Busca

Inteligência Artificial – Prof. Flávio Varejão
Departamento de Informática
Universidade Federal do Espírito Santo

Agenda

- Motivação
- Heurísticas
- MetaHeurísticas
  - Baseadas em Soluções Parciais
  - Baseadas em Soluções Completas
    - Busca Local
    - Busca Populacional
- Perspectivas e Desafios
- Conclusões

Motivação

Motivação

- Problemas NP
  - Bem definidos
    - Descritor de espaço de estados e operadores conhecidos
  - Função objetivo
    - Não é suave
    - Possui ótimos locais
  - Algoritmo conhecido resolve por enumeração
    - Intratável computacionalmente
- Soluções aproximadas são suficientes

Heurísticas

Heurísticas

- Procedimentos de resolução
  - Específicos para o problema a ser resolvido
  - Acham solução, mas não garantem que seja ótima
  - Normalmente baseados na experiência

Heurísticas

- Exemplo
  - Jogo da Velha
    - Se inicia então joga no canto superior esquerdo
    - Se adversário joga no meio então joga no canto inferior direito
      - Se adversário joga em um canto então jogue no outro canto – na sua próxima vez vença o jogo
      - Senão jogue as próximas rodadas de modo a impedir que o adversário vença o jogo e, se ele bobear, vença o jogo
    - Senão joga em canto que permita ganhar na sua próxima rodada e que não permita ao adversário alinhar a segunda jogada com a primeira
      - Se o adversário impedir sua vitória na próxima jogada então jogue no canto em que a linha e coluna não possui jogada do adversário – na sua próxima vez, vença o jogo

Heurísticas

- Problema
  - Desenvolver heurísticas específicas para problema a ser resolvido pode ser difícil e/ou trabalhoso

Meta-heurísticas

Meta-heurísticas

- Procedimentos heurísticos gerais aplicáveis a um variedade de problemas
  - Tipicamente inclui características para fazer com que o procedimento de busca fuja de ótimos locais
- Definem infraestrutura para criação de procedimentos heurísticos
- Duas categorias principais
  - Baseadas em Soluções Parciais
  - Baseadas em Soluções Completas

Baseadas em Soluções Parciais

Baseadas em Soluções Parciais

- Construção de Solução
  - Busca realizada na medida que a solução é construída
- Procedimentos
  - Hill Climbing
  - Beam Search
  - Branch and Bound

Hill Climbing

- Estratégia gulosa
  - Decisões baseadas em informação local
  - Uso de função estimativa heurística para escolher o próximo elemento da solução
  - Não há retrocesso na busca
  - Exemplo
    - Caixeiro Viajante
      - A partir da primeira cidade, vá para a cidade mais próxima ainda não visitada até que todas as cidades tenham sido visitadas. Então, retorne à primeira cidade.

Hill Climbing

- Algoritmo

procedimento *HillClimbing ()*

sm ← Ø

enquanto *sm não está completa* ***faça***

Avaliar elementos ainda não selecionados

r ← elemento melhor avaliado

sm ← sm U r

fim-enq

retorna *sm*

fim-proc

Beam Search

- Busca em amplitude
  - Uso de função objetivo para avaliação dos estados
  - Limita total de estados mantidos as m melhores expansões dos estados atuais

Beam Search

- Algoritmo

procedimento *BeamSearch (m)*

avaliar todos os possíveis estados iniciais e manter os m melhores na fila f

enquanto *solução não está completa* ***faça***

Expandir todos os estados na fila f e avaliá-los

Atualizar f com m melhores estados

fim-enq

retorna *o estado de melhor avaliação*

fim-proc

Branch and Bound

- Requer (estimativa de) valor de função objetivo de uma solução (trivial)
- Dividido em duas etapas
  - Expansão de estados correntes
  - Poda de ramos que apresentam custo superior ao da solução atual

Branch and Bound

- Função de avaliação leva em conta
  - Função objetivo do estado atual
  - Função heurística
    - Estimativa do custo para se chegar a solução
    - Otimista
      - Avalia sempre igual ou inferior a custo real
      - Garante achar o ótimo
    - Não otimista
      - Pode perder ótimo

Branch and Bound

- Algoritmo

procedimento *BranchBound ()*

obter custo de uma solução trivial

incluir na fila ordenada f todos os possíveis estados de partida com avaliação inferior ao custo da solução trivial

enquanto *f não está vazia* ***faça***

Remover o primeiro s da fila f

se *s for solução* ***então retorna*** *s*

Avaliar os próximos estados de s

Incluí-los na fila ordenada se custo inferior ao da solução trivial

fim-enq

fim-proc

Baseadas em Soluções Completas

Baseadas em Soluções Completas

- Necessário ter soluções completas para o problema para aplicar a heurística
  - Vetor descritor de estado totalmente preenchido
- A partir das soluções completas são feitas modificações com o intuito de encontrar uma melhor (boa) solução
- Duas subcategorias
  - Busca Local
  - Busca Populacional

Busca Local

Busca Local

- Busca na vizinhança de uma ou mais soluções
- Fácil de compreender/aprender
- Obtém soluções rapidamente
- Pouco tempo de implementação
- Necessário pouco ou nenhum conhecimento do problema

Busca Local

- Algoritmo Geral
  1. Gerar solução inicial
  2. Modificar levemente a solução
  3. Avaliar nova solução
  4. Repetir passos 2 e 3 até que nenhuma melhoria significativa é encontrada

Busca Local

- Procedimentos
  - Simple descent
  - Deepest descent
  - Multistart descent
  - Tabu search
  - Simulated annealing
  - GRASP

Busca Local

- Necessárias funções para
  - Gerar solução inicial (*s*)
  - Determinar vizinhança (*V(s)*)
  - Função objetivo (*f*)
- Minimização de função objetivo

