Trabalho 2 - Algoritmos Genéticos

Inteligência Artificial - 2024.2

Prof. Samy Sá

Universidade Federal do Ceará
Campus de Quixadá

Obs.: Para completar este trabalho, será necessário implementar algumas variações de algoritmos genéticos, experimentar com eles e escrever um relatório com suas conclusões. Em caso de plágio em qualquer parte deste trabalho, os envolvidos terão suas notas automaticamente zeradas. Em caso de reincidência, os envolvidos serão imediatamente reprovados na disciplina com nota 0,0 e 64 faltas.

1 Introdução

Este projeto visa a aplicação de Algoritmos Genéticos para a solução de um problema difícil de natureza combinatória. Conforme combinado em sala de aulas, cada equipe pode utilizar esta técnica para o *Problema Quadrático de Alocação* (**PQA**), que será descrito nesse documento, ou para um problema de sua escolha (que deve ser comunicado a mim e aprovado). Neste último caso, será necessário apresentar a modelagem do problema proposto junto aos itens pedidos. Em ambos os casos, será necessário propor variações de algoritmos genéticos e gerar entradas aleatórias do problema afim de comparar a performance das variações propostas.

Ao longo desde documento, começarei apresentando os passos necessários para trabalhar o *Problema Quadrático de Alocação*. Os requisitos do trabalho serão idênticos nos casos em que a equipe escolha outro problema. Isso será comentado de forma mais direta ao fim deste documento.

2 O Problema Quadrático de Alocação

O Problema Quadrático de Alocação (**PQA**) é um problema de natureza combinatória com duas dimensões relevantes: distância entre pontos (locais) e demandas de fluxo entre instalações (objetos) que devem ocupar estes locais.

Formalmente, uma entrada considera a existência de n objetos e n locais. Para cada par de locais x e y, conhecemos a distância $d(x, y)$ entre eles, representada aqui por um número inteiro (por simplicidade). Para cada par de objetos a e b, é especificado um valor $f(a, b)$ que chamamos de *peso* ou de *fluxo*, também representado por um número inteiro. O problema envolve alocar os n objetos nestes n locais de forma a minimizar o custo total de fluxo no sistema, que é dado pela função

$$custo(I, L) \;\; = \sum_{a,\, b \,\in\, I} f(a, b) \;.\; d(l(a), l(b))$$

onde I é o conjuto de objetos, L é o conjunto de locais, e $l : I \to L$ é uma função de alocação que determina, para cada objeto, em qual local ele deve ser fixado. Dessa forma, o problema visa caracterizar a melhor função de alocação l que minimiza o valor de $custo(I, L)$.

Exemplo 1 *Uma empesa está reformando seu prédio e pretende reorganizar a distribuição dos seus espaços afim de melhorar a produtividade dos seus colaboradores. Há três locais A, B, C na empresa que podem servir a múltiplos propósitos e três instalações cujo uso podem influenciar a produtividade: (1) a sala de refeições (bebedouro, geladeira), (2) a sala de cópias (impressora), (3) os banheiros.*

É comum que os colaboradores dessa empresa costumem deixar suas estações de trabalho apenas quando precisam utilizar mais de uma dessas instalações, por isso foram observados os flucos entre elas. As distância entre os três locais e o fluxo entre as instalações são dados pelas matrizes:

$d(x,y)$	A	B	C
A	0	10	15
B	10	0	12
C	15	12	0

$f(a,b)$	1	2	3
1	0	3	6
2	3	0	1
3	6	1	0

Uma possível solução para o problema seria a alocação l propondo $l(1) = A$, $l(2) = B$, $l(3) = C$, ou seja, colocarmos a sala de refeições (1) no local A, a sala de cópias (2) no local B e os banheiros (3) no local C.

