# Tarefa 01 - AG - Problema da Mochila

Aluno: Cesar Rafael Lopes

Repositório GitHub: ttps://github.com/ceslop84/ia\_ag

1. Explique como procedeu para penalizar o fitness de mochilas infactíveis (mochilas cujo peso é maior do que o máximo permitido.

a. Adicione o pseudo-código do método implementado.

b. Explique o método em linguagem natural.

O método de penalização implementado emprega uma proporção simples entre o peso atual da mochila, o fitness atual e qual seria o fitness estimado para a capacidade máxima da mochila. Por exemplo, caso a mochila tivesse um fitness de 245 mas um peso de 205 seria realizada a seguinte proporção, através de uma regra de três bastante simples:

O que resultaria nos seguintes valores:

Portanto, o fitness da mochila em questão seria penalizado em 102 pontos, indo de 245 para 143.

Cabe ressaltar que o seu peso e composição se manteriam inalterados após a penalização, apenas o seu valor de fitness é que sofreria atualização.

O psedo-código para este cálculo seria:

*se mochila.peso > capacidade então*

*mochila.fitness = (capacidade \* mochila.fitness) / mochila.peso*

2. Explique como procedeu para reparar mochilas infactíveis (cujo peso é maior do que o máximo permitido), ou seja, para que tivessem seu peso ajustado para atender à restrição de capacidade.

a. Adicione o pseudo-código do método.

b. Explique o método em linguagem natural.

O método de reparação implementado implementa uma repetição para remoção aleatória de itens de mochila até que o seu peso respeito a capacidade máxima pré-determinada. Por exemplo, para uma mochila de peso 132 e fitness 180, o método de reparação iria escolher aleatoriamente um item para ser removida da mochila (alterar o valor da posição de 1 para 0). Supondo que nesse exemplo ele escolhesse aleatoriamente a posição 39, o peso de 15 kg seria removido bem como o valor de 15 de seu fitness. Portanto, resultaria num novo peso de 117 e fitness de 165. Nesse caso não seria necessária nova iteração de remoção de itens aleatória, pois o peso da mochila já estaria respeitando o critério de carga máxima.

Contudo, supondo que item escolhido, de número 39, já não estivesse na lista de presença da mochila, nesse caso, o valor da composição da mochila para este item seria alterado de 0 para 0, o que na prática não representa alteração alguma. Devido a este fato, da remoção do item aleatório não ter surtido efeito no peso da mochila, uma nova iteração seria necessária para buscar a adequação da mesma ao critério de carga.

O psedo-código para este cálculo seria:

*enquanto mochila.peso > capacidade então*

*identificador\_item = seleção aleatória de número entre 0 e 41*

*mochila.remover\_item(identificador\_item)*

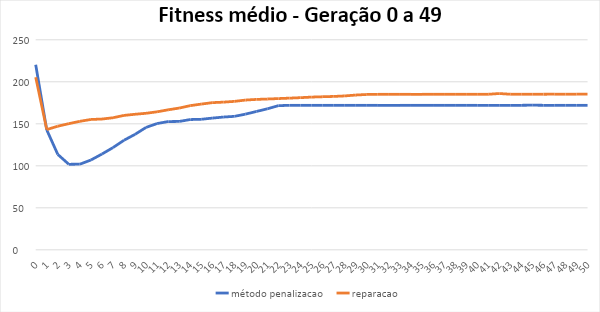
*mochila.calcular\_peso()*

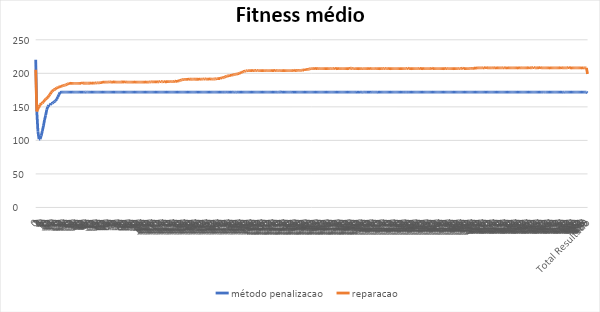
*fim enquanto*

*mochila.calcular\_fitness()*

3. Plote um gráfico valor do fitness x geração da execução na qual obteve o melhor fitness, portanto, o gráfico deve ter 2 curvas (uma para cada implementação). Responda: as curvas variam em função do modo de cálculo de fitness: penalização x reparação? Explique.

Empregando o fitness médio das execuções que obtiveram o melhor fitness para cada método, foram gerados 2 gráfico. O primeiro ilustra a evolução do fitness médio para as gerações 0 até 49, e um segundo com toda a evolução, da geração 0 até a 500.