Simple Descent

- Algoritmo

procedimento *Simple_Descent (s)*

repetir

Escolher s' ∈ V(s)

se *f(s') < f(s)* ***então***

s ← s'

fim-se

até *f(s') ≥ f(s), ∀s' ∈ V(s)*

fim-proc

Deepest Descent

- Algoritmo

procedimento *Deepest_Descent (s)*

repetir

Escolher $s' \in V(s)$ / $f(s') \leq f(s'')$ $\forall s'' \in V(s)$

se $f(s') < f(s)$ ***então***

$s \leftarrow s'$

fim-se

até $f(s') \geq f(s)$, $\forall s' \in V(s)$

fim-proc

Multistart Descent

- Algoritmo

```
procedimento Multistart_Descent
    iter = 1
    f(sm) = +∞
    repetir
        Escolher s' aleatoriamente
        s ← Simple_Descent(s') (ou Deepest_Descent(s'))
        se f(s) < f(sm) então
                sm ← s
                f(sm) ← f(s)
        fim-se
        iter = iter + 1
    até iter = iterMax
fim-proc
```

Tabu Search

- Movimento para pior solução é permitido
- Previne volta a soluções já visitadas através da lista tabu
  - Evita ciclo na busca
- Lembra a melhor solução encontrada até o momento
- Critérios de parada
  - Tempo de cpu, número de iterações, número de iterações sem melhorias

Tabu Search

- Algoritmo

procedimento *Tabu_Search (s)*

sm ← s

repetir

Escolher s'∈ V(s) / f(s') ≤ f(s'') ∀ s'' ∈ V(s)

se *f(s') < f(sm)* ***então***

sm ← s'

fim-se

se *s' não está na lista tabu* ***então***

s ← s'

atualiza lista tabu

fim-se

até *satisfazer critérios de parada*

fim-proc

Simulated Annealing

- Analogia com processo físico de resfriamento de sólido superaquecido
  - Na medida que esfriam, entram em estados de menor energia
  - Ao longo do processo pode entrar em estados com maior energia
    - Ocorre de modo aleatório
    - Menos frequente na medida que a temperatura se reduz

Simulated Annealing

- Busca não determinística
- Soluções melhores são sempre aceitas
- Movimento para pior solução é permitido
  - Quão pior a solução, menor a chance de ser aceita
  - Quão menor a temperatura, menor a chance de ser aceita

Simulated Annealing

- Algoritmo

procedimento *Simulated_Annealing (s)*
sm ← s
Escolher t (onde t é uma temperatura inicial)
Gerar α entre [0,1] (α é fator de redução de t)
repetir
para *i de 1 até numIter*
Escolher aleatoriamente um vizinho s' de s
se *f(s') < f(s) então*
s ← s'
se f(s') < f(sm) então
sm ← s
fim-se
senão
s ← s' com probabilidade $e^{(- (f(s') - f(s))/t)}$
fim-se
fim-para
*t ← α * t*
até *que o critério de parada seja satisfeito*
fim-proc

Análise de Sensitividade de Hiperparâmetros

- Hiperparâmetros de Simulated Annealing
  - t_0 e α
  - Várias instâncias do problema geradas aleatoriamente (30)
  - Monovariada
    - t_0
    - α
  - Multivariada
    - Busca em grade: pares (t_0 , α)

GRASP

- Greedy Randomized Adaptive Search
- Consiste em um método iterativo probabilístico, onde a cada iteração é obtida uma solução do problema em estudo
- Cada iteração GRASP é composta de duas fases: a primeira, a construtiva, que determina a solução que será submetida à busca local e a segunda fase, cujo objetivo é obter alguma melhoria na solução corrente
- Requer função de avaliação gulosa

GRASP

- Algoritmo

procedimento *Grasp ()*
para *i de 1 até numIter*
s ← GreedyRandomConstruct(sem)
s ← LocalSearch(s)
se *f(s) < f(sm) então*
sm ← s
fim-se
fim-para
retorna *sm*
fim-proc

GRASP

- Algoritmo

procedimento *GreedyRandomConstruct (sem)*

s ← Ø

enquanto *s não está completa* ***faça***

Avaliar os elementos ainda não selecionados

Escolher os m melhores elementos

r ← selecionar aleatoriamente um dos m

sm ← sm U r

fim-enq

retorna *sm*

fim-proc

Busca Populacional

Busca Populacional

- Baseada numa população de soluções
- Soluções são combinadas para gerar nova população

Busca Populacional

- Computação Evolutiva
  - Algoritmos Genéticos
  - Algoritmos Meméticos
- Inteligência Coletiva
  - Colônia de Formigas
  - PSO

Computação Evolutiva

- Inspirada na teoria da evolução natural e na genética
- Trata de sistemas para a resolução de problemas que utilizam modelos computacionais baseados na teoria da evolução natural
- Algoritmos genéticos é o representante mais importante e popular

Algoritmos Genéticos

Algoritmos Genéticos

- São programas evolutivos baseados na teoria de seleção natural e hereditariedade. Favorecem os candidatos mais promissores para a solução de um dado problema
- Desenvolvido por John Holland (1975) e seus alunos
- Popularizado por David Goldberg (1989)

Algoritmos Genéticos

- Características
  - Podem trabalhar com uma codificação do conjunto de parâmetros ou com os próprios parâmetros
  - Trabalham com uma população e não com um único ponto
  - Utilizam informações de custo ou recompensa
  - Utilizam regras de transição estocásticas e não determinísticas

Algoritmos Genéticos

- Características
  - São fáceis de implementar em computadores
  - Adaptam-se bem a computadores paralelos
  - São facilmente combinados com outras técnicas
  - Funcionam com parâmetros contínuos ou discretos

Algoritmos Genéticos

- Classe de procedimentos com um conjunto de passos distintos e bem especificados, no qual cada um destes passos possui muitas possíveis variações
- Não são limitados por suposições sobre o espaço de busca, relativas a continuidade, existência de derivadas, etc.

