Universidade Federal de Uberlândia
Faculdade de Computação
Curso de Graduação em Ciência da Computação

COMPUTAÇÃO BIO-INSPIRADA

DANIEL GONÇALVES (12011BCC011)

Relatório - Trabalho 1: Implementação de um algoritmo genético para o problema da mochila e comparação com outras abordagens

UBERLÂNDIA

2023

1 SUMÁRIO

2 INTRODUÇÃO

O objetivo deste trabalho é explorar diferentes implementações de um algoritmo genético para o problema da mochila binária, e em seguida, comparar seus resultados com implementações baseadas em Programação Dinâmica e GRASP (Greedy Randomized Adaptive Search Procedure).

Duas implementações do algoritmo genético foram exploradas neste trabalho, com a única diferença entre elas sendo o operador de crossover. Os operadores considerados foram de Crossover de 1 pontos e o de Crossover de 2 pontos. O elitismo não foi considerado neste trabalho. Por fim, o mecanismo de seleção utilizado foi o de torneiro, considerado quatro indivíduos em cada rodada.

3 DESENVOLVIMENTO

A implementação do algoritmo genético foi feita na linguagem Python, com uso de pacotes disponíveis nativamente na própria linguagem. Todo o código está contido em um único arquivo (main.py), com as funções um() e dois() correspondendo ao Operador de Crossover de 1 ponto e de Crossover de dois pontos, respectivamente.

3.1 Dados de Entrada

Os dados de entrada para o programa seguem um padrão fixo, na ordem indicada a seguir:

1. Quantidade de objetos disponíveis (representada por N).
2. Peso Máximo da mochila.
3. N linhas, cada uma contendo o valor e o peso de um objeto, respectivamente.
4. O valor ótimo, sendo um parâmetro opcional.

O programa aceita esses dados por meio da entrada padrão, logo os dados armazenados em um arquivo devem ser redirecionados via linha de comando.

3.2 Taxas de Crossover, Mutação e Tamanho da População

De forma global a taxa de crossover utilizada foi de 70% e a de mutação foi de 1%, ambos valores muito utilizados na literatura.

Já o tamanho da população foi definido como quatro vezes a quantidade de objetos disponíveis, após testes com um exemplo básico (arquivo 1.txt).

3.3 Cálculo da Aptidão

A aptidão de um cromossomo é simplesmente a soma dos valores dos objetos adicionados na mochila. Caso o peso desses objetos exceda a capacidade da mochila, a aptidão é definida como –1.

3.4 Geração da População Inicial

A geração da população inicial é um passo fundamental para que o algoritmo funcione corretamente. À princípio, cromossomos cuja aptidão fosse –1 eram adicionados normalmente à população inicial. Porém, isso acabou por envenenar o algoritmo, de modo que ele não conseguia evoluir pela alta presença desses cromossomos.

Para contornar esse problema, foi incluída uma função responsável pela correção desses cromossomos na população inicial. Um número é sorteado, e a partir daquele alelo objetos vão sendo retirados da mochila até que o peso total seja menor ou igual ao limite, tornando o indivíduo válido.

Esse mecanismo é apenas usado na população inicial, não sendo usado no restante do algoritmo.

3.5 Mecanismo de Seleção

O mecanismo de Torneio foi adotado para a seleção. Na presente implementação, quatro indivíduos são escolhidos aleatoriamente dentro da população e o que tiver a melhor aptidão é

escolhido para fazer parte da população intermediária, que posteriormente pode ou não realizar crossover.

3.6 Mecanismos de Crossover

3.6.1 Crossover de 1 ponto

No crossover de 1 ponto é sorteado um número entre 0 e N-2, onde N é a quantidade de bits do cromossomo. A razão de ser N-2 é o fato de que estamos considerando apenas os locais onde cortes são possíveis, e não a quantidade de alelos existentes.

O crossover pode acontecer com uma chance de 70%. Caso não ocorra, os dois pais são adicionados à próxima população. Caso ocorra, os pais geram dois filhos, que por sua vez sofrem mutação e são adicionados à próxima população.

3.6.2 Crossover de 2 pontos

A única diferença com o outro mecanismo de crossover é o fato de que dois números são sorteados entre 0 e N-2, sendo que eles não podem ser iguais. A região entre o menor e o maior é trocada entre os pais, resultando em dois filhos, que por sua vez sofrem mutação e são adicionados à próxima população.

3.7 Mecanismo de Mutação

Para o mecanismo da mutação foi utilizada a abordagem em que existe uma chance de 1% para que ocorra uma mutação em cada gene de um dado cromossomo.

