

CMC-15 Inteligência Artificial – Primeiro Laboratório -

Professor: Paulo Marcelo Tasinaffo.

Data de Divulgação: primeira semana do segundo bimestre.

Data de Entrega: *até sexta-feira da oitava semana do segundo bimestre*. O atraso na entrega da lista acarretará no desconto de 20% na nota da mesma. Depois da primeira semana de exames a lista de laboratório não será mais aceita pelo professor.

Regulamentos:

1. Esta Lista de Laboratório (LL1) pode ser resolvida em grupos de dois ou três alunos em cada grupo.
2. A média aritmética das notas das Listas Teóricas (LT1) e (LT2) com a ***nota da Lista de Laboratório*** (LL1) comporá a nota final do segundo bimestre da disciplina CMC-15.
3. O ***segundo laboratório*** (LL2) comporá a nota de Exame Final da parte do Prof. Tasinaffo.
4. Todas as listas e laboratórios deverão ser **entregues até a sexta-feira da última semana de aula do Segundo Bimestre (29/11/2024, Sexta-Feira)**. Deverão ser entregues quatro arquivos PDF (por grupo) e separados; quais sejam, LT1.pdf, LT2.pdf, LL1.pdf e LL2.pdf. Estes quatro arquivos deverão ser enviados e entregues via ambiente Classroom ou para o e-mail do Prof. Tasinaffo.
5. O e-mail do Professor Tasinaffo, para eventuais dúvidas, é tasinaffo@ita.br.
6. Serão disponibilizados três temas de laboratório. Entretanto, cada grupo poderá escolher dois temas, entre os três propostos. A escolha é livre.

Tema do Projeto: resolver o problema **das quatro cores para os estados do Brasil** utilizando o algoritmo de *Otimização por Enxames de Partículas (PSO)* ou o algoritmo de *Otimização por Evolução Diferencial (DE)*. De maneira alternativa, resolver o problema de **distribuição de postos de combustíveis no deserto**. É para resolver apenas um deles. A escolha é livre. Na próxima seção, por questões de didática, é explicado como resolver o problema das oito rainhas através de algoritmos genéticos. Em seguida, são descritos os dois problemas anteriores propriamente ditos.

Descrição Breve do Problema das Oito Rainhas Através de Algoritmos Genéticos

Os algoritmos genéticos (AG) e outras variantes (e.g., PSO e DE) começam com um conjunto de k estados gerados aleatoriamente, chamado **população**. Cada estado, ou **indivíduo**, é representado como uma cadeia sobre um alfabeto finito. Por exemplo, para o problema das 8 rainhas, um estado de 8 rainhas deve especificar as posições das 8 rainhas. Assim, o estado poderia ser representado como 8 dígitos, cada um no intervalo de 1 a 8. A Figura 01.(a) mostra uma população de quatro cadeias de 8 dígitos que representam estados de 8 rainhas. A explicação dos parágrafos seguintes é para o caso particular dos algoritmos genéticos.

A produção da próxima geração de estados é mostrada na Figura 01.(b)-(e). Em (b), cada estado é avaliado pela função de avaliação ou (na terminologia do AG) pela **função de fitness**. Uma função de fitness deve retornar valores mais altos para estados melhores; assim, para o problema da 8 rainhas, usamos o número de pares de rainhas *não-atacantes*, que têm o valor 28 para uma solução. Os valores dos quatro estados da Figura 01.(a) são 24, 23, 20 e 11. Nessa variante específica do algoritmo genético, a probabilidade de um indivíduo ser escolhido para reprodução é diretamente proporcional à sua pontuação de fitness, e as porcentagens são mostradas ao lado das pontuações brutas na Figura 01.(b).

Em 01.(c), dois pares escolhidos aleatoriamente são selecionados para reprodução, de acordo com as probabilidades mostradas em 01.(b). Note que um indivíduo é selecionado duas vezes, e um indivíduo não é selecionado de modo algum. Para cada par a ser cruzado é escolhido ao acaso um ponto de **crossover** dentre as posições na cadeia. Na Figura 01.(c), os pontos de crossover estão depois do terceiro dígito no primeiro par e depois do quinto dígito no segundo par.

Em 01.(d), os próprios descendentes são criados por crossover das cadeias pais no ponto de crossover. Por exemplo, o primeiro filho do primeiro par recebe os três primeiros dígitos do primeiro par e os dígitos restantes do segundo pai, enquanto o segundo filho recebe os três primeiros dígitos do segundo pai e o restante do primeiro par (ver Figura 02).

Finalmente, em 01.(e), cada posição está sujeita à **mutação** aleatória com uma pequena probabilidade independente. Um dígito sofreu mutação no primeiro, no terceiro e no quarto descendente. No problema de 8 rainhas, isso corresponde à escolha de uma rainha ao acaso e à movimentação da rainha para um quadrado aleatório em sua coluna.

