Algoritmo Genético - Problema da Mochila

André Minoro Fusioka

1 Penalização

Para a penalização de indivíduos infactíveis, foi adotada a seguinte métrica:

$$fitness\_ajustado(x_n) = \begin{cases} fitness(x_n) & \text{se } peso(x_n) \leq C \\ \frac{1}{peso(x_n)} & \text{se } peso(x_n) > C \end{cases}$$

onde $fitness\_ajustado$ é o valor do fitness após a penalização, os valores mais altos são selecionados para a próxima geração. O $peso(x_n)$ é a soma do peso de todos os itens na mochila do n-ésimo indivíduo. Desse modo, quanto maior o peso da mochila de um individuo, menor o fitness e por consequência são menores as chances de ser selecionado para a próxima geração.

O código abaixo ilustra o cálculo:

Algorithm 1 Fitness Ajustada

```
1:  function FitnessAjustada(população)
2:      fitness_ajustado ← []
3:      loop ∀ individuo da populao:
4:          peso ← calcular_peso(individuo)
5:          if peso < C then
6:              fitness ← calcular_fitness(individuo)
7:          else
8:              fitness ← 1/peso
9:          adicione o fitness à lista fitness_ajustado
        end loop
10:     return fitness_ajustado
```

Cada indivíduo tem o seu peso calculado, se estiver abaixo do limite o fitness real é adicionado à lista de fitness ajustado (sem realizar operação nenhuma), caso esteja acima do limite o fitness penalizado será calculado conforme descrito e então adicionado à lista de fintess ajustado.

2 Reparação

Para a reparação o peso de cada mochila de cada indivíduo é avaliado e enquanto for maior que o limite um item aleatório é retirado. Quando o peso estiver dentro do limite o fitness dessa mochila é avaliada e adicionada a lista de fintess reparada. O pseudocódigo é exibido abaixo:

Algorithm 2 Fitness Reparado

```
1: function FitnessReparado(população)
2:     fitness_reparado ← []
3:     loop ∀ individuo da populao:
4:         peso ← calcular_peso(individuo)
5:         loop enquanto peso > C
6:             individuo ← remove_item_aleatorio(individuo)
7:             peso ← calcular_peso(individuo)
           end loop
8:         adicione o fitness à lista fitness_reparado
       end loop
9:     return fitness_reparado
```

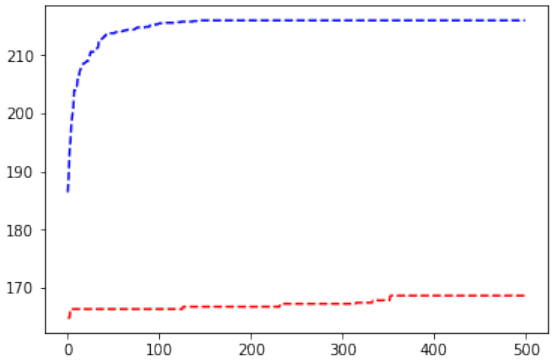
Dessa forma, caso os itens da mochila do indivíduo estejam dentro do limite permitido seu fitness é adicionado à lista de fintess reparados (sem realizar operação nenhuma), caso esteja acima do limite um item aleatório é removido da mochila até estar dentro do limite, quando estiver dentro do limite seu fitness é calculado e adicionado à solução.

3 Penalização x Reparação

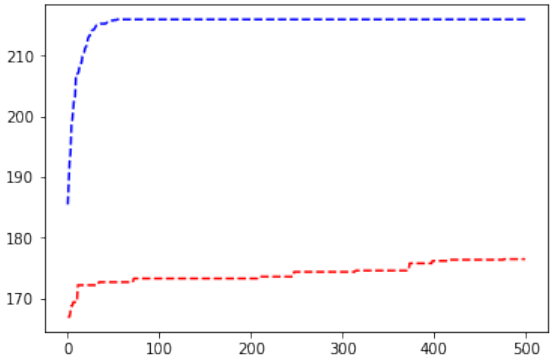
A figura 1a exibe um exemplo da comparação das abordagens de desenvolvimento em que o tamanho da população ($Np = 500$) é igual ao número de gerações, com uma probabilidade de crossover de 90% e 10% de mutação. A curva vermelha represanta a penalização e a azul a reparação.

