novos indivíduos
  4. Volta ao passo 2, se critério de parada não for atendido

Computação Evolutiva



- Perguntas
  - Como representar indivíduos?
  - Como criar população inicial?
  - Como avaliar indivíduos?
  - Como selecionar indivíduos?
  - Como reproduzir indivíduos?
  - Como variar indivíduos?
  - Como gerar indivíduos?
  - Como compor nova população?
  - Como saber quando parar?

Computação Evolutiva

- Pressão seletiva: explotação
  - Diminui o escopo de busca
  - Foca esforços em indivíduos mais aptos
- Reprodução: exploração
  - Aumenta o escopo de busca
  - Cria novos indivíduos
- Balanço apropriado
  - ex: baixa pressão seleção + pequena mutação
  - ex: alta pressão seleção + alta mutação

Computação Evolutiva

- Computação Evolutiva:
  uma meta-heurística de busca e otimização
- Não há garantia de solução ótima, ideal
- Não há garantia de convergência
- Para que então? Os problemas difíceis continuam difíceis, e novas abordagens podem ajudar.

Computação Evolutiva

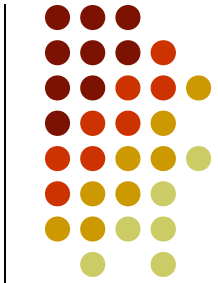
- Aplicações
  - Roteamento: ex. caixeiro viajante
  - Escalonamento: ex. timetabling
  - Empacotamento: ex. problema da mochila
  - Projeto: ex. projeto de filtros
  - Simulação e Identificação: ex. determinação de parâmetros em modelos
  - Controle: ex. adaptação de controladores
  - Classificação: ex. otimização de classificadores

Computação Evolutiva

- Não há algoritmo para resolver todos problemas que seja melhor em geral que qualquer outro

- Mas certos algoritmos evolutivos podem ser melhores em certos problemas

Computação Evolutiva

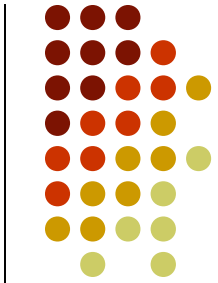


- Desvantagens
  - Não tão bons quanto métodos fortes e específicos para alguns problemas
  - Muitos ajustes e decisões ao projetar um AE
  - Não há garantias
  - Alto custo computacional

Computação Evolutiva

- Vantagens
  - Muito adequados para problemas mais difíceis
  - E para problemas sem métodos de solução conhecidos
  - Aplicação ampla, genérica
  - Mais robustos

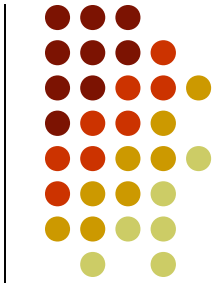
Computação Evolutiva



- Comparação de métodos para solução de problemas
  - Qualidade do resultado
  - Quantidade de recursos computacionais consumido
  - Simplicidade do algoritmo
  - Flexibilidade do algoritmo

Baseado em slides do prof. Fernando von Zuben (Unicamp)

Computação Evolutiva



- Heurísticas Clássicas de Otimização
  - *Hill climbing* é um método de busca (local) que utiliza um procedimento de melhora iterativa (*iterative improvement*). A estratégia é aplicada a um único ponto **x** (solução candidata) no espaço de busca.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

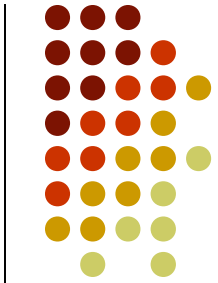
- Heurísticas Clássicas de Otimização

  Hill Climbing Padrão - Seu algoritmo padrão (ou simples) - *standard* (*simple*) *hill climbing* - pode ser descrito como segue:

  - Inicialize (aleatoriamente) o ponto **x** na região factível do problema.
  - A cada iteração um novo ponto **x'** é selecionado aplicando-se uma pequena perturbação no ponto atual, ou seja, selecionando-se um ponto **x'** que esteja na vizinhança de **x**, **x'** $\in$ N(**x**).
  - **Se** este novo ponto apresenta um melhor valor para a função de avaliação, **então** o novo ponto torna-se o ponto atual.
  - O método é terminado quando nenhuma melhora significativa é alcançada, um número fixo de iterações foi efetuado, ou um objetivo foi atingido.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva