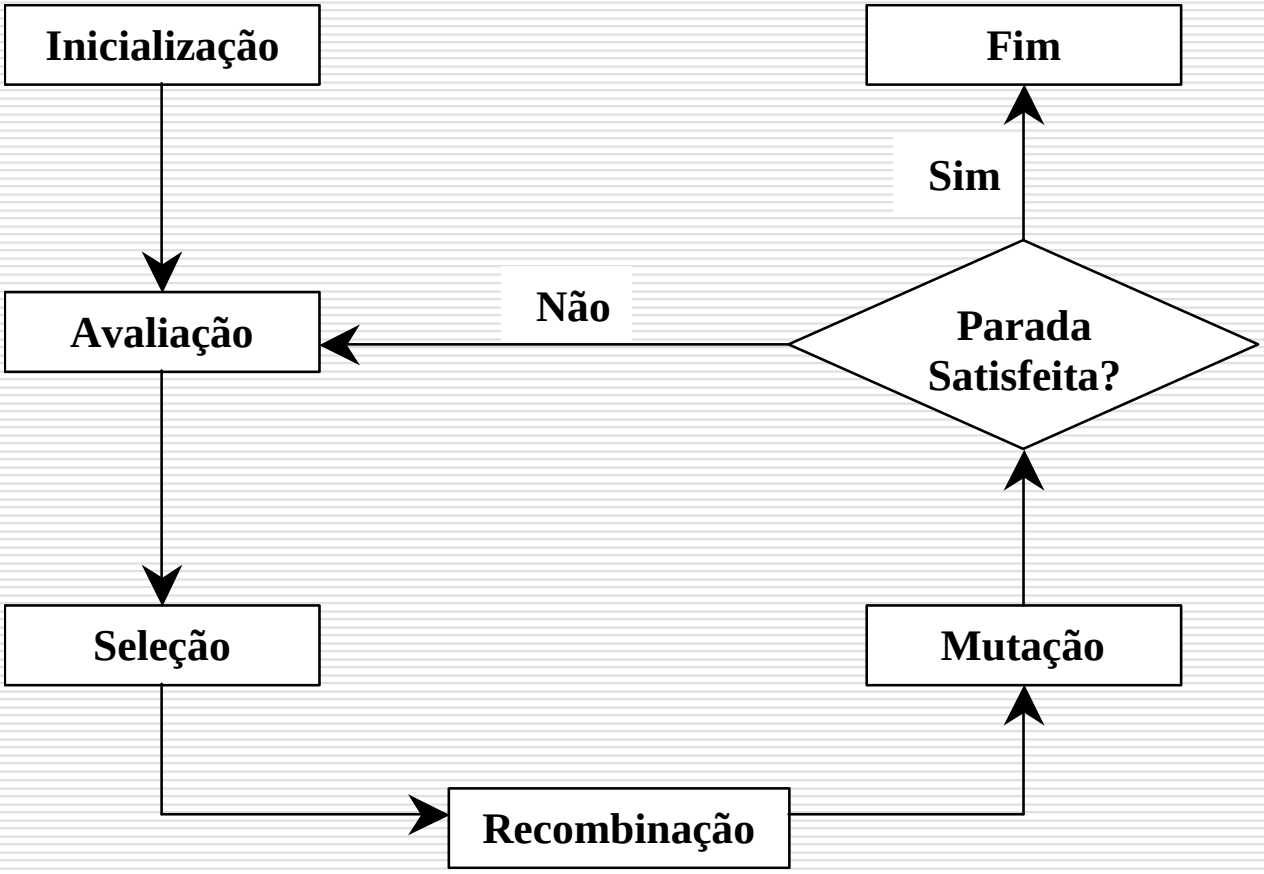
Representação

- Os parâmetros do problema são representados como genes em um cromossomo
- Cada gene pode assumir valores específicos, sendo cada um destes valores chamados de alelo do gene
- Um cromossomo representa um indivíduo, sendo composto por uma configuração de alelos
- A posição de um gene num cromossomo corresponde a um locus gênico

Representação

Evolução Natural	**Problema Computacional**
Indivíduo	Solução de um problema
População	Conjunto de soluções
Cromossomo	Representação de uma solução
Gene	Parte da representação de uma solução
Crossover	Operador de Busca
Mutação	Operador de Busca
Seleção natural	Reutilização de boas aproximações

Estrutura Básica



Genético

- Algoritmo

procedimento *Genético()*

gerar população inicial e avaliar

enquanto *não satisfaz critério de parada* ***faça***

selecionar os mais aptos

gerar novos através de crossover e mutação

repor inviáveis

avaliar nova população

fim-enq

retorna *melhor*

fim-proc

Seleção

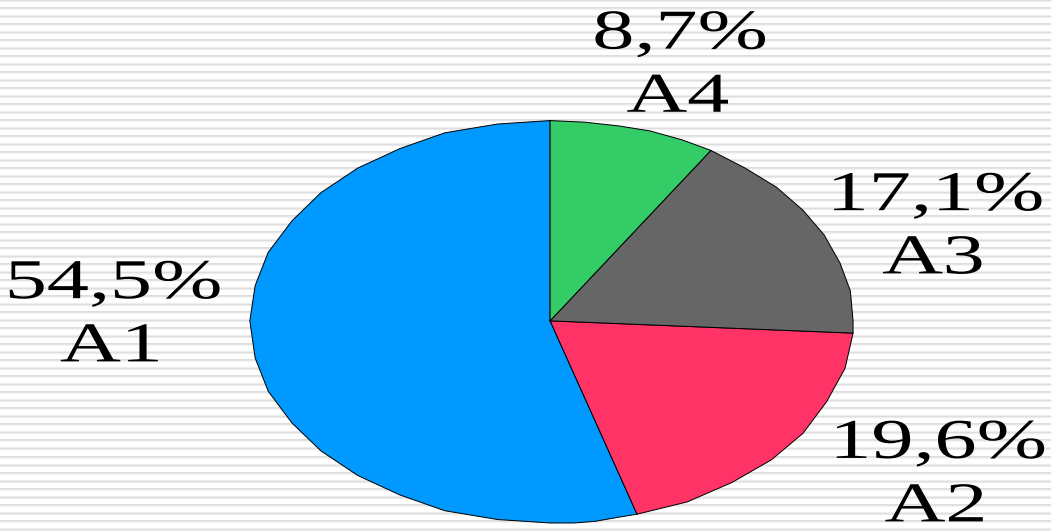
- O princípio básico do funcionamento dos AGs é que um critério de seleção vai fazer com que, depois de muitas gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos

Seleção

- Uso de função objetivo como avaliação de aptidão
  - A aptidão pode ser vista como uma nota que mede o quão boa é a solução codificada por um indivíduo
  - Normalmente baseada no valor da função-objetivo, específica para cada problema
- Métodos de Seleção
  - Roleta
  - Torneio
  - Amostragem Universal Estocástica

Método da Roleta

- Aptidão usada para definir fatia
- Valor aleatório para selecionar cromossomo
- Processo repetido até gerar os n indivíduos necessários



Método do Torneio

- Escolha aleatória de m indivíduos
- Uso de função de aptidão para escolher o melhor
- Processo repetido até gerar os n indivíduos necessários

Método da Amostragem

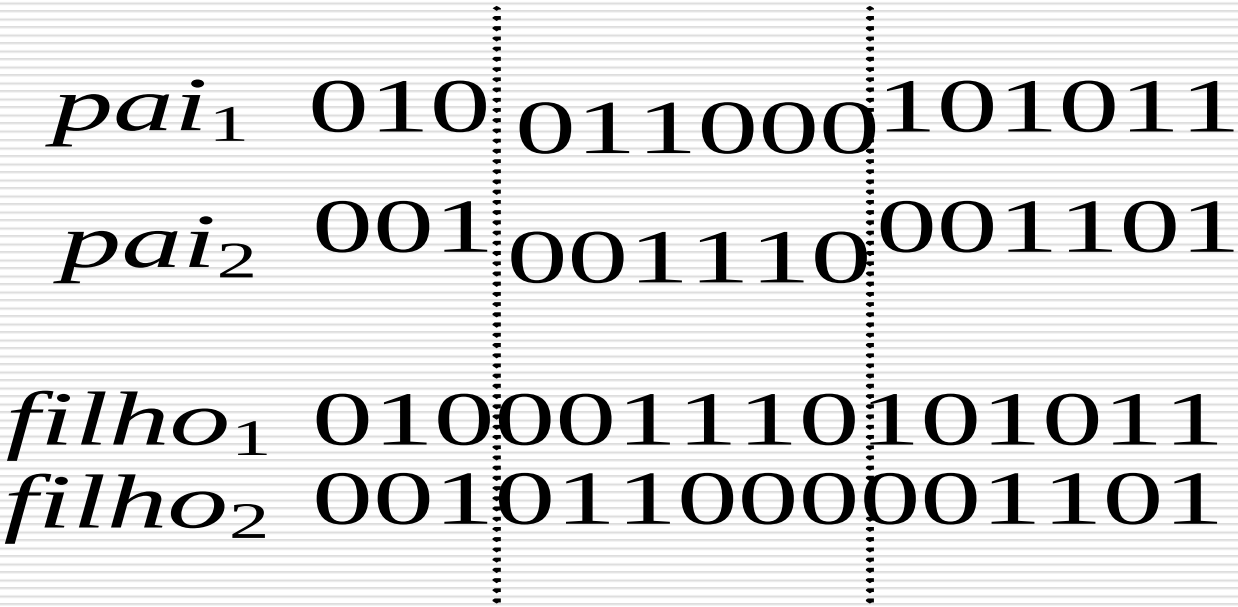
- Método da roleta com n agulhas igualmente espaçadas
- Roleta é girada uma única vez

Operadores Genéticos

- Cruzamento
  - Cruzamento de pais para gerar dois filhos
  - Taxa de crossover
  - Tipos
    - Ponto Único
    - Dois Pontos
    - Multiponto
    - Uniforme
- Mutação

