Trabalho Prático I - Computação Evolutiva

Athos B. Lagemann, Pedro N. Costa

Instituto de Informática - Universidade Federal do Rio Grande do Sul (UFRGS)

Porto Alegre - RS - Brasil

{ablagemann, pncosta}@inf.ufrgs.br

1. Introdução

O problema da mochila é um problema clássico na área da computação, e possui uma inestimável importância para a área de otimização combinatória. Apesar de resolução simples para instâncias pequenas, por ser um problema de otimização combinatória, o custo da solução escala rapidamente com o tamanho da instância, ficando virtualmente impossível de ser solucionado com métodos tradicionais para instâncias maiores. Foi por este motivo que optamos por explorá-lo usando algoritmos genéticos.

O problema consiste em, dado um conjunto de itens com valor $\boldsymbol{v}$ e peso $\boldsymbol{p}$, deve-se escolher a melhor combinação de itens que comportem o peso $\boldsymbol{P}$ de uma mochila, otimizando ao máximo o somatório $\boldsymbol{V}$ dos itens escolhidos. Para este trabalho, escolhemos uma variante do problema que considera que os itens têm seu peso igual ao seu valor, e que a mochila é binária, onde os itens ou estão na mochila (1) ou não estão na mochila (0).

A formulação do problema utilizada consiste em, dado $\boldsymbol{n}$ itens com valores e pesos proporcionais, obter a melhor soma que satisfaz a capacidade da mochila. O nosso trabalho tem por objetivo comparar o método de seleção Elitista com o método de seleção por Torneio. A nossa hipótese é a de que o método de seleção Elitista tem um melhor desempenho sobre a seleção por Torneio, visto que a seleção elitista não descarta as melhores soluções encontradas até o presente momento, e a

maior diversidade da seleção por torneio pode ser prejudicial para certas soluções.

2. Trabalhos Relacionados

Obtivemos e avaliamos os princípios teóricos dos algoritmos genéticos no artigo *Algoritmos Genéticos: Fundamentos e Aplicações[1]*. A partir deste artigo, pudemos formular os princípios fundamentais do algoritmo genético geral, e a partir deste ponto, buscar as soluções mais adequadas ao problema específico.

A modelagem para o problema da mochila foi baseada na apresentada pelo artigo *Comparação de Métodos de Computação Evolucionária para o Problema da Mochila Multidimensional[5]*, que consiste em uma mochila binária multidimensional. Porém adaptamos esta modelagem para apenas uma mochila e com os itens com valores e pesos proporcionais.

A implementação do primeiro método de seleção foi feita baseando-se na implementação Elitista usada no artigo *Implementações Paralelas para o Problema da Mochila Multidimensional usando Algoritmos Genéticos[2]*. No referido artigo, foram selecionados apenas os dois indivíduos mais aptos, enquanto nós optamos por utilizar os $\sqrt{n}$ mais aptos, sendo n o número de indivíduos da população. Estes $\sqrt{n}$ indivíduos era cruzados entre si gerando novamente n indivíduos na próxima geração.

Optamos por tal método de seleção por ter a vantagem de otimizar o espaço de busca ao não descartar as soluções ótimas já encontradas, como pode ser verificado no artigo *Analysis of Selection Schemes for Solving an Optimization Problem in Genetic Algorithm*[3], que compara algumas implementações de diversos tipos de seletores para algoritmos genéticos aplicados em problemas de otimização combinatória.

Para o segundo método de seleção, optamos por utilizar a seleção por Torneio (*Tournament Selection*), onde são executados $\sqrt{n}$ torneios entre os

n indivíduos da população, selecionados aleatoriamente. O vencedor de cada torneio (o indivíduo com melhor *fitness*) é levado à próxima geração. Este método de seleção foi baseado nos conceitos expressos no artigo *Genetic Algorithm Performance with Different Selection Methods in Solving Multi-Objective Network Design Problem[4]*, e teve a sua eficiência comparada no artigo *Analysis of Selection Schemes for Solving an Optimization Problem in Genetic Algorithm[3]*, já citado no parágrafo anterior.