A figura 1b exibe um exemplo da comparação das abordagens de desenvolvimento em que o tamanho da população ($Np = 500$) é igual ao número de gerações, com uma probabilidade de crossover de 80% e 5% de mutação.

A figura 2 exibe um exemplo da comparação das abordagens de desenvolvimento em que o tamanho da população ($Np = 100$) é **inferior** ao número de gerações ($Ng = 500$). A figura 2a representa uma configuração com $Pc = 90\%$ e $Pm = 10\%$ e a figura 2b ilustra $Pc = 80\%$ e $Pm = 5\%$ de mutação.



(a) Np=500 Pc=90% Pm=10%



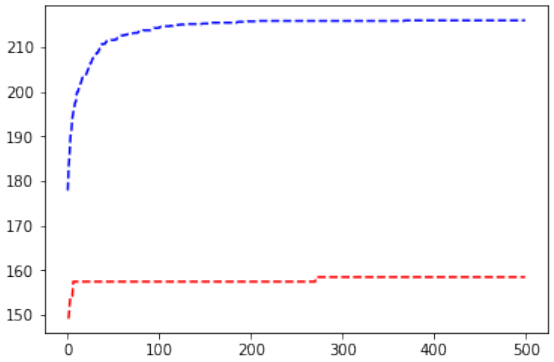
(b) Np=500 Pc=80% Pm=5%

Figura 1: Gráficos de comparação entre as abordagens de penalização e reparação com população **igual** ao número de gerações

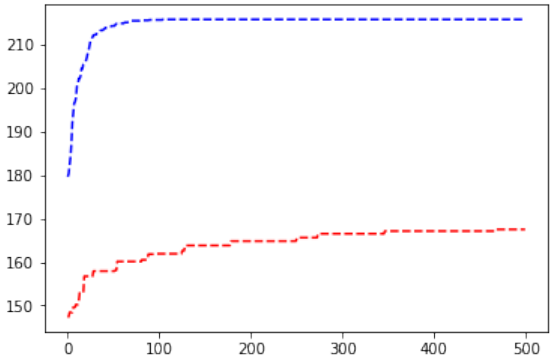
Em todos os casos o comportamento a reparação se mostrou superior, enquanto subindo rápidamente para um valor ótimo, enquanto a penalização teve um crescimento mais lento, com uma melhora lenta e inferior à reparação.

Para a reparação nota-se um crescimento rápido nas primeiras gerações e então uma estabilização na solução ótima, enquanto para a penalização possuí um leve crescimento durante as gerações com alguns crescimentos mais abruptos dando o comportamento de "degrau"à curva.

A vantagem da reparação é não permitir que indivíduos não factíveis passem para as próximas gerações, sempre havendo uma melhora (ou estabilização) de uma geração para a outra. Enquanto que o uso penalização pode haver indivíduos não factíveis, principalmente nas primeiras gerações.



(a) Np=100 Ng=500 Pc=90% Pm=10%



(b) Np=100 Ng=500 Pc=80% Pm=5%

Figura 2: Gráficos de comparação entre as abordagens de penalização e reparação com população **menor** que o número de gerações

Então, caso haja uma inicialização ruim (com muitos indivíduos infactíveis) pode haver uma demora maior para a melhora das soluções.

É interessante notar que com uma probabilidade de crossover de 80% e probabilidade de mutação de 5% a penalização teve uma leve melhora em relação à sua configuração com $Pc = 90\%$ e $Pm = 10\%$, porém ainda ficando abaixo da reparação.

4 Soluções encontradas

Abordagem: Penalização

Tamanho População (Np): 500

Máximo gerações: 500

P. Crossover (Pc): 90%

P. Mutação (Pm): 10%

Executando o algoritmo dez vezes obteve-se um total de 496 mochilas com valor igual a melhor solução (contando todas as execuções). A primeira solução foi encontrada na 3ª geração de 500.