Nesta alocação, o custo pra resolver o fluxo entre as instalações 1 e 2 será

$$f(1,2) \,.\, d(l(1), l(2)) = f(1,2) \,.\, d(A, B) = 3 \,.\, 10 = 30.$$

Por sua vez, o custo total será

$$\begin{aligned} custo(I, L) &= \sum_{a,\, b \,\in\, \{1,2,3\}} f(a,b) \,.\, d(l(a), l(b)) \\ &= f(1,2) \,.\, d(l(1), l(2)) + f(1,3) \,.\, d(l(1), l(3)) + f(2,3) \,.\, d(l(2), l(3)) \\ &= f(1,2) \,.\, d(A, B) + f(1,3) \,.\, d(A, C) + f(2,3) \,.\, d(B, C) \\ &= 3 \,.\, 10 \;+\; 6 \,.\, 15 \;+\; 1 \,.\, 12 \\ &= 30 \;+\; 90 \;+\; 12 \\ &= 132. \end{aligned}$$

Note que essa proposta de alocação sugere colocarmos o par de instalações cujo fluxo é maior nos locais mais distantes entre eles. Logo, esperamos que haja uma

solução melhor. Como exemplo, uma alocação l' gulosa com ajuste heurístico que priorize dispor as instalações de maior fluxo nos lugares mais próximos, por ordem decrescente de fluxo, proporia $l'(1) = B$, $l'(2) = C$, $l'(3) = A$. Saltando os primeiros passos, o custo total dado por l' será

$$\begin{aligned} custo(I, L) &= \sum_{a,\ b\ \in\ \{1,2,3\}} f(a,b)\ .\ d(l(a), l(b)) \\ &= f(1,2)\ .\ d(l'(1), l'(2)) + f(1,3)\ .\ d(l'(1), l'(3)) + f(2,3)\ .\ d(l'(2), l'(3)) \\ &= f(1,2)\ .\ d(B, C) + f(1,3)\ .\ d(B, A) + f(2,3)\ .\ d(C, A) \\ &= 3\ .\ 12\ +\ 6\ .\ 10\ +\ 1\ .\ 15 \\ &= 36\ +\ 60\ +\ 15 \\ &= 111. \end{aligned}$$

Dessa forma, a alocação l' é melhor que a alocação l.

Com estes exemplos, espera-se que fique claro como o custo total do sistema é calculado e como a posição das instalações afeta esse custo total. O exemplo acima tem apenas três locais e instalações, mas as instâncias reais do problema podem envolver dezenas de objetos e locais.

O PQA é um problema representativo da classe NP de problemas (é NP-Difícil) e é considerado um dos problemas fundamentais de otimização combinatória. Tem inúmeras aplicações em logística, eletrônica e pesquisa operacional. As posições das teclas no teclado padrão de computadores que utilizamos, inclusive, é uma aplicação deste problema. Apesar da sua utilidade, suas instâncias admitem uma representação bastante simples, pois o problema assume apenas apenas grafos completos (um dos locais, um de fluxo). Como no exemplo acima, tudo o que precisamos como entrada é um par de matrizes $n \times n$.

OBSERVAÇÃO 1: O problema em si admite variações. Por exemplo, é possível ter mais locais candidatos do que instalações pra serem alocados, de forma que a alocação fornecida decide quais são os melhores lugares para usar. Outra possibilidade é que os caminhos entre os locais não sejam imediatamente simétricos por design (bloqueios, vias de mão única, etc.), possibilitando que a distância do ponto A ao ponto B seja diferente da distância do ponto B ao ponto A. Neste projeto, assumiremos o formato mais simples, onde a quantidade de objetos e locais é a mesma (n) e tanto os valores de distância como os de fluxo são simétricos. Isso permitirá contabilizar a função de custo de forma bem mais simples, como no exemplo acima.