A partir do conteúdo absorvido dos artigos, evidencia-se que:

- Os algoritmos genéticos não utilizam métodos de busca totalmente aleatórios (fato evidenciado nos artigos *[2][3][4][5]* e na nossa implementação)
- Algoritmos genéticos são apropriados para trabalhar com espaços de busca grandes demais para técnicas de busca tradicionais

Os melhores resultados que obtivemos não são diretamente comparáveis com instâncias dos artigos analisados, por motivos de variações do problema. Contudo, os resultados gerais das técnicas utilizadas são condizentes com os resultados obtidos nos artigos referenciados, como será visto mais adiante.

3. Materiais e Métodos Utilizados

As instâncias usadas como entrada foram geradas a partir de um script de geração em Python. Define-se quantos itens haverão na mochila e qual a valor máximo de cada item da mesma. A partir disso, o script gera os itens e seus respectivos valores/pesos, assim como o valor que a mochila deve ter como solução. Optamos por utilizar uma entrada gerada automaticamente porque os artigos abordados não possuíam nenhuma relação com a variante do problema da mochila escolhido por nós. Os autores de artigos relacionados argumentam que gerar uma entrada para o problema é extremamente simples e portanto seria omitida.

A nossa implementação foi feita em Python 3.6, usando-se as bibliotecas *numpy* para o processamento e *matplotlib* para a geração dos gráficos. Optamos por utilizar duas variantes de seletor de geração e compará-las entre si.

O primeiro método de seleção que utilizamos foi a seleção elitista (*elitist selection*), que consiste em selecionar os *n* indivíduos mais aptos da última geração, passá-los para a próxima geração intactos, e cruzá-los (através do *crossover*), de modo a obter suas variações desejáveis. A vantagem deste método, como pode ser verificada no artigo [1], é a de manter as melhores soluções já exploradas e basear as gerações seguintes nestes indivíduos. O artigo [3] considera esta abordagem muito eficiente por acelerar a taxa de convergência do algoritmo, favorecendo as gerações mais aptas.

O segundo método de seleção escolhido foi a seleção por torneio (*tournament selection*), que consiste em gerar *n* torneios de dois participantes, escolhidos aleatoriamente, e persistir o que tiver melhor desempenho dos dois. A implementação da seleção foi similar à idealizada no artigo [4] (torneio binário), que considerou seus prós como sendo preservar a diversidade da população ao dar chance de indivíduos menos aptos a passarem para as gerações seguintes, uma complexidade de seleção mais eficiente e ser menos suscetível a ser tomada por indivíduos dominantes. O principal contra deste método de seleção é a degradação da velocidade de convergência, visto que soluções "ruins" têm chance de serem propagadas.

4. Resultados

Os resultados do experimento são extremamente satisfatórios. O algoritmo se comportou de maneira muito parecida utilizando tanto a seleção por torneio quanto a seleção elitista. Nem sempre o algoritmo foi capaz de encontrar a solução do problema dado, porém ele é muito mais

eficiente que o algoritmo de força bruta. Entradas com mais de 50 milhões de combinações possíveis foram resolvidas em menos de 80 mil gerações.

Outro ponto importante a se considerar é que notamos uma correlação entre o número de itens na mochila com a dificuldade do algoritmo em achar a solução. Em uma extremidade do desafio, onde a solução do problema era utilizar apenas um único item, o algoritmo genético não consegui resolver no limite de iterações dada. No extremo oposto, quando a solução era utilizar todos os itens da mochila, o algoritmo conseguiu resolver com certa facilidade. Nós acreditamos que isso se deva ao fato do algoritmo avaliar -1 no fitness do indivíduo que possui uma soma maior que a valor a ser somado na mochila. Por este motivo, quando o valor da mochila era muito baixo em relação aos itens, a maioria dos indivíduos recebiam um fitness de -1 e o algoritmo não conseguia convergir.

