AT3: Algoritmos Genéticos

Inteligência Artificial
Período: 2023.2

Samy Sá

Universidade Federal do Ceará
Campus de Quixadá
Quixadá, Brasil
samy@ufc.br

Requisitos: Conhecimentos de Programação e Algoritmos Genéticos.
Prazo: **08/12/23**

1 Introdução

Este documento descreve a terceira atividade da disciplina e consiste em uma aplicação de Algoritmos Genéticos. Utilizaremos esta técnica para trabalhar o problema de alocação de profissionais em turnos de trabalho. Para uma melhor compreensão da técnica, sugere-se realizar simulações com um problema pequeno e depois seguir para uma instância com tamanho maior, como essa que desejamos resolver.

Este enunciado é inspirado nas tarefas disponibilizadas em `https://www.cs.usfca.edu/~brooks/F06classes/cs662/assignments/assignment4.html` da disciplina de Programação em Inteligência Artificial do Prof. Chris Brooks, University of San Francisco, EUA.

2 Enunciado do Problema

O objetivo deste trabalho é atacar o problema de alocação de enfermeiros nos turnos de trabalho de um hospital. Para a representação dos estados do espaço de busca, assuma que há k enfermeiros denominados $e_1, e_2, \ldots, e_k$, com n turnos $t_1, t_2, \ldots, t_n$, e um conjunto de m restrições $r_1, r_2, \ldots, r_m$ que desejamos satisfazer em uma alocação adequada. Dadas duas possíveis alocações de enfermeiros em turnos, a melhor das duas será aquela que mais se aproximar de satisfazer todas as restrições. Ou seja: na impossibilidade de satisfazer a todas as restrições, devemos buscar a alocação que mais se aproximar desta meta.

2.1 Representação de Estados

Os estados do problema podem ser facilmente visualizados na forma de uma matriz com enfermeiros nas linhas e turnos nas colunas. Neste caso, cada célula na matriz poderá conter o número 1, indicando que o enfermeiro está alocado naquele turno, ou o número 0, indicando que ele não está. Por exemplo, em um problema com 3 enfermeiros e 3 turnos, a seguinte matriz representa um dos estados no espaço de busca:

	t_1	t_2	t_3
e_1	1	0	1
e_2	1	0	0
e_3	0	1	0

Este estado sugere que deve-se alocar o enfermeiro e_1 nos turnos t_1 e t_3, enquanto e_2 deve ser alocado somente ao turno t_1 e e_3 somente ao turno t_2.

A matriz pode ser representada em forma de string, que é o formato que discutimos para a representação de estados em algoritmos genéticos: 101100010 . Para isso, basta concatenar as linhas da matriz uma após a outra.

2.2 Restrições e Fitness

As restrições do problema devem ser modeladas como funções para contabilizar quantas violações são violadas por cada estado. Considere, por exemplo, a restrição r_1 : de que "cada enfermeiro deve trabalhar somente 1 turno". Para esta restrição, a matriz do nosso exemplo de estado (codificada como "101100010") fornece 1 violação, pois o enfermeiro e_1 estaria sendo alocado em dois turnos. A função relacionada a essa restrição deve, então, retornar "-1" como avaliação deste estado. Caso o estado propusesse alocar dois enfermeiros diferentes em mais que um turno, a avaliação desta restrição deve retornar "-2".

No caso de múltiplas restrições no problema, a função de fitness deve somar os valores retornados por cada uma dessas restrições. Estes números devem ser utilizados para retornar porcentagens de fitness no seu algoritmo genético.

3 Requerimentos de Código

Seu trabalho é implementar a técnica de algoritmo genético para rodar simulações em problemas de alocação. Planeje seu código para que seja possível modificar facilmente os seguintes parâmetros:

1. Instância do problema, ou seja, os números k de enfermeiros e n de turnos de trabalho.
2. O conjunto de retrições, ou seja, a composição da função de fitness utilizada para avaliar indivíduos (mais detalhes ao fim do arquivo).
3. O tamanho da população.
4. O número de gerações (iterações) da simulação.
5. As taxas de mutação e elitismo.

As mudanças para instâncias do problema e das restrições utilizadas podem ser modificáveis apenas em nível de código, mas o seu código deverá permitir que um usuário ajuste os demais parâmetros (tamanho da população, número de iterações, e taxas de mutação e elitismo) diretamente na interface do programa.

4 Enunciado Principal, Fitness, e Experimentação

Você deve utilizar seu código e compor uma função de fitness para a seguinte instância do problema: construir a alocação de enfermeiros para uma semana em um certo hospital. O hospital tem 3 turnos de 8h de trabalho por dia, todos os dias da semana, totalizando 21 turnos $t_1, t_2, \ldots, t_{21}$ para alocação de pessoal. Além disso, existem 10 enfermeiros $e_1, e_2, \ldots, e_{10}$ que trabalham neste hospital e deve-se buscar satisfazer as seguintes restrições:

- r_1: É necessário haver no mínimo 1 enfermeiro e no máximo 3 enfermeiros em cada turno.
- r_2: Cada enfermeiro deve ser alocado em 5 turnos por semana.
- r_3: Nenhum enfermeiro pode trabalhar mais que 3 dias seguidos sem folga.
- r_4: Enfermeiros preferem consistência em seus horários, ou seja, eles preferem trabalhar todos os dias da semana no mesmo turno (dia, noite, ou madrugada).

