MC906 A - Sudoku

Gustavo Libraiz - 198537
Lucas Rodrigues - 182557
Matheus Mazon - 203609
Vinícius Balbino de Souza - 178183

Introdução

Sudoku é um jogo baseado na colocação lógica de números. O objetivo do jogo é a colocação de números de 1 a 9 em cada uma das células vazias numa grade de 9x9, constituída por 3x3 subgrades chamadas regiões. O quebra-cabeça contém algumas pistas iniciais, que são números inseridos em algumas células, de maneira a permitir uma indução ou dedução dos números em células que estejam vazias. Cada coluna, linha e região só pode ter um número de cada um dos 1 a 9. Resolver o problema requer apenas raciocínio lógico. Existem diversas variações e regras alternativas para resolução desse tipo de *puzzle*, porém todas as instâncias apresentadas são soluções do modo clássico do sudoku.

Metodologia

Após utilizar o pré-processamento [2] para facilitar o problema inicial e reduzir o número de gerações necessárias para resolver uma instância, realizamos a utilização do código genético, definindo assim: limite da fitness - definido como a soma de elementos distintos em linhas, colunas e blocos, gerando assim o valor máximo de 243 -, tamanho da população inicial, limite de gerações, probabilidade de mutação e de crossover e métodos de seleção e substituição.

- **População Inicial**: Pode preencher os espaços vazios do sudoku de maneira aleatória - com random do python -, com uma permutação da lista $[1, 2, 3, 4, 5, 6, 7, 8, 9]$, ou com uma permutação da linha, que preenche os espaços vazios respeitando as limitações impostas.
- **Seleção**: Do tipo torneio ou roleta. No torneio, k indivíduos são selecionados de maneira aleatória e é selecionado o indivíduo com melhor fitness. O processo é realizado duas vezes para que possa ser feito o crossover entre esses dois individuos. Na roleta, individuos são selecionados de maneira que sua probabilidade de ser escolhido é diretamente proporcional a sua fitness.
- **Crossover**: Foram implementados três métodos, o *one point* realiza um crop em uma posição c de um individuo e cola nos caracteres restantes do outro individuo. O *vertical one point* realiza uma ideia semelhante, porém preservando as linhas de cada puzzle inteiramente, alterando apenas colunas. Já o *vertical two point* é uma implementação aditiva das duas implementações anteriores, visto que realiza a mesma operação que a função vertical one point, mas utiliza dois pontos no array para isso [1].
- **Mutação**: Novamente implementados três métodos, *flip*, que escolhe uma posição aleatória do sudoku e substitui o número colocado por outro aleatório, *swap*, que escolhe duas posições aleatórias do individuo e realiza a troca, e o *horizontal swap*, que realiza o mesmo que o swap, mas para indivíduos na mesma linha.
- **População**: Pode ser crescente ou não, mas sempre elitista, isso é, substituindo os novos elementos a partir da retirada dos elementos com pior fitness

Resultados

Comparando todos os pontos apresentados acima, chegamos aos seguintes resultados (para verificação e justificativa dos resultados, olhar notebook): para o **crossover**, a técnica *two points* é superior a *one-point*, para a **mutação**, é importante definirmos uma taxa de **mutação** alta para geração de diversidade, e assim verificamos que o *horizontal swap* é a melhor heurística a ser escolhida, por não quebrar as regras de sudoku na mutação, reduzindo a geração de filhos piores que os pais. Observando-se os resultados e durante o desenvolvimento, pode-se observar que o *método da roleta* tende a cair mais em máximo locais, isso porque como o *torrneio* escolhe seus participante de forma aleatória, ele tende a criar uma população mais diversidada. A configuração 8 - que faz swap entre partes aleatórias do sudoku, sem preservar linhas - (verificar notebook) se mostrou a pior configuração, tanto em número de gerações, quanto no tempo gasto. Como esperado, as configurações que preservam linhas foram as mais eficientes. Isto, é crossover vertical, swap horizontal e forçando que cada linha seja uma permutação de $range(1, 10)$. Dessa forma, as linhas são preservadas e a função fitness fica mais barata, pois toda linha é consistente. Em especial, a melhor configuração foi a segunda, que funciona com crossover vertical de dois pontos, $mut - rate = 0.5$, método de seleção de torneio e uma população inicial de tamanho 1000 indivíduos. Em média, essa configuração resolveu os sudokus em 5.7 segundos e com 32 gerações.

Referências

[1] DWDyer. Sudoku watchmaker framework.

[2] Norvig. Solving every sudoku puzzle.