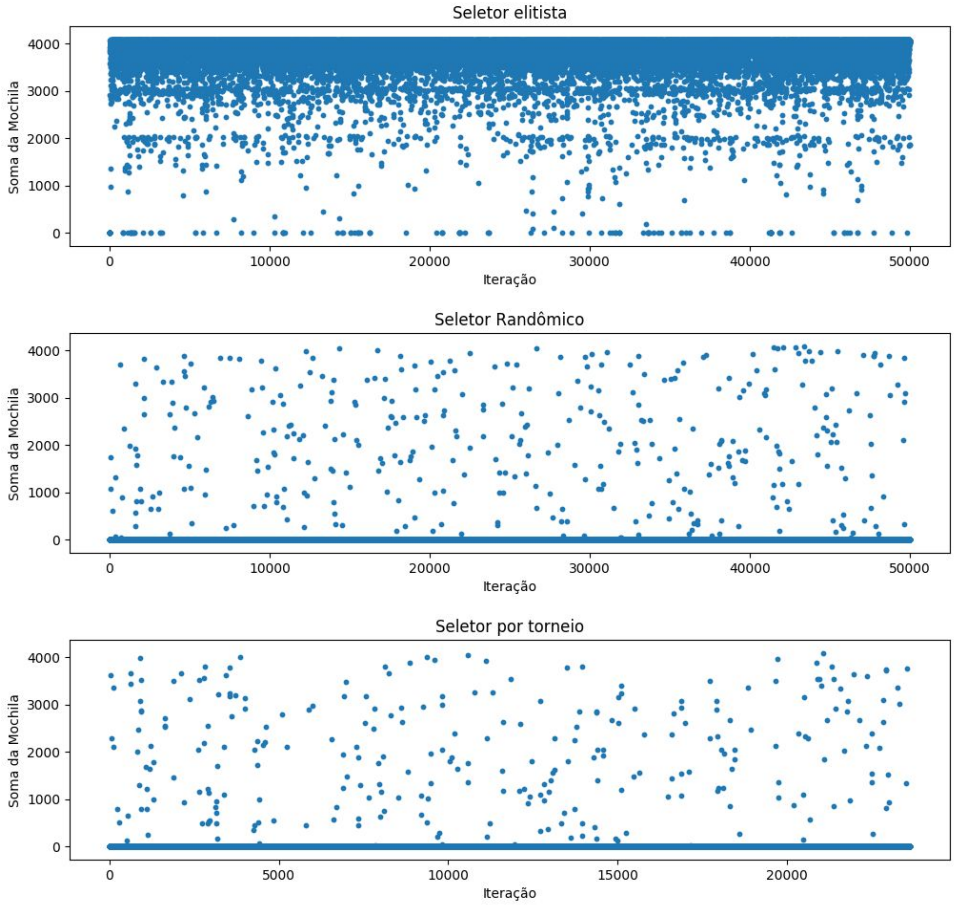

Figura 1: Comparação entre os métodos de seleção quando a solução possui um único item.

```
Rodando Seletor elitista:
Total de iterações  50000

Rodando Seletor randômico:
Total de iterações  50000

Rodando Seletor por torneio:
Solução encontrada:  [4096]
Total de iterações  23629
```

Figura 2: Saída gerada pelo algoritmo quando a solução do problema possui um único item

A figura 1 mostra o fitness, i.e. a soma dos valores de cada item, de cada indivíduo da população a cada iteração. Neste caso, o seletor por torneio teve o melhor desempenho por ter sido o único a achar a solução. Dá para notar nos gráficos que o espaço de busca do seletor por torneio e randômico são maiores que o do elitista, que focou em solução próximas ao valor da mochila, provavelmente com vários itens.