Recomenda-se que você comece propondo funções pra contabilizar as violações a cada restrição. Cada função deve retornar um valor de 0 até $-k$, onde k é o máximo de violações que um estado pode oferecer à restrição relacionada. Por exemplo, um estado em que um enfermeiro tem 6 turnos invés de 5, enquanto outro tem apenas 4, e os demais têm 5 cada um, deve retornar "-2" quando avaliado pela restrição r_2. Contemple diferentes funções para cada restrição e experimente com elas para compor sua função de fitness.

Dica: Você vai querer testar cada uma dessas funções separadamente em simulações independentes.

5 Experimentação e Avaliação

Após codificar as restrições do problema, você fará múltiplas execuções do algoritmo variando seus parâmetros e gerará um relatório.

Espera-se que este relatório tenha aproximadamente 3 páginas. Para compô-lo:

- Explique a sua implementação de crossover. Nesta parte, fale sobre os passos no seu procedimento, os possíveis valores que cada variável do código admite, como os estados fornecidos como entrada serão combinados e o que a função retorna para cada entrada.
- Explique a sua implementação de elitismo e de mutações. Nesta parte, fale sobre os passos do seu procedimento e como estes parâmetros afetam a execução *do seu código*.

Em seguida, você deve responder a algumas perguntas baseadas nos experimentos do roteiro abaixo. Para estas análises, você deve executar o seu código com cada combinação de parâmetros 10 vezes, totalizando 50 execuções no primeiro bloco e 60 no segundo bloco.

Bloco de Experimentação 1

Rode seu algoritmo genético com taxa de mutação em 0.1, uma população de 100 indivíduos, e 1000 iterações. Sem mudar estes parâmetros, varie apenas o elitismo para os valores 0, 0.1, 0.25, 0.5, e 0.75. Assista ou gere um log das simulações, verificando como cada uma se comporta e como variam o fitness médio e máximo de cada geração.

Responda às seguintes perguntas:

1. Como o elitismo do seu GA influenciou no número de iterações necessárias para encontrar uma solução?
2. Como o elitismo do seu GA influenciou na qualidade das soluções encontradas?

Bloco de Experimentação 2

Repita o processo acima, mas agora fixe o elitismo em 0.1 e varie apenas o tamanho da população para os tamanhos 10, 25, 50, 100, 500, e 1000.

Responda às seguintes perguntas:

1. Como o tamanho da população do seu GA influenciou no número de iterações necessárias para encontrar uma solução?
2. Como o tamanho da população do seu GA influenciou na qualidade das soluções encontradas?

Respostas curtas são suficientes, contanto que tenham todos os detalhes relevantes. As suas respostas devem mostrar que você teve atenção às simulações e que compreendeu o papel dos parâmetros do algoritmo em cada caso.

Recomenda-se que você gere gráficos com os resultados referentes a cada pergunta utilizando as médias de resultados para cada valor de entrada fixado. Fazendo isso, você poderá utilizar estes gráficos para considerar valores intermediários dos parâmetros e fazer execuções adicionais com base nestes valores para complementar sua análise. Estas execuções podem ser automatizadas.

IMPORTANTE: Seu executável deve permitir que eu repita todas suas simulações e as avalie da mesma forma que você o fez.

Se desejar, para pontuação extra, você pode fazer outras simulações e reportá-las no mesmo documento. Nesse caso, descreva as simulações que fez, destacando precisamente os valores dos parâmetros utilizados, quais as suas intenções com estas variações e os efeitos observados. Da mesma forma, a sua documentação deve apresentar todos os detalhes necessários para que eu possa repetir o experimento proposto. Seu relatório pode ser mais longo se você decidir reportar análises complementares.

6 Submissão

A submissão do trabalho se dará via Moodle. Você deve fazer upload de um arquivo .zip contendo o seguinte:

– Todos os códigos da implementação (incluindo o executável, caso haja um)

– um arquivo "**readme.txt**" ou similar indicando como rodar seu programa, como alterar os parâmetros da simulação, e, conforme a necessidade, como compilar o código, quais as bibliotecas, ferramentas e versões utilizadas, versão do sistema, etc., de acordo como a necessidade.

– um arquivo "**relatorio.pdf**" respondendo às perguntas de experimentação da Seção 4.

Este enunciado foi disponibilizado em 16/11/23 e a submissão do trabalho deve ser feita até o dia 08/12/23.