Exemplos de mochila (demais exemplos omitidos devido ao tamanho da saída):

```
 < 19, 118.000000, 179.000000, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 1, 1 >
< 19, 118.000000, 179.000000, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 1, 1 >
< 19, 118.000000, 179.000000, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
1, 0, 0, 0, 1, 0, 0, 1, 1 >
```

Abordagem: Reparação

Tamanho População (Np): 500

Máximo gerações: 500

P. Crossover (Pc): 90%

P. Mutação (Pm): 10%

Executando o algoritmo dez vezes obteve-se um total de 3936 mochilas com valor igual a melhor solução (contando todas as execuções). A primeira solução foi encontrada na geração 146ª de 500.

Exemplos de mochila (demais exemplos omitidos devido ao tamanho da saída):

```
 < 14, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
< 14, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
< 14, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
```

Abordagem: Penalização

Tamanho População (Np): 500

Máximo gerações: 500

P. Crossover (Pc): 80%

P. Mutação (Pm): 50%

Executando o algoritmo dez vezes obteve-se um total de 499 mochilas com valor igual a melhor solução (contando todas as execuções). A primeira solução foi encontrada na geração 2ª de 500.

Exemplos de mochila (demais exemplos omitidos devido ao tamanho da saída):

```
 < 14, 114.000000, 186.000000, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0,
1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
1, 1, 0, 0, 1, 0, 0, 0, 1 >
< 14, 114.000000, 186.000000, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0,
1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
1, 1, 0, 0, 1, 0, 0, 0, 1 >
< 14, 114.000000, 186.000000, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0,
1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
1, 1, 0, 0, 1, 0, 0, 0, 1 >
```

Abordagem: Reparação

Tamanho População (Np): 500

Máximo gerações: 500

P. Crossover (Pc): 80%

P. Mutação (Pm): 50%

Executando o algoritmo dez vezes obteve-se um total de 4584 mochilas com valor igual a melhor solução (contando todas as execuções). A primeira solução foi encontrada na geração 23ª de 500.

Exemplos de mochila (demais exemplos omitidos devido ao tamanho da saída):

```
 < 13, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
< 13, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
< 13, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
```

Abordagem: Penalização

Tamanho População (Np): 100

Máximo gerações: 500

P. Crossover (Pc): 90%

P. Mutação (Pm): 10%

Executando o algoritmo dez vezes obteve-se um total de 993 mochilas com valor igual a melhor solução (contando todas as execuções). A primeira solução foi encontrada na 3ª geração de 500.

Exemplos de mochila (demais exemplos omitidos devido ao tamanho da saída):

```
 < 14, 117.000000, 173.000000, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
1, 0, 1, 0, 1, 0, 0, 1, 1 >
< 14, 117.000000, 173.000000, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
1, 0, 1, 0, 1, 0, 0, 1, 1 >
< 14, 117.000000, 173.000000, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
1, 0, 1, 0, 1, 0, 0, 1, 1 >
```

Abordagem: Reparação

Tamanho População (Np): 100

Máximo gerações: 500

P. Crossover (Pc): 90%

P. Mutação (Pm): 10%

Executando o algoritmo dez vezes obteve-se um total de 3312 mochilas com valor igual a melhor solução (contando todas as execuções). A primeira solução foi encontrada na 94ª geração de 500.

Exemplos de mochila (demais exemplos omitidos devido ao tamanho da saída):

```
 < 13, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
< 13, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
< 13, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
```

Abordagem: Penalização

Tamanho População (Np): 100

Máximo gerações: 500

P. Crossover (Pc): 80%

P. Mutação (Pm): 5%

Executando o algoritmo dez vezes obteve-se um total de 418 mochilas com valor igual a melhor solução (contando todas as execuções). A primeira solução foi encontrada na 18ª geração de 500.