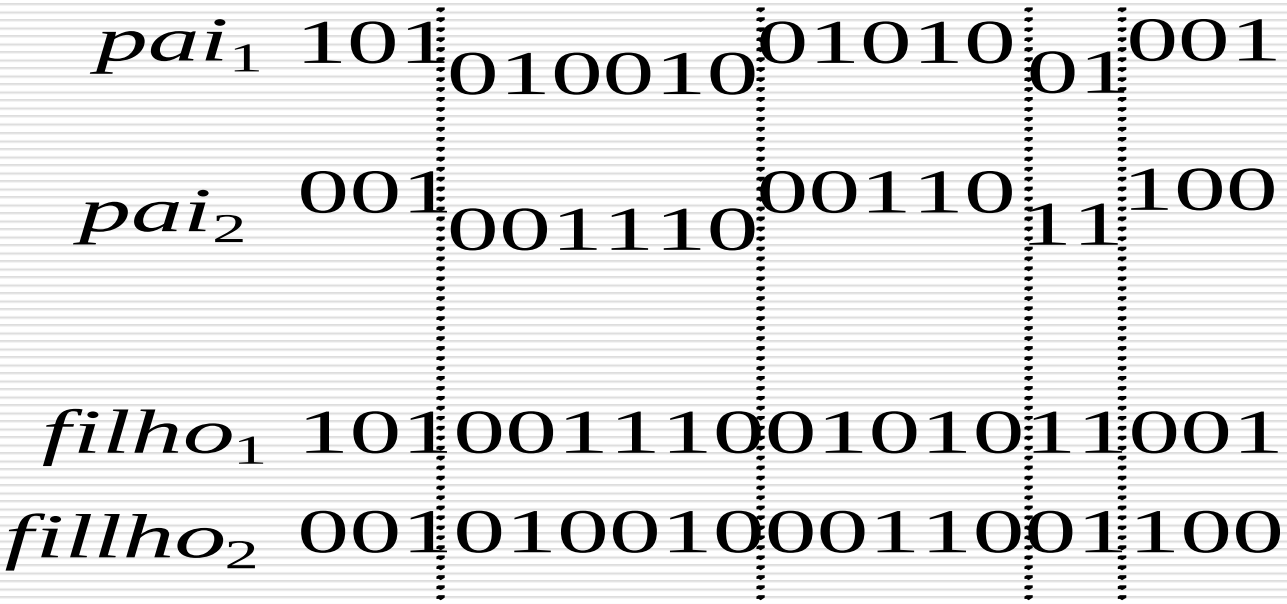
Cruzamento

- Dois Pontos



Cruzamento

- Multipontos



Parâmetros Genéticos

- Tamanho da população
- Taxa de cruzamento
- Taxa de mutação
- Intervalo de geração
  - Percentual de renovação da população

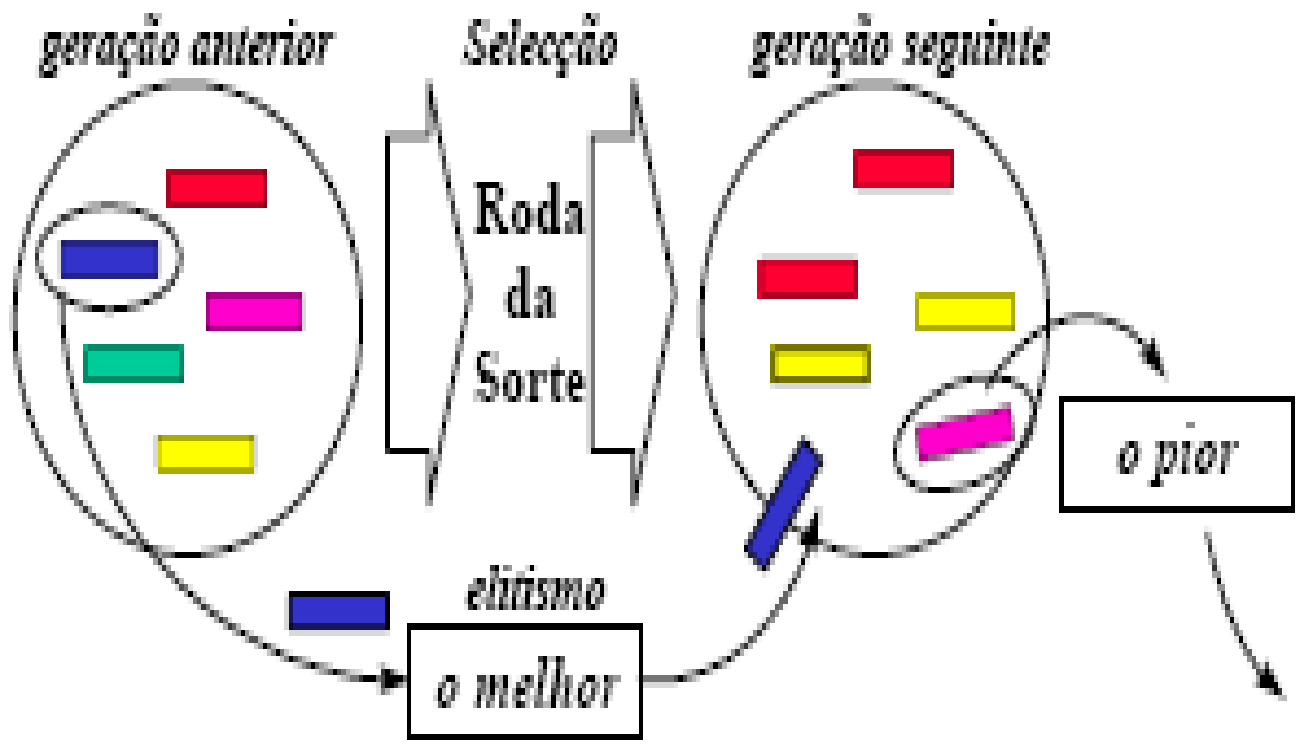
Parâmetros Genéticos

- Critério de parada
  - Número de gerações
  - Convergência da função de aptidão na população
  - Não melhoria da aptidão do melhor indivíduo após um número de gerações

Elitismo

- Um elemento que tenha maior aptidão que outro tem também maior probabilidade de ser selecionado
- Nada impede que seja selecionado o pior, perdendo-se assim talvez o melhor elemento da população, que poderia levar a uma convergência mais rápida
- Para tentar minimizar este possível problema, *elitismo* pode ser adicionado à seleção
- Percentual de indivíduos com melhor aptidão são mantidos na nova geração

Elitismo


geração anterior
Selecção
geração seguinte
Roda
da
Sorte
o pior
elitismo
o melhor

Variações sobre AGs

- Diversidade Populacional
  - Grau de diferença entre os cromossomos
    - Uso de medidas de diferença
  - Quanto maior a diversidade, maior o espalhamento das soluções avaliadas
  - Ideal
    - Grande no início, menor no final
  - Como aumentar diversidade?
    - Aumentar taxa de mutação
    - Aumentar população
  - Pode ser usado para controlar convergência prematura

