Insper, 2024-2

Supercomputação

Atividade extra do algoritmo genético

Jerônimo de Abreu Afrange

https://github.com/jeronimo-a/algoritmo-genetico-mochila-binaria-supercomp-2024-1

Resumo da atividade

A atividade foi realizar uma implementação de algoritmo genético para o problema da mochila binária em C++. Após a implementação do algoritmo, disponível no *link* do *github* acima, foram realizados testes com o objetivo de compreender como diferentes parâmetros do algoritmo influenciam a qualidade dos resultados obtidos.

- N_PARENTS: número de soluções que são passadas para a próxima geração a cada iteração;

- MUTATION_RATE: chance de ocorrer uma única mutação em um gene aleatório para cada nova solução gerada em uma iteração;

- CROSSOVER_RATE: chance de ocorrer *crossover* entre cada par subsequente de soluções.

Para cada teste apenas o parâmetro em questão foi variado, enquanto os demais foram mantidos no meio do intervalo de valores possíveis, que é 50% para as taxas de mutação e *crossover*, e, neste caso, 16 para a quantidade de indivíduos sobreviventes. Cada variação de cada parâmetro, nos gráficos da página seguinte, foi testada com 1000 *seeds* diferentes, e o resultado reduzido para cada valor foi a média dos *fitness* finais obtidos para cada *seed*.

Resultados

Para os parâmetros de quantidade de soluções sobreviventes por geração e taxa de mutação, observou-se uma diferença do valor máximo obtido em relação ao valor mínimo de aproximadamente 25%, sendo que para o primeiro, existe um ponto ótimo, enquanto para o segundo, quanto maior a taxa, melhor. Quanto à taxa de *crossover*, a influência sob as dadas circunstâncias foi muito menor, apresentando uma diferença de relativa de

apenas 1%. O que faz sentido, pensando que a taxa de crossover, nesta implementação, serve apenas como uma aleatorização de quais soluções sobreviventes produzirão uma nova solução, considerando que sempre ocorre a mesma quantidade de *crossovers*.

Possíveis melhorias na análise seriam: fazer uma análise multivariada (descobrir o *fitness* ótimo em função de mais de uma variável ao mesmo tempo), analisar também outros parâmetros (como a quantidade de gerações) e gerar a solução ótima a partir de algoritmos de força bruta para ser usada como *benchmark*.