Exemplos de mochila (demais exemplos omitidos devido ao tamanho da saída):

```
 < 16, 117.000000, 175.000000, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0,
0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 0, 0, 0, 0, 1, 0, 1 >
< 16, 117.000000, 175.000000, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0,
0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 0, 0, 0, 0, 1, 0, 1 >
< 16, 117.000000, 175.000000, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0,
0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
0, 1, 0, 0, 0, 0, 1, 0, 1 >
```

Abordagem: Reparação

Tamanho População (Np): 100

Máximo gerações: 500

P. Crossover (Pc): 80%

P. Mutação (Pm): 5%

Executando o algoritmo dez vezes obteve-se um total de 3946 mochilas com valor igual a melhor solução (contando todas as execuções). A primeira solução foi encontrada na 18ª geração de 500.

Exemplos de mochila (demais exemplos omitidos devido ao tamanho da saída):

```
 < 13, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
< 13, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
< 13, 120.000000, 216.000000, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
1, 0, 0, 0, 1, 0, 1, 0, 1 >
```

Observando tanto o valor fitness encontrado como o número de mochilas com melhor valor podemos observar que o método de reparação obteve um

número muito maior de soluções. As saídas omitidas podem ser vistas em: https://github.com/Minoro/data-science-theory/tree/master/IA

5 Comparação de Cálculo de Fitness

Para as análises a seguir assume-se a função fitness como sendo apenas a soma dos valores dos itens da mochila, e também que é permitido a geração de uma população inicial com indivíduos infactíveis, sendo assim a penalização e reparação também devem ser aplicados aos pais.

Para a solução de penalização o cálculo de fitness é realizado na seleção por roleta para cada indivíduo (pais) e durante a penalização para cada pai e filho que estiver dentro do peso limite, pois caso seja penalizado o valor da mochila não é levado em consideração, apenas seu peso. Nesse caso em que a soma dos pesos não está sendo considerada para a análise, temos:

$$O(n) = Np_{pais} + (Np_{pais} + Np_{filhos} - Nc) \tag{1}$$

Onde Nc é o número de indivíduos que precisam ser penalizados, pois estão fora do limite de peso.

Assumindo o pior caso para o número de cálculos de fintess (não para a solução do problema), em que todos os filhos precisam ter o fintess calculado, temos $Nc = 0$ e como o número de filhos é o mesmo que o número de pais (Np), então:

$$O(n) = Np_{pais} + (Np_{pais} + Np_{filhos} - Nc) = Np + (Np + Np - 0) = 3Np \tag{2}$$

Ou seja, três vezes para cada indivíduo de uma geração, sendo um Np para a roleta e as demais avaliações para a penalização.

Já para a abordagem de reparação temos a avaliação fitness realizada para todos os indivíduos para a roleta (pais) e para cada indivíduo da população (pais e filho) que passe do peso é necessário calcular mais uma vez mais uma vez

Dessa forma temos:

$$O(n) = Np_{pais} + Np_{pais} + Np_{filhos} = 3Np \tag{3}$$

Comparando as abordagens, nesse cenário, para o pior caso temos para ambas as soluções $O(n) = 3Np$. Porém, o caso de de todos os indivíduos precisarem ser penalizados temos $Nc = Np$ aplicando em (1):

$$O(n) = Np + (Np + Np - Np) = 2Np \tag{4}$$

Dessa forma,temos que o cálculo da penalização é mais eficiente, pois há chances que não aplicar o cálculo de fitness a todos os indivíduos.

Vale lembrar que o cálculo do peso da mochila não está sendo levado em consideração, caso este cálculo seja considerado a reparação se torna ainda pior na comparação. Uma vez que é necessário calcular o peso da mochila para cada item removido.

6 Considerações

A abordagem utilizando a reparação tem uma melhora notável sobre a penalização, obtendo um número maior de mochilas com o maior resultado encontrado. Observando os gráficos das gerações pode-se observar que a reparação sempre se manteve acima da penalização, exibindo os melhores resultados.

A melhora, provavelmente, se dá devido ao fato de não haver indivíduos infactíveis após a reparação, e com isso se obtém melhores valores de fintess e uma convergência mais rápida para o valor ótimo.