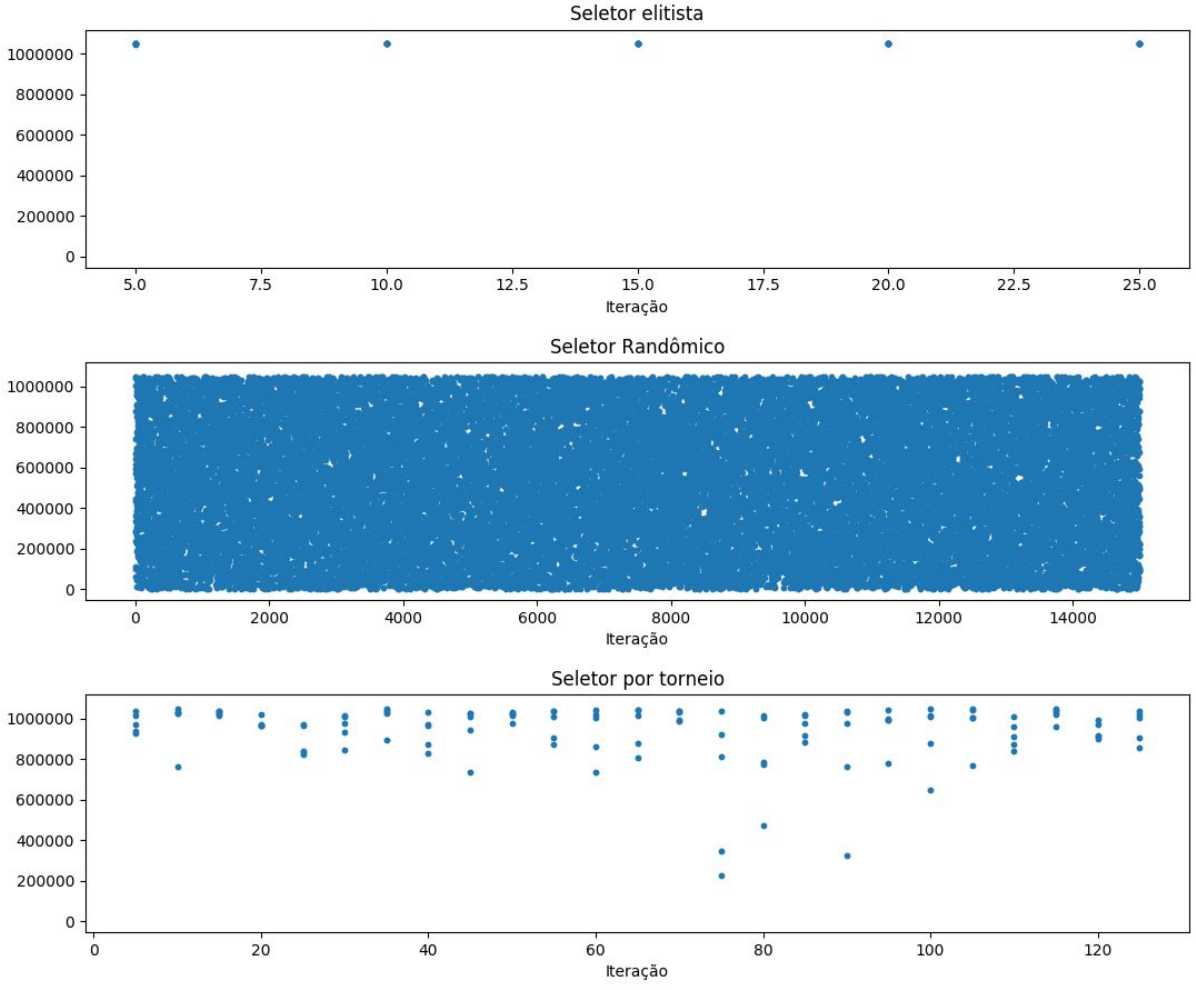

Figura 3: Comparação entre os métodos de seleção quando a solução possui todos os itens.

```
Rodando Seletor elitista:
Solução encontrada:  [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288]
Total de iterações  27

Rodando Seletor randômico:
Total de iterações  15000

Rodando Seletor por torneio:
Solução encontrada:  [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288]
Total de iterações  129
```

Figura 4: Saída do algoritmo quando a solução possui um único item.

