UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

Disciplina: Inteligência Artificial

Experimentação e Avaliação

Equipe:

Anderson Luís Bento Soares — 511525
Guilherme dos Santos Cavalcante — 510831
Jorge Bruno Costa Alves — 509718

Crossover:

A implementação do crossover no código segue alguns passos específicos para combinar os estados fornecidos como entrada (indivíduos da população). Analisemos o procedimento passo a passo:

1. Conversão para String:

Antes de realizar o crossover, os indivíduos (matrizes) são convertidos para strings usando a função `matrizString(individuo)`. Isso é feito para facilitar a manipulação dos bits que representam os genes do indivíduo.

2. Escolha do Ponto de Corte:

Um ponto de corte é escolhido aleatoriamente. Isso é feito pelas linhas:

```
pontoDeCorte = range(len(individuo1)
pontoDeCorte = random.choice(pontoDeCorte)
```

O ponto de corte define onde a troca de genes ocorrerá entre os dois pais.

3. Crossover:

Os genes dos dois pais são trocados a partir do ponto de corte. O código realiza a

troca da seguinte forma:

```
individuo1Backup = individuo1

individuo1 = individuo1[:pontoDeCorte] + individuo2[pontoDeCorte:]

individuo2 = individuo2[:pontoDeCorte] + individuo1Backup[pontoDeCorte:]
```

Os genes à esquerda do ponto de corte permanecem inalterados, enquanto os genes à direita são trocados entre os pais.

4. Conversão de Volta para Matriz:

Após o crossover, os indivíduos resultantes (agora representados como strings) são convertidos de volta para matrizes usando a função `StringMatriz(string, enfermeiras, turnos)`.

5. Retorno:

A função de crossover retorna os dois novos indivíduos resultantes do processo de crossover:

```
return individuo1, individuo2
```

Em resumo, o crossover neste código segue uma abordagem de ponto de corte aleatório para trocar genes entre dois pais. Isso é feito convertendo os indivíduos para strings, realizando a troca de genes, e, em seguida, convertendo os resultados de volta para matrizes antes de retornar os novos indivíduos.

Elitismo:

A implementação de elitismo no código é responsável por manter os melhores indivíduos da população atual na próxima geração, preservando assim parte da informação genética mais promissora. Aqui estão os passos relacionados ao elitismo:

1. Seleção dos Melhores Indivíduos:

Os melhores indivíduos da população atual são armazenados em uma lista chamada elitistas. Essa seleção é feita com base em uma porcentagem da população original, determinada pelo parâmetro taxaElitismo.

2. Criação da Nova População:

A nova população é inicialmente vazia. Em seguida, ela é preenchida com cruzamentos e mutações.

3. Inclusão dos Melhores Indivíduos na Nova População:

Os melhores indivíduos da população anterior (armazenados em elitistas) são adicionados à nova população.

4. Manutenção do Tamanho da População:

Para manter o tamanho da população, os piores indivíduos gerados pelos cruzamentos e mutações são excluídos.

5. Preenchimento com Cruzamentos e Mutações:

O restante da nova população é preenchido com indivíduos resultantes de cruzamentos e mutações.

Mutação:

A mutação é uma operação que visa introduzir variação genética na população, ajudando a explorar novas regiões do espaço de busca. Aqui estão os passos relacionados à mutação:

1. Conversão para String:

Antes de realizar a mutação, o indivíduo é convertido para uma representação em string usando a função `matrizString(individuo)`.

2. Escolha Aleatória de um Gene para Mutação:

Um gene é escolhido aleatoriamente para ser mutado. Isso é feito selecionando uma posição aleatória na string.

3. Mutação do Gene Escolhido:

O gene escolhido é invertido, ou seja, se for '0', torna-se '1', e vice-versa.

4. Conversão de Volta para Matriz:

Após a mutação, a string modificada é convertida de volta para uma matriz usando a função `StringMatriz(string, enfermeiras, turnos)`.

5. Retorno:

A função de mutação retorna o indivíduo modificado.

Bloco de Experimentação 1:

1. À medida que aumentamos a taxa de elitismo, observamos um aumento no número de iterações necessárias para o algoritmo alcançar um resultado significativo.
2. Os resultados obtidos com taxas de elitismo mais altas não superam aqueles associados a taxas de elitismo mais baixas.

Bloco de Experimentação 2:

1. À medida que reduzimos o tamanho da população, observamos um aumento no número de iterações necessárias para o algoritmo atingir resultados significativos.
2. Resultados superiores são alcançados com populações maiores em comparação com aquelas de tamanho reduzido, resultados com as populações maiores também apresentam um número menor de iterações.