- Heurísticas Clássicas de Otimização
  - **Hill Climbing Iterativo -** Como pode ser verificado, o algoritmo de hill climbing padrão realiza uma busca local em torno de seu ponto inicial. No intuito de aliviar esta limitação, o algoritmo também pode ser implementado automaticamente partindo-se de várias condições iniciais distintas e armazenando a melhor solução obtida; cria-se assim, uma espécie de memória para o hill climbing. Este procedimento é denominado de *hill climbing iterativo - iterated hill climbing*:

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização

  Hill Climbing Iterativo

  - Inicialize a melhor solução encontrada até o momento (variável melhor).
  - Para cada condição inicial, faça
    - Inicialize (aleatoriamente) um ponto x na região factível do problema.
    - A cada iteração, um novo ponto **x'** é selecionado aplicando-se uma pequena perturbação no ponto atual, ou seja, selecionando-se um ponto **x'** que esteja na vizinhança de **x**, **x'** $\in$ $N(\mathbf{x})$.
    - **Se** este novo ponto apresenta um melhor valor para a função de avaliação, **então** o novo ponto torna-se o ponto atual.
    - O método é terminado quando nenhuma melhora significativa é alcançada, um número fixo de iterações foi efetuado, ou um objetivo foi atingido.
  - **Se x** é melhor do que *melhor*, **então** *melhor* $\leftarrow$ **x**.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização
  - **Hill Climbing Estocástico (Probabilístico)** - O algoritmo padrão de hill climbing também pode ser modificado de forma a permitir uma escolha probabilística da solução candidata **x'**. Esta probabilidade depende da qualidade relativa entre **x'** e **x**, ou seja, da diferença entre os valores retornados pela função de avaliação quando aplicada a **x'** e a **x**. Este procedimento é denominado de *hill climbing estocástico - stochastic hill climbing*:

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização

  Hill Climbing Estocástico (Probabilístico)

  - Inicialize (aleatoriamente) o ponto **x** na região factível do problema.
  - A cada iteração um novo ponto **x'** é selecionado aplicando-se uma pequena perturbação no ponto atual, ou seja, selecionando-se um ponto **x'** que esteja na vizinhança de **x**, $\mathbf{x}' \in N(\mathbf{x})$.
  - Selecione **x'** com probabilidade $p = (1/(1+\exp[(\text{eval}(\mathbf{x})-\text{eval}(\mathbf{x}'))/T]))$.
  - O método é terminado quando nenhuma melhora significativa é alcançada, um número fixo de iterações foi efetuado, ou um objetivo foi atingido.
  - O parâmetro *T* é mantido fixo durante o processo de busca.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização
  - Simulated Annealing (Recozimento Simulado)
  - O algoritmo de *simulated annealing* proposto por Kirkpatrick et al. (1983) foi inspirado pelo processo de *recozimento* (*annealing*) de sistemas físicos. Uma analogia com o recozimento de sólidos poderia fornecer uma estrutura para o desenvolvimento de um algoritmo genérico de otimização capaz de escapar de ótimos locais.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização
  - O algoritmo de simulated annealing foi desenvolvido baseado em um procedimento utilizado para levar um material a seu estado de *equilíbrio máximo*, ou seja, a um estado de *energia mínima*.
  - Um determinado material é inicialmente aquecido a uma alta temperatura, de forma que ele derreta e seus átomos possam se mover com relativa liberdade.
  - A temperatura desta substância derretida é lentamente reduzida de forma que, a cada temperatura, os átomos possam se mover o suficiente para adotarem uma orientação mais estável.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização
  - Se a substância derretida for resfriada apropriadamente, seus átomos serão capazes de atingir um estado de equilíbrio máximo (energia mínima), produzindo um cristal.
  - Caso contrário, um vidro ou outra substância (quebradiça) semelhante será produzida.
  - Ao processo de aquecimento seguido de um resfriamento lento denominamos de *recozimento* (*annealing*).