3.8 Critérios de Parada

Dois critérios de parada foram definidos:

1- Caso o ótimo tenha sido passado como parâmetro, o programa irá parar assim que o primeiro cromossomo com esse valor seja gerado.
2- Caso contrário, e quando o programa não for capaz de obter nenhum cromossomo com o ótimo passado, o algoritmo irá parar quando 80% dos cromossomos apresentarem a mesma aptidão.

4 RESULTADOS

Para os testes considerando as quatro implementações (PD, GRASP, AG Crossover 1-ponto, AG Crossover 2-pontos) foram selecionadas 4 instâncias do problema da mochila disponíveis junto com as implementações do problema em PD e GRASP. Em específico, foram os conjuntos de dados input1.in à input4.in.

O conjunto input1 apresenta 20 objetos, o input2 apresenta 40 objetos e os inputs 3 e 4 apresentam 50 objetos cada.

Para cada uma das abordagens foram realizadas 10 execuções para cada dos conjuntos de dados. Em seguida, as médias e desvios padrões dos ótimos obtidos e do tempo necessário para a execução foram calculados. A seguir discutirei alguns dos resultados observados:

4.1 Média e Desvio-Padrão dos Ótimos

Como pode ser observado no Gráfico 1, a Programação Dinâmica (PD) foi a abordagem cujas médias mais se aproximaram do ótimo global, seguida por perto por ambas as versões do algoritmo genético. Por último, o GRASP apresentou as piores médias entre os algoritmos considerados.

Além disso, pela Tabela 1, temos que com exceção do conjunto de dados 2, o algoritmo genético com crossover de 2 pontos apresenta melhores resultados que o de apenas um ponto.

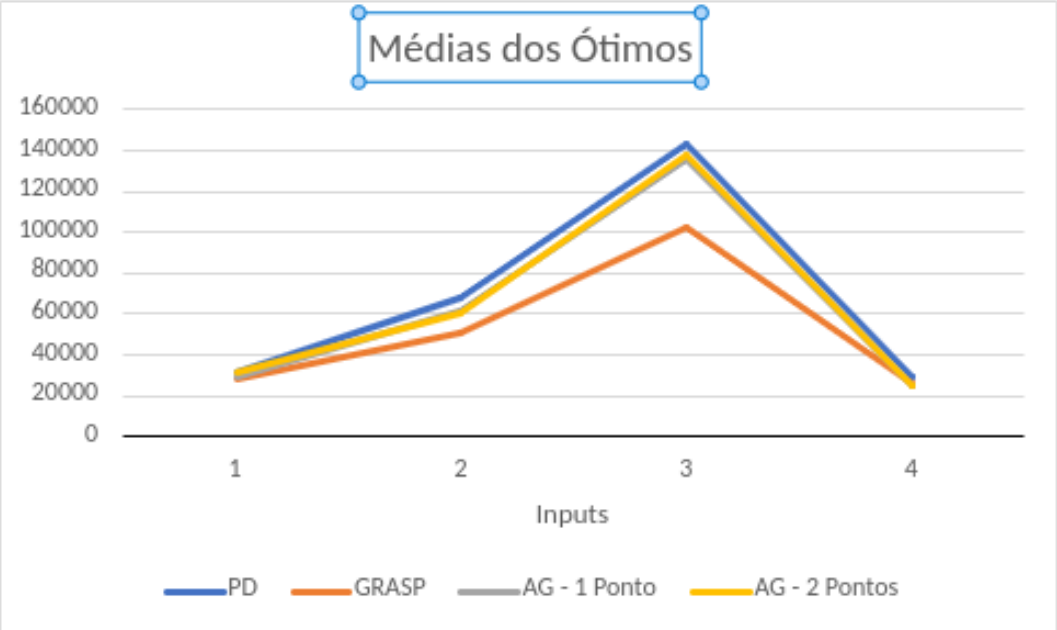

Gráfico 1 - média dos ótimos obtidos por cada um dos algoritmos considerando cada um dos conjuntos de entrada.

Média dos Ótimos				
Algoritmo	**Input1**	**Input2**	**Input3**	**Input4**
Dynamic	31621	67829	143449	28840
Grasp	27939,1	50816,3	102641,9	26570,7
AG - 1 Ponto Crossover	28912,1	62104,6	135884,8	24973,9
AG - 2 Ponto Crossover	31101,6	60358	138163,3	25056,2

Tabela 1 - média dos ótimos obtidos por cada um dos algoritmos.

Agora, se analisarmos os desvios-padrões dos ótimos obtidos, temos curiosamente que para todos os conjuntos de dados, a PD tem desvio-padrão de 0, como visto na Tabela 2. Isso indica que a abordagem baseada em PD alcançou o ótimo global em 100% das ocasiões. Já os outros algoritmos apresentam diferentes graus de desvio em relação a suas médias.