Observa-se que até à geração 130 ambas os métodos apresentaram comportamento similar, ainda que o método de reparação já se destacasse com valores de fitness médios superiores ao do método de penalização. Contudo percebe-se que os valores de fitness médios do método de penalização se estabilizaram em torno do valor 175 a partir da geração 25 até o final dos cálculos na geração 500. Por outro lado o método de reparação apresentou consistentes avanços no valor médio de fitness, saindo de um patamar de 175 para atingir valores médio em torno de 210 ao final dos cálculos.

Considerando que o método de penalização não altera os indivíduos (a sua composição e peso permanecem inalterados), há a possibilidade de que não estejam sendo selecionados os melhores elementos para realizarem a reprodução ao longo das gerações. Ou seja, mesmo que apresentem fitness em valores relativamente altos, na sua composição de itens, estes elementos ainda não estão adequados à restrições de carga impostas para a simulação. Ou seja, a cada geração a aplicação do método de penalização parece surtir efeitos muito sutis, uma vez que os elementos mais adequados para atender o critério de carga poderiam não estar sendo selecionados e assim, a cada geração há uma nova seleção equivocada de elementos para reprodução. Considerando que o método de penalização aplicou uma regra de proporção para reduzir o valor de fitness dos elementos não enquadrados como factíveis, pondera-se se talvez fosse necessário uma aplicação de uma penalização mais severa, visando surtir melhores resultados para o método.

Por outro lado, o método de reparação, mesmo que aplicando um método aleatório para reparar cada elemento que não respeitasse o critério máximo de carga, parece ter demonstrado uma eficiência e eficácia mais acentuada. Mais eficiente pois são notáveis os momentos onde houveram saltos no desempenho do fitness médio entre poucas gerações de distância. E também eficaz pois parece que o método resultado numa convergência para o objetivo proposto, obter a mochila mais valiosa para a capacidade de carga proposta.

Portanto, após breve análise denota-se que o método de reparação proposto obteve melhor desempenho na simulação por atuar mais diretamente na raiz do problema, que seria a composição da mochila de fato. O método da penalização, todavia, focou apenas na consequência de uma mochila inadequada, que seria o peso acima da capacidade, sem buscar alterar a composição da mesma. Ou seja, a manipulação simples do indicador de fitness não foi eficaz em obter resultados satisfatórios para a simulação, uma vez que seu melhor resultado é 17,3% inferior ao melhor resultado do método de reparação (208 contra 172). Adicionalmente, cabe destacar que até mesmo o pior resultado com o método da reparação é ainda melhor que o melhor resultado do método de penalização (187 contra 172).

4. Sobre as melhores soluções obtidas, responda para cada uma das implementações:

a. Qual foi o valor máximo para os itens de uma mochila que você encontrou (sem violar a capacidade em Kg da mochila)?

b. Quantas mochilas com valor máximo foram obtidas?

c. Liste todas as mochilas que obteve que apresentaram valor máximo. Para cada uma delas coloque os itens, valor total e peso total.

Para o método de penalização foi obtida uma mochila com um fitness (valor) de 172 com 15 itens e peso total de 120 kg. Para o método de reparação foi obtida uma mochila com um fitness (valor) de 208 com 13 itens e peso total de 120. Para cada método foi obtida uma única mochila com o valor de maior fitness.

Segue em anexo tabela com as listas das mochilas obtidas para as 10 execuções de cada método.

5. Compare a taxa de sucesso das implementações penalização e reparação. Para este problema, taxa de sucesso é o número de vezes que a solução de maior valor (possivelmente a ótima) foi encontrada nas execuções realizadas (recorda-se que o total de execuções foi definido no item a do método). Responda:

a. Quais foram as taxas de sucesso obtidas?

b. Quantas vezes o cálculo de fitness é executado para a configuração em questão por execução? Escreva a fórmula.

c. Qual método implementado é mais custoso temporalmente: o de reparação ou de penalização?

O método mais custos é o método de reparação, considerando a sua natureza aleatória e de tentativa e erro. Muitas vezes o método tem que ser realizado mais de uma vez para o mesmo elemento, pois na primeira tentativa a carga da mochila ainda estaria acima da capacidade máxima pré-determinada.

O método de penalização possui aplicação mais objetiva, sendo realizado apenas um teste para verificar se a carga da mochila em análise respeita o critério de carga e caso contrário é realizada a operação matemática para aplicar a proporcionalidade sobre seu valor de fitness. Ou seja, em nenhum momento este cálculo é realizado mais de uma vez sobre o mesmo elemento.