OBSERVAÇÃO 2: Note que a definição da função de custo pede, em princípio, por todas as combinações de valores para x, y, num total de nove possibilidades que incluem, entre outras, o cálculo do custo de fluxo da instalação 1 para a instalação 1. Esse custo

será zero, pois a distância de 1 para 1 será necessariamente zero. O cálculo no exemplo é simplificado, pois podemos ignorar as instâncias com valores iguais para x e y, bem como, por conta da simetria de valores, contabilizar o custo entre as instalações 1 e 2 e não contabilizar o custo entre 2 e 1 (que é idêntico). Isso nos permitirá economizar alguns passos no cálculo da função total de custo do sistema.

3 Entradas e Soluções do Problema

Como em todo problema de IA, consideramos como as soluções do problema podem ser representadas. Isto é o que nos dá a coleção de estados em um espaço de busca. Diferente dos algoritmos que trabalhamos antes, os Algoritmos Genéticos não exploram o espaço de busca navegando entre estados vizinhos. Essa técnica utiliza elementos de randomização para navegar o espaço de busca de maneira errática na expectativa de que isso revele um bom candidato à solução do problema.

Estados do Espaço de Busca. No QPA, o objetivo é gerar uma função de alocação de n objetos em n locais, o que corresponde a uma *função bijetora*. Dessa maneira, podemos assumir que os n locais estão organizados em uma ordem fixa e listarmos apenas em que ordem os objetos devem ser organizados. Isso nos permitirá tratarmos os espaços do estado de busca como permutações simples das instalações desejadas.

No Exemplo 1, se assumirmos que os locais serão preenchidos na ordem A, B, C, a alocação l (que propôs $l(1) = A$, $l(2) = B$, $l(3) = C$) poderá ser representada de forma simplificada pela permutação $(1, 2, 3)$. Já a alocação l' (que propôs $l'(1) = B$, $l'(2) = C$, $l'(3) = A$) poderá ser representada pela permutação $(3, 1, 2)$.

Representação do Problema. A implementação típica deste problema envolve vetores e matrizes, especialmente com a representação de estados como permutações dos objetos. Dessa forma, recomendo referir-se aos n locais pelos valores inteiros $0, 1, 2, \ldots, n-1$ e, igualmente, referir-se aos n objetos pelos valores inteiros $0, 1, 2, \ldots, n-1$. Isso facilitará lidar com os índices nas matrizes e vetores e o contexto de uso das variáveis deve ser suficiente para diferenciar no seu código quando se tratarem de locais ou objetos. Outras opções que podem facilitar a sua leitura e compreensão dos códigos enquanto os estiver desenvolvendo, então sinta-se à vontade para mudar esses rótulos. A sugestão acima serve apenas como referência do mínimo necessário para tratar os objetos e locais. No restante do documento, assumiremos o uso da representação acima.

Entradas do PQA. Uma entrada de tamanho n envolve duas matrizes de tamanho $n \times n$:

1. Uma matriz d de distâncias entre cada par de locais $x, y \in \{0, 1, 2, \ldots, n-1\}$.
2. Uma matriz f de fluxo entre cada par de instalações $x, y \in \{0, 1, 2, \ldots, n-1\}$.

Para gerar uma entrada aleatória de tamanho n, você deve gerar n pares de coordenadas aleatórias e calcular a Distância Euclidiana entre esses pontos. Para simplificar mais as coisas, recomendo usar o *piso* da distância euclidiana, pois isso garantirá valores

inteiros. Recomendo também limitar os valores das coordenadas ao grid que usamos no primeiro trabalho com valores inteiros no intervalo $[0, 30]$. Isso permitirá que você reutilize partes do código do primeiro trabalho para obter a matriz de distâncias.

Lembrando que a Distância Euclidiana entre dois pontos (x_1, y_1) e (x_2, y_2) será

$$\texttt{euc}((x_1, y_1), (x_2, y_2)) = \sqrt{|x_1 - x_2|^2 + |y_1 - y_2|^2}$$

Note que isso é o mesmo que calcular as distâncias entre (x_2, y_2) e (x_1, y_1), de forma que o mesmo cálculo pode ser usado para preencher duas posições da matriz. Além disso, a diagonal principal de uma matriz de distâncias será sempre nula.