Variações sobre AGs

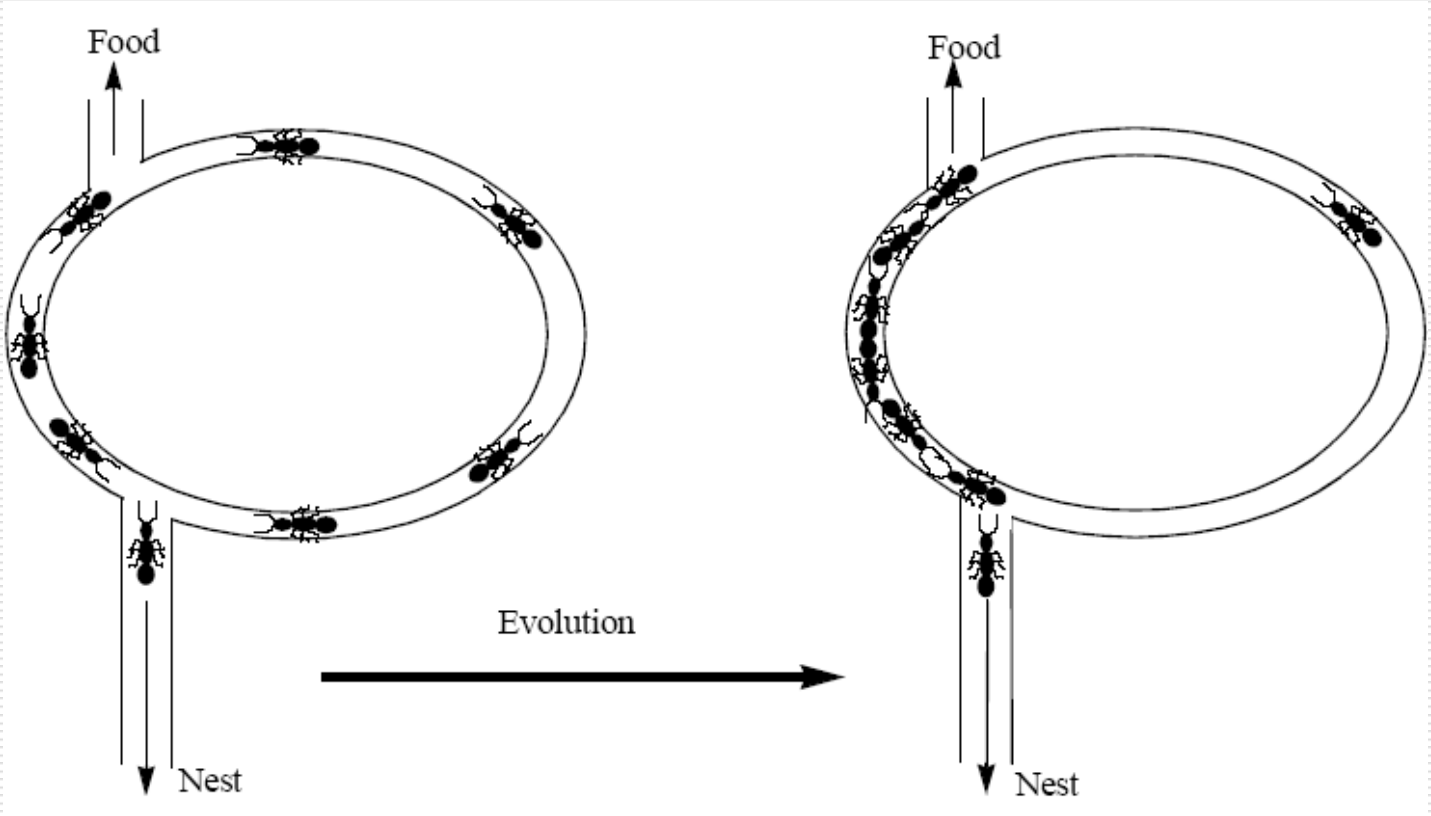
- Outros Operadores
  - Operador de Dominação
    - Alelo dominante e recessivo
  - Operador de Inversão
    - Segmento de cromossoma é invertido
  - Operador de Segregação
    - Combinação aleatória gene a gene
  - Operador de Translocação
    - Cruzamento de múltiplos cromossomos

Inteligência Coletiva

Colônia de Formigas

Colônia de Formigas

- Experimento com formigas reais



Colônia de Formigas

- Experimento com formigas reais
  - Inicialmente metade vai pelo menor e metade vai pelo maior
  - Formigas que usam o menor caminho vão e voltam mais rapidamente
    - Ocorre maior depósito de feromônios no menor caminho
    - Evaporação de feromônios é maior
  - Ao final formigas usam sempre o menor caminho

Colônia de Formigas

- Proposto por Dorigo e Gambardella em 1997
- Aplicado principalmente para Caixeiro Viajante
  - Mas pode ser adaptado para outros problemas

Colônia de Formigas

- Algoritmo

procedimento *AntColony (tam)*

enquanto *condições de parada não satisfeitas* ***faça***

construir soluções

aplicar opcionalmente busca local

atualizar informação de feromônio

fim-enq

fim-proc

Colônia de Formigas

- Algoritmo

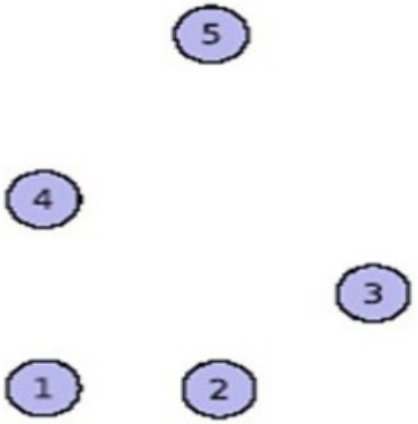
procedimento *construir soluções*
    para *cada formiga na colônia* ***faça***
        sorteie aleatoriamente estado inicial da formiga
        enquanto *não completa solução* ***faça***
            avaliar próximos estados baseados na função objetivo e no depósito de feromônio
            escolher aleatoriamente próximo estado com maior probabilidade para estados de melhor avaliação
        fim-enq
    fim-para
fim-proc