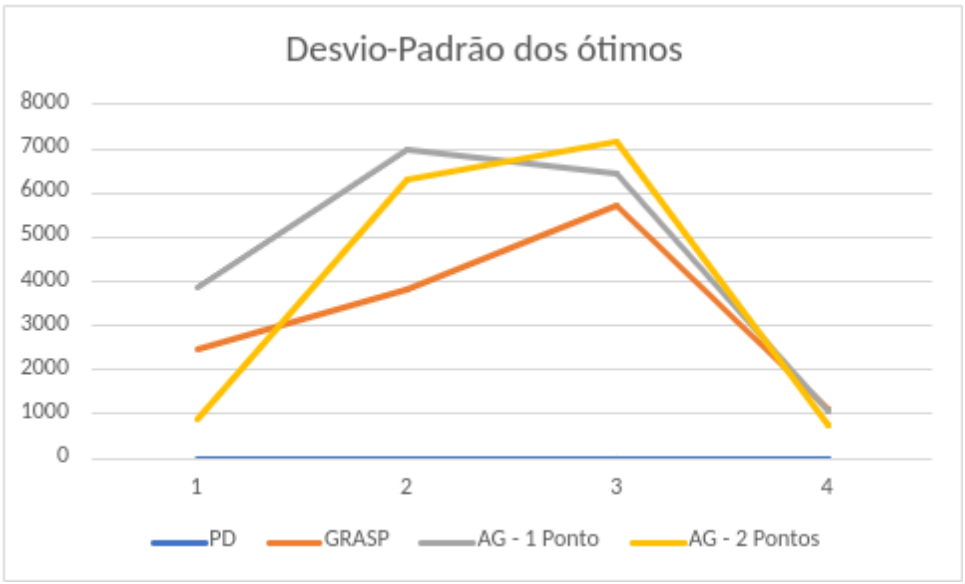

Gráfico 2 – desvio=padrãp dos ótimos obtidos por cada um dos algoritmos considerando cada um dos conjuntos de entrada.

Desvio Padrão dos Ótimos				
Algoritmo	**Input1**	**Input2**	**Input3**	**Input4**
Dynamic	0	0	0	0
Grasp	2451,751956	3830,385013	5728,462901	1128,612427
AG - 1 Ponto Crossover	3860,914441	6976,836408	6449,428619	1081,300703
AG - 2 Ponto Crossover	886,4660425	6323,22163	7174,22304	751,1394012

Tabela 2 – desvios-padrões dos ótimos obtidos por cada um dos algoritmos.

4.2 Média e Desvio Padrão dos Tempos

Como pode ser observado no Gráfico 3 e na Tabela 3, quanto mais objetos uma instância do problema da mochila tem, maior tende a ser a o tempo necessário para que o algoritmo seja executado.
Em particular, o algoritmo GRASP apresentou as menores médias, o que acaba não sendo muito vantajoso por ter sido o algoritmo com as piores médias. Em seguida, o melhor algoritmo em tempo de execução foi o PD, o que considerando sua precisão para achar o ótimo global indica um ótimo custo-benefício.
No que tange os algoritmos genéticos considerados, seu desempenho foi um pouco inferior aos outros dois até a base de dados com 40 objetos, mas que aumentaram muito quando a base chegou à 50 objetos. Com isso, seu uso, da maneira como estão, só seria viável se o tempo não fosse um problema.

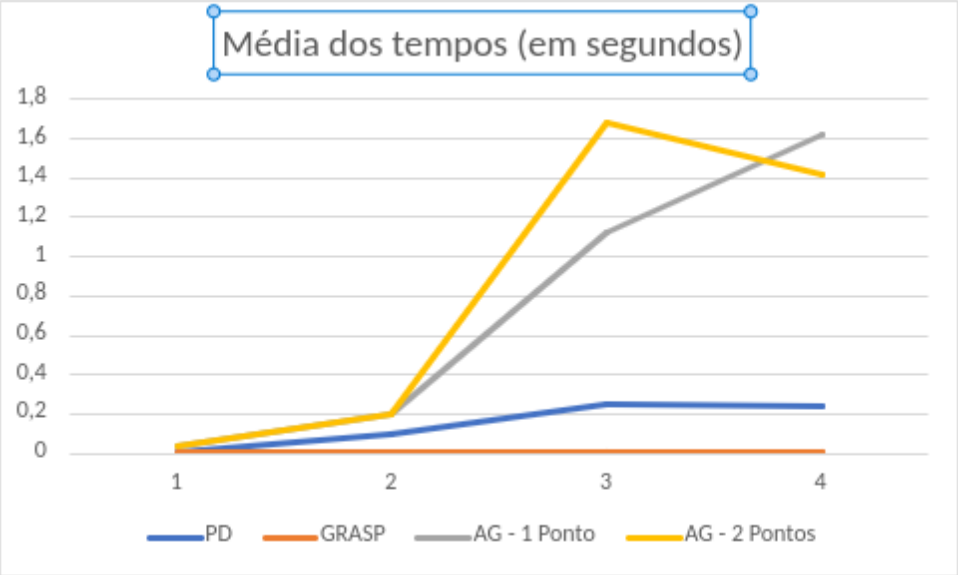

Gráfico 3 – média dos tempos das execuções de cada algoritmo considerando os conjuntos de dados.