Para obter uma matriz de fluxo, basta preencher a matriz aleatoriamente com números inteiros aleatórios, preservando a simetria e a diagonal principal nula. Sugiro manter esses valores entre os inteiros do conjunto $\{0, 1, 2, 3, \ldots, 2n - 1, 2n\}$.

4 Implementação

A técnica de Algoritmos Genéticos permite ampla variação nas implementação de funções de emparelhamento, crossover, fitness, elitismo e mutação, entre outras possibilidades.

Para este projeto, você deve propor variações de algoritmos genéticos contemplando pelo menos:

1. duas formas diferentes de emparelhamento dos indivíduos (seleção)
2. duas formas diferentes de efetuar o crossover (reprodução)
3. duas formas diferentes de promover elitismo entre gerações
4. duas formas diferentes de promover mutação

As suas escolhas para cada parte desses algoritmos devem ser discutidas no relatório que acompanhará a entrega dos códigos. Quais foram as suas intuições? Por que decidiram por estas formas de implementar cada aspecto dos algoritmos genéticos?

No total, as combinações dessas escolhas lhe dará um conjunto de 16 algoritmos genéticos diferentes, sem contar a possibilidade de varias os parâmetros como tamanho das gerações, critério de parada, taxas de elitismo e mutação...

A próxima sessão lhe pedirá para escolher algumas variações por vez para realizar experimentos com entradas aleatórias, de forma que cada comparação vá ser feita entre duas a quatro variações dos algoritmos.

OBSERVAÇÃO 3: A função de fitness, nesse caso, será a função de custo total do sistema $custo(I, L)$, especificada na Seção 2. Para este problema, quanto menor o valor da função, melhor será o fitness de cada indivíduo.

OBSERVAÇÃO 4: Na representação que discutimos, cada indivíduo nestes algoritmos será uma string de inteiros sem repetição. Cada indivíduo lista os mesmos inteiros

$(0, 1, \ldots, n-1)$ em alguma ordem, consistindo, portanto em permutações da lista de instalações que serão alocadas. Ao efetuar crossover entre esses indivíduos, é possível que o resultado tenha números repetidos, portanto será necessário corrigir as strings geradas pelo crossover para que se refiram a estados válidos do problema (strings válidas). Propor uma maneira de corrigir é uma decisão de implementação e, por isso, é parte deste trabalho.

Saídas. Os valores que estaremos interessados em ver para cada experimentos vão depender um pouco das variações que você propuser. No geral, é interessante anotar dados como:

- Fitness do indivíduo mais apto em cada geração (imprimir ao longo da execução e/ou como lista ao final)
- Fitness médio de cada geração (imprimir ao longo da execução e/ou como lista ao final)
- Fitness do indivíduo menos apto em cada geração (imprimir ao longo da execução e/ou como lista ao final)

Esses dados permitirão estabelecer se as gerações da população estão melhorando em fitness ao longo das operações, algo que pode significar eficácia do algoritmo proposto. Se isso acontecer, estes números também podem ser usados para medir em que velocidade, em média, cada uma dessas abordagens promove melhorias nas gerações. Organize a saída conforme observar quais números e medidas lhe ajudam em qualquer parte desse trabalho.

Documentação Dos Códigos. É importante que o seu código seja legível e bem comentado para a sua avaliação. Além de comentar seu código, adicione um arquivo `readme.txt` ao folder com seus códigos com as instruções necessárias para executá-lo e para rodar os diferentes experimentos. Caso se aplique, indique a versão do compilador/interpretador utilizado e se é necessário instalar alguma biblioteca ou recurso adicional para rodar seu código.