Colônia de Formigas

ACO aplicado ao PCV

Matriz de distâncias

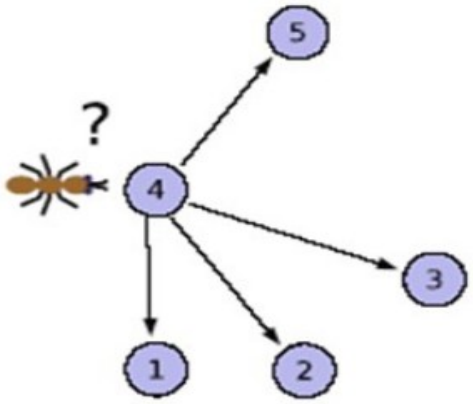
dij	1	2	3	4	5
1	0,0	1,0	2,2	2,0	4,1
2	1,0	0,0	1,4	2,2	4,0
3	2,2	1,4	0,0	2,2	3,2
4	2,0	2,2	2,2	0,0	2,2
5	4,1	4,0	3,2	2,2	0,0



Cidades do PCV

Colônia de Formigas

- Começando de uma cidade *i*, a formiga move-se escolhendo probabilisticamente a cidade vizinha *j* (entre os vizinhos factíveis)



Colônia de Formigas

Probabilidade de Transição

- A probabilidade da formiga k que está na cidade i de escolher a cidade j é dada pela regra

$$p_{ij}^{k} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{l \in N_j^k} \tau_{il}^{\alpha} \eta_{jl}^{\beta}}, \quad quando \quad j \in N_i^k$$

onde:

- τ_{ij} é o feromônio associado à aresta (i, j)
- α e β são parâmetros para determinar a influência do feromônio e da informação heurística
- N_j^k é a vizinhança factível da formiga k (isto é, o conjunto de cidades ainda não visitadas pela formiga k).

Colônia de Formigas

Informação heurística do PCV

- Associada a aresta (i, j) existe uma valor heurístico η_{ij} dado por

$$\eta_{ij} = 1/d_{ij}$$

que representa a atratividade da formiga visitar a cidade i depois de visitar a cidade j

- O valor η_{ij} é inversamente proporcinal a distância d_{ij} entre as cidades i e j
- A partir de uma cidade i, a escolha da cidade candidata j é feita de acordo com a probabilidade de transição, com idéia similar à escolha por roleta de algoritmos genéticos

Colônia de Formigas

Atualização do Feromônio

No feromônio τ_{ij} associado a aresta (i, j) ocorrem dois eventos:

1. Evaporação
   - ❑ evita que o feromônio acumulado cresça indefinidamente
   - ❑ permite esquecer pobres decisões do passado de busca
   - ❑ permite soluções diferentes
2. Depósito de feromônio de todas as formigas que passaram sobre (i, j)

Colônia de Formigas

Depois que todas as formigas contruíram suas viagens, o feromônio é atualizado

- $\Delta\tau_{ij}^{k}$ é a quantidade de feromônio que a formiga k deposita sobre a aresta (i, j):

$$\begin{cases} \Delta\tau_{ij}^{k} = Q/L_k \text{ quando a aresta } (i,j) \text{ pertence } S_k \\ \Delta\tau_{ij}^{k} = 0 \text{ em caso contrário} \end{cases}$$

onde Q é uma constante

evaporação

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$$

depósito

onde $0 < \rho \leq 1$ é a taxa de evaporação de feromônio

Colônia de Formigas

- Critérios de parada
  - Número máximo de iterações
  - Estagnação ou convergência
    - Todas formigas percorrem o mesmo percurso
    - Tende a indicar ótimo local

Enxame de Partículas

Enxame de Partículas

- PSO (Particle Swarm Optimization)
  - Propostos inicialmente em 1995 por James Kennedy e Russel Eberhart
    - Otimização Contínua
      - Mas pode ser adaptado para Otimização Combinatória
  - Inspirada
    - Observação do comportamento de enxame de peixes ou pássaros
    - Computação Evolucoonária

Enxame de Partículas

- Mantém múltiplas soluções promissoras em paralelo
  - Cada solução é uma partícula no espaço de busca
- Durante iterações, cada solução é avaliada pela função objetivo para obter sua adaptação
  - Partículas se movem no espaço de busca procurando valor ótimo da função objetivo

Enxame de Partículas

- Cada partícula mantém
  - Posição no espaço de busca
    - Solução
    - Aptidão (valor da função objetivo)
  - Velocidade
  - Melhor posição encontrada
- Enxame
  - Melhor posição encontrada

Enxame de Partículas

- Algoritmo

enquanto *não atender critério de parada* ***faça***

avaliar aptidão de cada partícula

atualizar melhor individual e melhor global

atualizar velocidade e posição de cada partícula

fim-enq

Enxame de Partículas

- Atualização da velocidade de cada partícula

$$v_i(t+1) = wv_i(t)+c_1r_1[\hat{x}_i(t)-x_i(t)]+c_2r_2[g(t)-x_i(t)]$$

  - w é coeficiente de inércia
    - Mantém partícula se movendo na mesma direção
    - Geralmente entre 0.8 e 1.2

    Valores baixos aceleram convergência enquanto valores altos estimulam exploração do espaço de busca

Enxame de Partículas

- c1 e c2 são coeficientes de aceleração

  0 <= (c1, c2) <= 2

  - Normalmente próximos de 2

  - c1 é o componente cognitivo
    - Atua como memória da partícula incentivando o retorno da partícula para a sua melhor posição individual $\hat{x}_i(t)$
    - Limita o tamanho do passo da partícula na direção da melhor posição individual