Esses primeiros parágrafos exemplificaram o funcionamento da procura pela solução do problema das oito rainhas utilizando algoritmos genéticos. Portanto, cada grupo de aluno deverá adaptar o enunciado destes primeiros parágrafos para o problema ser resolvido via Otimização por Enxame de Partículas (PSO) e por Evolução Diferencial (ED). Por exemplo, a Figura 01 explica perfeitamente a função de custo

deste problema utilizando GA. Essa mesma função de custo poderá ser utilizada diretamente em PSO e em DE.



Figura 01 O algoritmo genético. A população inicial em (a) é classificada pela função de fitness em (b), resultando em pares de correspondência em (c). Eles produzem descendentes em (d), sujeitos à mutação em (e).

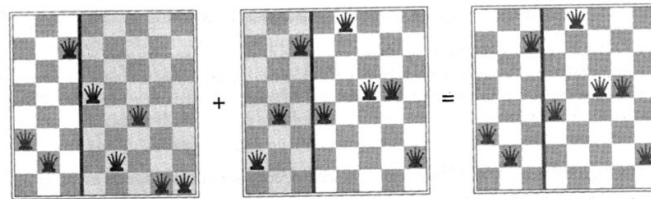


Figura 02 Os estados de 8 rainhas correspondentes aos dois primeiros pais na Figura 01.(c) e à primeira descendência da Figura 01.(d). As colunas sombreadas foram perdidas na etapa de crossover, e as colunas não-sombradas foram preservadas.

O uso bem-sucedido de algoritmos genéticos (GA), Otimização por Enxames de Partículas (PSO) e Evolução Diferencial (DE) exige uma cuidadosa engenharia na representação. Na prática, os algoritmos genéticos tiveram um amplo impacto sobre problemas de otimização, como layout de circuitos e escalonamento de prestação de serviços. No momento, não está claro se a atração de algoritmos genéticos surge de seu desempenho ou de suas origens esteticamente agradáveis na teoria da evolução. Ainda há muito trabalho a ser feito para identificar as condições sob as quais os algoritmos genéticos funcionam bem.

O Problema das Quatro Cores no Colorimento dos Estados da Austrália

3.3.2 Descrição do Algoritmo

O algoritmo estocástico PSO é inspirado pela natureza e defini o comportamento de enxame sociais, isto é, as partículas transmitem informações entre si. Assim, as atualizações de posição e velocidade de cada partícula dependem de suas experiências individuais e das experiências coletadas pelas outras partículas do enxame. O objetivo de cada partícula é encontrar a melhor solução entre todas as possíveis. Dessa forma, o algoritmo deve seguir os seguintes passos:

1. Inicialização: primeiramente, um enxame de partículas é criado e cada partícula recebe uma posição e velocidade iniciais. O número de partículas no enxame geralmente varia no intervalo $[20,60]$, porém, de acordo com a literatura, o tamanho do enxame dificilmente produz qualquer efeito sobre a performance do algoritmo em questão (EBERHART; SHI, 2000).
2. Definir a Função de *Fitness*: esta função deve ser definida para atribuir um valor de *fitness* para cada partícula.
3. Avaliação: calcular o *fitness* associado a cada partícula do enxame.
4. Atualizar melhor solução local: para cada partícula, comparar o seu valor de *fitness* atual com o seu melhor resultado individual. Se for melhor, deve-se atualizar a melhor solução local.
5. Atualizar melhor solução global: identificar a partícula com o melhor valor de *fitness* dentre todas as partículas do enxame. Este valor representa a melhor solução conhecida até o momento.
6. Atualizar posição e velocidade: para cada partícula, seguindo as equações 3.1 e 3.2, atualizar o valor da posição e velocidade. Essas equações determinam como a partícula se move no espaço de busca.
7. Condição de parada: o algoritmo deve continuar a rodar até que a condição de parada é atingida. Esta condição pode ser que o máximo ou mínimo global da função de fitness foi atingido ou que o algoritmo percorreu um número máximo de iterações, por exemplo. Se a condição de parada ainda não foi atingida, o algoritmo deve retornar ao passo 3.

Considerando os parâmetros e equações definidas acima, segue logo abaixo um pseudocódigo para o PSO.]

Data: $x_i(0)$ e $v_i(0)$ inicializadas randomicamente
Result: Posição aproximada da solução global X
 $N \leftarrow$ número de partículas;
while *condição de término não acontece* **do**
 for $i=1$ até N **do**
 calcula a função de fitness $F(X_i)$;
 atualiza p_i e g_i ;
 atualiza a velocidade da partícula pela equação 3.1;
 atualiza a posição da partícula pela equação 3.2 ;
 end
 incrementa i ;
end

Algorithm 2: Pseudo-código para o PSO clássico

Aplicar o algoritmo PSO no contexto do T4C envolve aplicar cada solução possível como uma partícula em um enxame. As cores associadas a cada vértice do grafo correspondem à posição da partícula no espaço de soluções de forma que cada partícula representa um possível colorimento do grafo.