5 Experimentação

Espera-se que você experimente com as variações de algoritmos genéticos que tiver proposto para avaliar, dentro do possível, quais variações parecem melhores para este problema. Lhes encorajo procurar estabelecer livremente maneiras de medir a performance destas variações de algoritmos para compará-las.

Parte 0: Escolha de Parâmetros Numéricos

Os algoritmos genéticos normalmente recebem uma variedade de parâmetros inteiros, quais como o tamanho da população inicial, a quantidade de gerações a serem reproduzidas (se esse for o critério de parada), valores de taxas de elitismo e mutação... As variações são muitas e alguns parâmetros só existirão em algumas variações desses algoritmos.

Neste começo, que também é fundamental, desejamos encontrar uma boa combinação de valores destes parâmetros numéricos para utilizar como padrão nos experimentos seguintes. Escolha uma variação para tomar como referência e a execute para entradas aleatórias variando apenas estes parâmetros numéricos. Como a performance do seu algoritmo responde a variações no tamanho da população? E ao número de gerações reproduzidas? Avalie em busca de um conjunto de valores que tenha boa performance em tempo e qualidade das respostas.

OBSERVAÇÃO 5: Como estamos trabalhando com entradas aleatórias, será necessário rodas o mesmo algoritmo para a mesma entrada inicial (par de matrizes de distância e fluxo) várias vezes, comparando as melhores respostas retornadas em cada execução com a melhor resposta retornada entre todas as execuções. A única maneira que teremos de medir a qualidae de uma resposta retornada para cada entrada é de forma relativa, pois não teremos uma maneira de calcular a resposta ótima.

Parte 1: Variação do Parâmetro de Seleção

O propósito deste experimento é intuir se alguma das funções de seleção que você implementou tem impacto na performance dos algoritmos.

Escolha duas variações de algoritmos genéticos entre as implementadas para as quais as funções de emparelhamento (seleção) são diferentes, mas todas as outras partes dos algoritmos são iguais.

Gere uma entrada aleatória de tamanho 10 e execute cada uma das variações selecionadas para esta entrada.

Repita 20 vezes, anotando a cada execução de algoritmo qual foi o indivíduo mais apto encontrado por cada variação e o seu valor de fitness.

Analise os números encontrados e busque identificar qual destas duas variações se mostrou melhor que a outra.

Partes 2-4: Variações nos Demais Parâmetros

Estes experimentos seguem o mesmo formato da Parte 2, mas com escolhas diferentes de parâmetros em cada caso:

- Na Parte 2, escolha duas variações implementadas nas quais as funções de crossover são diferentes, mas todas as outras partes dos algoritmos são iguais.
- Na Parte 3, escolha duas variações implementadas nas quais as funções elitismo são diferentes, mas todas as outras partes dos algoritmos são iguais.
- Na Parte 4, escolha duas variações implementadas nas quais as funções de mutação são diferentes, mas todas as outras partes dos algoritmos são iguais.

Cada experimento pode pedir por formas diferentes de medir e comparar sua performance, visto que estamos tentando determinar se as diferenças nos parâmetros propostos

são melhores em cada caso. Em cada um destes experimentos, procure identificar, de forma justificada, qual variação tem melhor performance.

Parte 5: Tamanho Máximo de Entrada Viável

Este primeiro experimento foi pensado para que você possa observar o impacto do tamanho de entrada sobre o tempo de execução dos algoritmos e determinar, pelo menos intuitivamente, o tamanho da maior instância do PQA que você conseguiria tratar por Algoritmos Genéticos.

Seguindo os experimentos anteriores, você deve ter identificado quatro variações de algoritmos genéticos campeãs, uma variação por experimento. Utilize estas quatro variações no experimento seguinte. À medida que essas variações compartilharem parâmetros de entrada, escolha valores que achar adequados para esses parâmetros e os mantenha fixos.