Enxame de Partículas

- c1 e c2 são coeficientes de aceleração

  0 <= (c1, c2) <= 2

  - Normalmente próximos de 2

- c2 é o componente social
  - Atua como memória do enxame incentivando o retorno da partícula para a sua melhor localização global $g(t)$
  - Limita o tamanho do passo da partícula na direção da melhor posição global

Enxame de Partículas

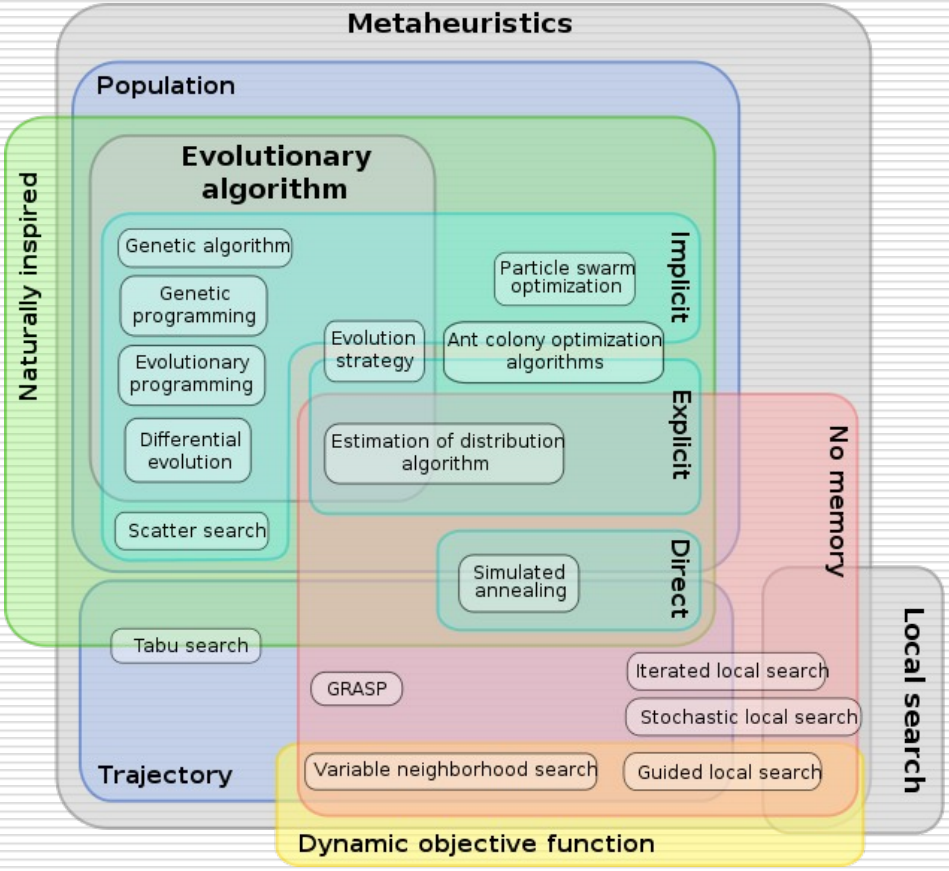
- r1 e r2 são valores aleatórios

  0 <= (r1, r2) <= 1

  – Gerados a cada atualização de velocidade
- Atualização da posição de uma partícula

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

Distribuição de Metaheurísticas



Distribuição de Metaheurísticas

- Trajetória
  - Busca em vizinhança
- Sem memória
  - Não recua para ponto melhor avaliado
- Função objetivo dinâmica
  - Varia a função objetivo de acordo com o grau de exploração ou intensificação da busca
- Interação Populacional
  - Implícita: interação não é influenciada diretamente pela função de aptidão
  - Explícita: interação direcionada por função de aptidão

Perspectivas e Desafios

- Como comparar diferentes algoritmos?
  - Experimentação
    - Univariada
      - Só Custo
      - Só Tempo de Execução
    - Multivariada
      - Custo e Tempo de Execução
    - Tempo fixo

Perspectivas e Desafios

- Como comparar diferentes algoritmos?
  - Procedimento Usual
    - Várias instâncias de um problema
    - Aplicar repetidamente os algoritmos
    - Comparação
      - Baseada em média e desvio padrão dos resultados
      - Baseada em ranking

Perspectivas e Desafios

- Como comparar diferentes algoritmos?
  - Problemas comuns com procedimento usual
    - Amostragem
      - Exemplos reais ou artificialmente gerados
        - Não representa a distribuição real
        - Em quantidade insuficiente
    - Ajuste de parâmetros do algoritmo
      - Não separação de conjuntos de validação e teste
      - Resultados enviesados
        - Ajuste de parâmetros até obter melhores resultados

Perspectivas e Desafios

- Como comparar diferentes algoritmos?
  - Necessários Testes Estatísticos
    - Contudo,
      - Grande variedade de testes
        - Vários requisitos para aplicação
      - Não se conhece a distribuição estatística das instâncias do problema
        - Problema pode ser parametrizável
      - Necessidade de grande esforço computacional para realização dos experimentos

Perspectivas e Desafios

- Como comparar diferentes metaheurísticas?
  - Vários problemas de vários domínios
  - Necessários mais testes estatísticos
    - Necessidade de maior esforço computacional para realização dos experimentos

Conclusões

- Meta-heurísticas podem ser aplicadas a vários problemas bem definidos complexos
- Contudo
  - Existe uma grande variedade de heurísticas
    - Escolha de qual aplicar é difícil
  - Devem ser ajustadas para cada aplicação
    - Ajuste pode ser a etapa mais difícil

Conclusões

- Contudo
  - Cada uma requer a especificação de parâmetros que podem afetar o seu desempenho
    - Obter bons valores para os parâmetros requer experimentação
    - Métodos estatísticos de experimentação são recomendados mas demandam trabalho árduo
  - Solução ótima não é garantida