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização

  Simulated Annealing
  - Assuma o problema genérico de minimizar uma função:
  - Inicialize (aleatoriamente) um ponto **x** na região factível do problema.
  - A cada iteração um novo ponto **x'** é selecionado aplicando-se uma pequena perturbação ao ponto atual, ou seja, selecionando-se um ponto **x'** que esteja na vizinhança de **x**, $\mathbf{x}' \in N(\mathbf{x})$.
  - Determine a energia de cada uma destas configurações, $E(\mathbf{x})$ e $E(\mathbf{x}')$, e verifique a variação na energia do sistema: $\Delta E = E(\mathbf{x}') - E(\mathbf{x})$.
  - **Se** $\Delta E \leq 0$, **então x'** torna-se o ponto atual ($\mathbf{x} \leftarrow \mathbf{x}'$); senão, a probabilidade de **x'** ser aceito como o ponto atual é dada por um caso particular da distribuição de Boltzmann-Gibbs, ou seja, $P(\Delta E) = \exp(-\Delta E/T)$.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização
  - O método é terminado quando nenhuma melhora significativa é alcançada, um número fixo de iterações foi efetuado, ou a temperatura *T* atingiu seu valor mínimo.
  - O parâmetro *T* é iniciado com um valor elevado e é lentamente reduzido durante o processo de busca.
  - Geralmente implementa-se um decrescimento geométrico para *T*; $T \leftarrow n.T$.
  - A seqüência de temperaturas e o número de rearranjos dos parâmetros até a chegada a um estado de equilíbrio a cada temperatura é denominado de *seqüência* de recozimento (*annealing Schedule*) (Kirkpatrick et al., 1983).

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização
  - Busca Tabu
  - A busca tabu está baseada na premissa de que o processo de resolução de problemas, para ser considerado *inteligente*, precisa incorporar uma *memória adaptativa* e uma *exploração responsável* do espaço de busca. Assim como os algoritmos de *hill climbing* e *simulated annealing*, a busca tabu também é baseada em uma busca em torno de uma vizinhança, onde a principal característica é a utilização de uma *memória* que força o procedimento de busca a explorar novas áreas do espaço de busca de maneira econômica e eficiente.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização
  - Busca Tabu
  - A memória é implementada através de um armazenamento implícito de soluções (movimentações) vistas (efetuadas) no passado. Para isso são usadas estruturas simples, mas eficientes, centradas na criação de uma *lista tabu* de movimentos que foram executados num passado recente da busca e que são considerados *tabu* ou *proibidos* por um determinado número de passos. A utilização de uma lista, ou tabela, tabu ajuda a evitar ciclos e promover diversidade durante a busca.

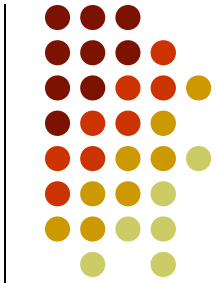
Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização
  - Busca Tabu
  - Na busca tabu, um registro histórico **H** dos estados previamente visitados é mantido, de forma que a perturbação (busca em torno de uma vizinhança) é modificada para levar em consideração o histórico de atualização da busca. A maneira mais simples de definir **H** é através de uma proibição de revisitar alguns pontos na vizinhança do ponto atual $N(\mathbf{x})$. Estes pontos são denominados *tabu*, e **H** tem o efeito de restringir a busca dentro de uma vizinhança.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva



- Heurísticas Clássicas de Otimização
  - Busca Tabu
  - As várias formas de se identificar características do histórico do processo de busca são descritas pelos diferentes tipos de memória:
    - Memória recente (recency);
    - Memória baseada em freqüência (frequency);
    - Memória baseada em qualidade (quality); e
    - Memória baseada em influência (influence).

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização
  - A busca tabu gera um conjunto X de soluções candidatas na vizinhança do ponto **x**. Dentre estas soluções candidatas, uma nova solução deve ser escolhida. Em geral, o mecanismo de seleção é do tipo elitista, ou seja, a melhor solução, dada uma função de avaliação ou fitness, é escolhida. Entretanto, dois conceitos importantes em busca tabu são: intensificação e diversificação.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon

Computação Evolutiva

- Heurísticas Clássicas de Otimização
  - As *estratégias de intensificação* se baseiam na modificação do mecanismo de seleção de forma a encorajar combinações de movimento e características de soluções historicamente boas. Elas também podem forçar um retorno a regiões atrativas do espaço de busca para, forçadamente, explorá-las melhor.
  - As *estratégias de diversificação*, por outro lado, encorajam o processo de busca a examinar regiões inexploradas do espaço e gerar soluções que diferem daquelas vistas anteriormente.

Fonte: LVCoN - Laboratório Virtual de Computação Natural
http://lvcon.tuilux.com.br/lvcon