Repita os passos abaixo para valores incrementais de n a partir de 10:

(a) Gere uma entrada aleatória de PQA com tamanho n
(b) Execute cada variação escolhida sobre essa entrada
(c) Anote o tempo que cada variação levo para concluir sua execução

Incremente n até que as variações do algoritmo comecem a demorar demais (de acordo com a sua avaliação) para chegar a uma conclusão.

6 Relatório

Procurem ser concisos e técnicos no que forem relatar. Discutam que tipo de testes vocês fizeram para avaliar se as implementações estão corretas e o que a equipe foi capaz de observar em cada experimento. Divida seu relatório de acordo com as seções:

1. **Implementação** - Linguagem utilizadas, bibliotecas e frameworks relevantes, o que é necessário para rodar seus algoritmos e como eu devo proceder para rodá-los com parâmetros de teste escolhidos por mim.

2. **Design dos Algoritmos** - Discuta as variações que você propôs para cada parte dos seus algoritmos genéticos, justificando suas escolhas. Detalhe o processo que usou para escolher os parâmetros numéricos na **Parte 0** da experimentação.

3. **Experimentação** - Divida em subseções nomeadas com **Parte 1** até **Parte 5**. Em cada uma destas, discuta o que foi possível observar na execução dos experimentos e nos dados gerados. Conforme necessário, havendo arquivos diferentes relacionados à sua análise, indique quais são para que eu possa visualizá-los. Para cada experimento, indique o que você utilizou como parâmetros para comparar as variações de algoritmos disponíveis. É importante que estas comparações sejam suportadas pelos dados e

baseadas em métodos claros. Medidas estatísticas podem ajudar bastante em cada parte destas análises.

4. **Considerações Finais** - Caso desejem, podem adicionar uma seção com discussão de resultados de forma mais ampla: comparações entre as diferentes partes do experimento, outras variações que vocês tenham considerado, experimentação com valores diferentes de parâmetros (taxas de elitismo e mutação, tamanho de população,...), etc. Esta seção é inteiramente opcional.

7 Pontuação

Para uma avaliação adequada, é necessário que eu possa observar os seus códigos sendo executados e, idealmente, as populações que são geradas a cada nova geração.

- 1,0 ponto: todos os requisitos implementados (experimentos em loop e possibilidade de repetí-los manualmente)
- 1,0 ponto: documentação adequada dos códigos
- 1,5 ponto: implementação adequada da técnica
- 1,5 ponto: discussão das escolhas que a equipe fez para as variações de algoritmos genéticos e valores de parâmetros
- 5,0 pontos: até um para cada parte de 1 a 5 no relatório (avaliação independente)

Para evitar excesso no número de trabalhos para correção, é mandatório que cada equipe seja composta por três pessoas, enquanto possível. Para reforçar essa diretriz, a soma dos pontos obtidos será multiplicada por:

- $1,00$ para trabalhos submetidos por uma equipe com três pessoas
- $0,75$ para trabalhos submetidos por uma equipe com apenas duas pessoas
- $0,50$ para trabalhos submetidos por uma equipe com apenas uma pessoa
- $0,50$ para trabalhos submetidos por uma equipe com mais que três pessoas

Como a quantidade de estudantes da turma não é um múltiplo de três, é possível que tenhamos uma ou duas equipes com menos que três pessoas ao final. Esses casos só serão aceitos quando não houverem mais colegas disponíveis para formar equipes e tratados diretamente comigo. Equipes nessa situação não serão penalizadas por terem menos membros.

OBS.: As equipes com menos que três estudantes no primeiro trabalho podem ser mantidas sem penalidade na nota.

8 Abordagem de Outro Problema

Opcionalmente, a equipe pode escolher trabalhar com um problema diferente do PQA. Nesse caso, será necessário apresentar a modelagem do problema que será utilizada. Como neste documento, é necessário justificar como os indivíduos serão descritos e como será a função de fitness.

À parte da modelagem, as equipes podem (e devem) seguir os mesmos moldes no que se refere às variações de algoritmos genéticos e à experimentação com elas.