A figura 3 mostra o fitness dos indivíduos de um teste com itens e mochilas aleatórios. O total de itens utilizado foi 60. O algoritmo elitista se saiu extremamente bem e com apenas 27 iterações conseguiu achar uma solução para o problema.

O algoritmo randômico se saiu melhor que um algoritmo de força bruta, que precisaria de cerca de meio bilhão de bilhão de tentativas para achar uma solução, enquanto que este achou uma solução em cerca de 15 mil gerações.

O algoritmo de seleção por torneio gerou um resultado bem interessante. Pelo fato dele não selecionar necessariamente sempre os melhores indivíduos de uma geração, as possíveis soluções encontradas a cada geração tendem a ser mais espalhadas pelo domínio de solução

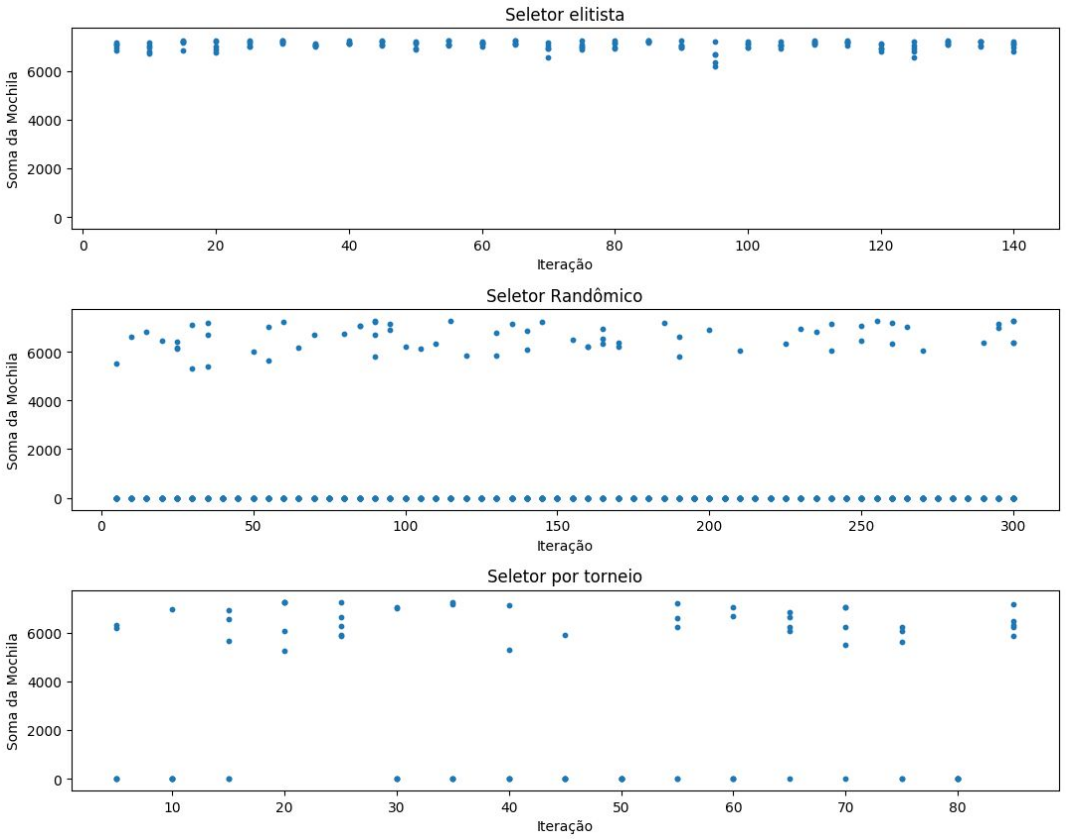

Figura 5: Comparação entre os métodos de seleção para uma entrada aleatória com 60 itens.

```
Rodando Seletor elitista:
Solução encontrada:  [3, 51, 82, 87, 87, 122, 127, 182, 218, 282, 309, 315, 320, 344, 382, 432, 454, 475, 541, 564, 581, 622, 689]
Total de iterações  144

Rodando Seletor randômico:
Solução encontrada:  [87, 122, 197, 218, 282, 309, 313, 320, 432, 468, 496, 511, 552, 574, 581, 587, 591, 629]
Total de iterações  300

Rodando Seletor por torneio:
Solução encontrada:  [51, 82, 89, 121, 182, 218, 290, 313, 315, 382, 384, 454, 468, 492, 498, 511, 581, 587, 622, 629]
Total de iterações  86
```

Figura 5: Saída do algoritmo para uma entrada aleatória com 60 itens

5. Conclusões

Como foi conjecturado, a seleção elitista converge para o resultado ótimo mais rapidamente do que a seleção por torneio. Isso foi explicado no artigo [1], justificado no artigo [3] e pode ser demonstrado por nós nos experimentos relatados.

Este resultado se deve ao fato de que a seleção elitista mantém e propaga as melhores soluções entre as gerações, ao passo de que a seleção por torneio propicia que soluções não tão boas sejam propagadas. Apesar da seleção por torneio possuir uma população mais diversa, ela acaba convergindo à solução ótima mais lentamente do que a seleção elitista.