Média dos Tempos				
Algoritmo	**Input1**	**Input2**	**Input3**	**Input4**
Dynamic	0,0033841	0,101316071	0,247328138	0,235338259
Grasp	0,000791693	0,001397824	0,001694489	0,003050256
AG - 1 Ponto Crossover	0,034760809	0,20354836	1,124678588	1,618087983
AG - 2 Ponto Crossover	0,037771344	0,199394655	1,68001883	1,415991712

Tabela 3 – médias dos tempos de exeeução de cada algoritmo.

Por fim, no que tange o desvio-padrão dos tempos, pelo Gráfico 4 e tabela 4, temos que o PD e o GRASP apresentam tempos mais consistentes, enquanto os dois algoritmos genéticos apresentam tempos mais voláteis, causados provavelmente por sua natureza não-determinística e aleatória.

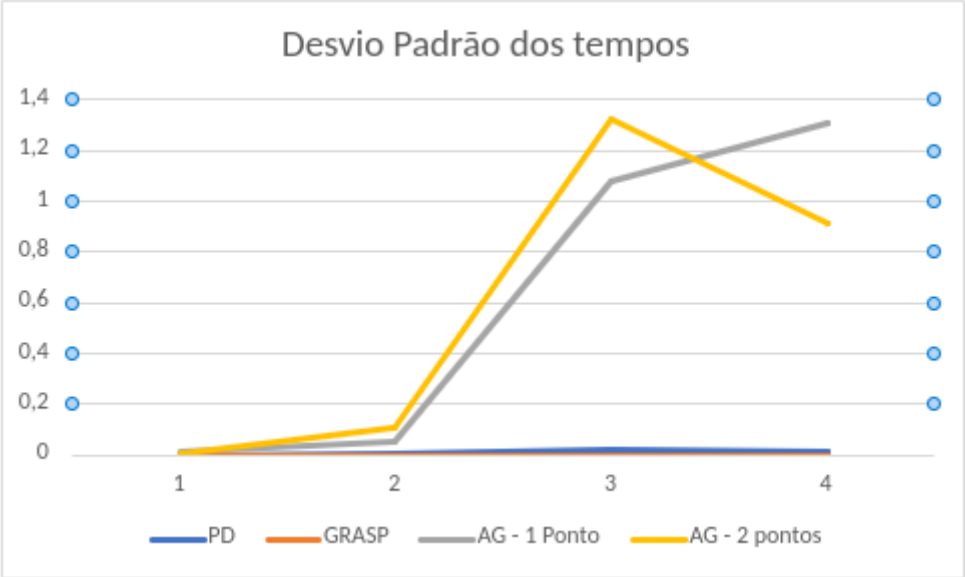

Gráfico 4 – desvio-padrão dos tempos das execuções de cada algoritmo considerando os conjuntos de dados.

Desvio Padrão dos Tempos				
Algoritmo	**Input1**	**Input2**	**Input3**	**Input4**
Dynamic	0,000334286	0,006586124	0,022516564	0,009193624
Grasp	0,000044	0,000057	0,000140098	0,000183527
AG - 1 Ponto Crossover	0,012836485	0,048013854	1,080695755	1,308693876
AG - 2 Ponto Crossover	0,007566032	0,105317341	1,327212668	0,914262564

Tabela 4 – desvio-padrão dos tempos de exeeução de cada algoritmo.

5 CONCLUSÃO

A abordagem utilizando PD acabou dominando o cenário tanto no quesito de encontrar o ótimo global sempre, em contraste com os outros algoritmos que nem sempre o encontram, quanto no quesito tempo. Apesar de ser menos eficiente que o algoritmo GRASP, o algoritmo PD apresenta um melhor custo-benefício.

Considerando o baixo desempenho em encontrar valores mais próximos do ótimo global em relação aos outros algoritmos, o GRASP apresenta uma ótima oportunidade de uso quando o tempo é limitado e a resposta esperada não precisa ser muito boa, mas suficientemente eficiente.

Já os algoritmos genéticos implementados neste trabalho precisam ser refinados para que possam competir em tempo de execução e eficiência em encontrar o melhor global quando colocados lado a lado com outros algoritmos. Em particular, a inclusão do mecanismo de elitismo e outras abordagens para seleção, crossover e mutação podem aumentar o custo-benefício das implementações feitas.