Vantagens da aplicação do PSO:

- o algoritmo PSO permite que as partículas explorem uma gama de configurações de colorimento possíveis enquanto se mantém o foco em outras soluções promissoras, o que é crucial para problemas de colorimento de mapas;
- por ser um algoritmo estocástico, a aleatoriedade ao se atualizar a posição e velocidade das partículas ajudam a escapar de soluções locais, o que aumenta a chance de se encontrar configurações válidas em grafos complexos.
- como múltiplas partículas atuam de maneira simultânea, o algoritmo pode acelerar o processo de busca de soluções;
- os parâmetros do algoritmo, como o peso inercial ω e os coeficientes C_1 e C_2 podem ser atualizados por motivos de otimização.

Possíveis Desvantagens:


- ao conceber uma função de *fitness* que cumpre os requisitos do T4C, o algoritmo pode exigir recursos computacionais demasiados;

- dependendo da complexidade do grafo, pode ser difícil garantir que o algoritmo convirja em uma solução válida dentro de um espaço de tempo razoável.

A Função de *Fitness*

A função de *fitness* utilizada no PSO visa a percorrer para cada vértice do grafo e calcular o número total de confrontos, isto é, os casos em que vizinhos adjacentes possuem a mesma cor. Dessa maneira, o objetivo será minimizar o valor desta função para cada partícula a fim de se obter uma solução global ótima em que número de confrontos é zero, o que garante que o T4C foi respeitado.

A Figura abaixo exibe uma solução para o problema das quatro cores para os estados da Austrália.





Problemas de Satisfação de Restrições

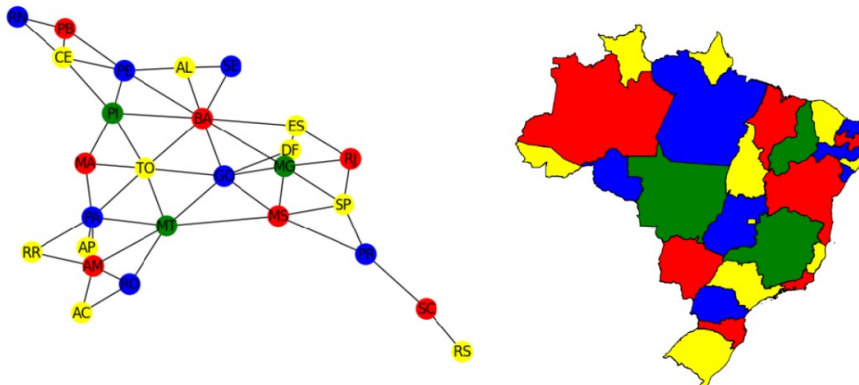
Capítulo 5

Caso Discreto: Busca com Retrocesso

- **Restrição Associada aos Vínculos:** Nós vizinhos não podem possuir a mesma COR.
- Uma solução possível:

A Função de Fitness é construída a partir do grafo de fronteiras entre os estados (ver o exemplo do grafo do Brasil abaixo). Dois vizinho não podem ter a mesma cor. Deve-se então minimizar o número de estados vizinho com a mesma cor.



O Problema de Distribuição de Postos de Combustível no Deserto

O problema que se pretende analisar e resolver é o do “*Caminhão no Deserto*” (Wood, 1986 e 1988) e que pode ser enunciado da seguinte forma:

“Um caminhão atinge a extremidade de um deserto de 400 milhas de extensão. O veículo faz apenas 1 milha com um galão de gasolina, e a capacidade total do caminhão, incluindo bujões adicionais, é de 180 galões; assim sendo, terão de ser providenciados depósitos provisórios no deserto. Existe gasolina à vontade na extremidade do deserto. Se fizermos um bom planejamento, qual é o menor consumo de gasolina necessário para que o veículo consiga atravessar o deserto?”

Wood, L. E. *Estratégias do Pensamento: Técnicas de Aptidão Mental*. São Paulo, SP, Brasil: Círculo do Livro, 1988.

Exemplo de função objetivo para otimização

Um exemplo de função objetivo não linear para otimização é mostrado abaixo:

$$f_3(\mathbf{x}) = -20 \exp \left\{ -0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2} \right\} - \exp \left\{ \frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right\} + 20 + e$$

Esta é uma função que pode ter dimensão n e é um *benchmark* para testar se um algoritmo de otimização é eficaz. Além disso, pode ser incluídas restrições, limitando o espaço de busca de \mathbf{x} . Essa função é usada para minimização.

Descrição do Corpo do Relatório

1. **Capa** (Nome da Instituição, Nome da Disciplina, Título, Nome do Professor, Integrantes do Grupo e Data/local).
2. **Corpo do Relatório**
 - Introdução (máximo de uma página contendo enunciado do problema e outras informações relevantes que se acharem necessárias)
 - Objetivo (máximo de uma página)
 - Descrição técnica da metodologia utilizada para resolver o problema proposto
 - Análise e apresentação dos resultados (pelo menos três casos de teste)
 - Conclusão
3. **Apêndice A**: um pequeno resumo e/ou descrição da linguagem utilizada
4. **Apêndice B**: listagem completa do código fonte

Boa Sorte ☺!

Prof. Paulo Marcelo Tasinaffo.

DCTA-ITA-IEC Divisão de Ciência da Computação.

Sala 107, TEL: +55 12 3947-6945.

e-mail: tasinaffo@ita.br.