As peculiaridades encontradas, como foi explicado nos resultados encontrados, é a de que o algoritmo consegue encontrar a solução mais facilmente nos casos em que a mochila possui mais itens, contraintuitivamente.

O trabalho atual pode ser estendido e comparado com outros métodos de seleção, visto que exploramos apenas dois dos inúmeros métodos abordados nos artigos analisados. Outra opção para a extensão do trabalho seria habilitar a mochila a possuir pesos e valores distintos, de modo a aumentar a complexidade das soluções. Contudo, para esta última extensão seriam necessárias diversas adaptações no código.

6. Bibliografia, Referências e Links

Link do código: *https://github.com/athoslag/Genetic-Knapsack.* [6]

(1) A.F. de Pinho, J.A.B. Montevechi, F.A.S. Martins, R.C. Miranda - *Algoritmos Genéticos: Fundamentos e Aplicações*, Omnipax.com.br, 2013.

(2) B.A. Dantas, E.N. Cáceres - *Implementações Paralelas para o Problema da Mochila Multidimensional usando Algoritmos Genéticos*, XLVI Simpósio Brasileiro de Pesquisa Operacional, 2014.

(3) P. Sharma, A. Wadhwa, Komal - *Analysis of Selection Schemes for Solving an Optimization Problem in Genetic Algorithm*, International Journal of Computer Applications, www.ijcaonline.org, 2014.

(4) R.O. Oladele, J.S. Sadiku - *Genetic Algorithm Performance with Different Selection Methods in Solving Multi-Objective Network Design Problem*, International Journal of Computer Applications, www.ijcaonline.org, 2013.

(5) J. Krause, J.A. Cordeiro, H.S. Lopes - *Comparação de Métodos de Computação Evolucionária para o Problema da Mochila Multidimensional*, Research Gate, 2013.

(6) Lagemann, Athos B., Costa, Pedro N. - Repositório com o código fonte desenvolvido para o trabalho da disciplina. Disponível em <https://github.com/athoslag/Genetic-Knapsack >

(7) KendallPark. - Repositório com a implementação do algoritmo genético implementado na linguagem Java. Disponível em <https://github.com/KendallPark/genetic-algorithm>

(8) E. Zitzler, L. Thiele - *Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach,* IEEE Computational Intelligence Society, 1999.

(9) P.C. ChuJ.E. Beasley - A Genetic Algorithm for the Multidimensional Knapsack Problem, Journal of Heuristics, 1998.