# Um Algoritmo Genético para o Problema da Árvore de Steiner em Grafos

**Antônio Costa de Oliveira[1], Filipe de Oliveira Saraiva[1]**

[1]Departamento de Informática e Estatística - Universidade Federal do Piauí (UFPI)
Teresina – Piauí – Brasil

costa@ufpi.com, wolverlipe@yahoo.com.br

***Abstract:*** *This paper deals with the implementation of a genetic algorithm for the Steiner Tree Problem in graphs. The definition of the problem, the conceptualization of genetic algorithm and a quarrel on the parameters used for the researchers in the resolution of the considered problem, are described in this paper. After that, the characteristics of the set chosen for the tests and the results gotten during the experiments are exposed.*

***Resumo:*** *Este artigo trata da implementação de um algoritmo genético para o Problema da Árvore de Steiner em grafos. Nele está descrito a definição do problema, a conceituação de algoritmo genético e uma discussão sobre os parâmetros utilizados pelos pesquisadores na resolução do problema proposto. Em seguida, expõem-se as características do conjunto escolhido para os testes e os resultados obtidos durante os experimentos.*

## 1. Introdução

Problemas de otimização combinatória são considerados de difícil solução devido a sua alta complexidade e custo computacional. Os algoritmos convencionais, baseados em análise de probabilidades, levariam, dependendo do problema analisado, uma quantidade de tempo demasiada alta para proporcionar uma solução em tempo viável.

Para tanto, existem diversas linhas de pesquisa voltada ao desenvolvimento de heurísticas. Heurísticas são métodos que procuram boas soluções a um custo computacional razoável. Tais pesquisas evoluíram os conceitos e implementações destes métodos, de forma que eles são comumente utilizados para a resolução de problemas de otimização combinatória.

Neste artigo, escolheu-se a heurística Algoritmo Genético para a resolução do Problema da Árvore de Steiner. Nele, encontrar-se-á a definição do problema e da heurística; em seguida, desenvolve-se uma discussão e descrição dos parâmetros utilizados na implementação, além da exposição das características dos conjuntos de dados a partir dos quais foram feitos os experimentos.

## 2. Problema da Árvore de Steiner em Grafos

Dado um grafo não-direcionado $G = (V, E)$, onde o custo das arestas é sempre positivo ou zero ($c: E \rightarrow R+$) para qualquer aresta pertencente a $E$. Temos

também um subconjunto de vértices do grafo *G*, que chamaremos de *S* (ou seja, $S \subseteq V$ ). O Problema da Árvore de Steiner em Grafos (PASG) consiste em encontrar um subgrafo *G' = (V', E')* tal que:

- *V'* contém todos os vértices de *S;*
- *G'* é conexo; e
- $\sum c(e)$ { $e \in E'$ } é mínimo.

Ou seja, o subgrafo *G'* conterá todos os nós de *S*, será conexo e a somatória das arestas que o formam deverá ser a menor possível. O subgrafo *G'* poderá conter vértices que não façam parte de *S*. A inclusão desse tipo de vértice, denominados Vértices de Steiner, normalmente são necessárias para a conexidade da solução e, dependendo daqueles que forem colocados, poderão reduzir ou não o custo total das arestas de *G'*.

Percebe-se então que a restrição do PASG não é a necessidade de que os vértices de *G'* sejam formados apenas pelo conjunto *S*, nem que o número de arestas de *G'* seja o mínimo possível. O problema requer a conexidade da solução e a necessidade de que o custo dessa conexidade seja o mínimo possível, podendo para este fim serem utilizados vértices que não pertençam a *S*.

Exemplificando, considere o grafo da Figura 1. (a) indica o grafo original, onde pode-se identificar os vértices do conjunto *S* (também chamados de Vértices Especiais) marcados com um círculo ao redor. São eles v1, v3, v7 e v9. (b) mostra visualmente como seria a resolução de um PASG sobre este grafo, onde os Vértices de Steiner estão indicados por retângulos; são eles v6 e v4. Esta solução determina o menor custo possível para se "ligar" os Vértices Especiais, que no caso usará os dois já citados Vértices de Steiner.

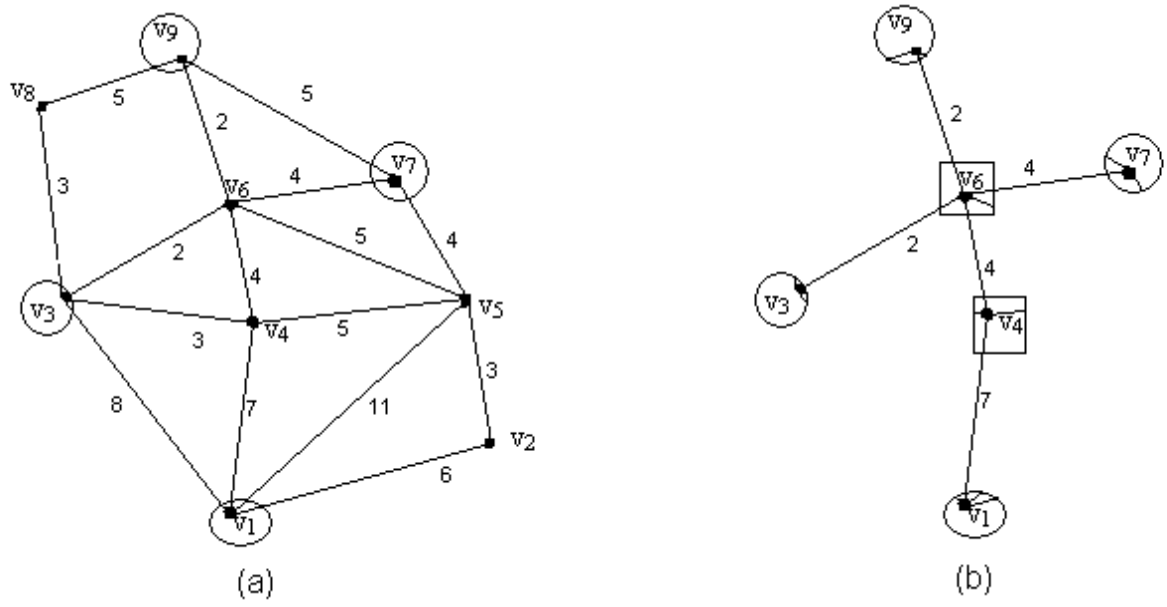


**Figura 1: (a) Um grafo mostrando Vértices Especiais (circulados). (b) A solução do PASG com os Vértices de Steiner (entre os retângulos).**

O PASG é considerado, ao lado do Problema do Caixeiro Viajante, um clássico problema de otimização combinatória. Em alguns casos especiais, o PASG pode ser solucionado em um tempo polinomial. Por exemplo, se $|S| = 2$, o PASG se resume ao problema do menor caminho entre dois vértices de um grafo, podendo ser solucionado sem problemas pelo Algoritmo de Dijkstra. Se $|S| = V$, o PASG se transforma no problema de se encontrar a árvore geradora mínima do grafo, podendo ser utilizado o Algoritmo de Prim ou o Algoritmo de Kruskal para encontrar sua solução.

O principal problema do PASG em seu caso geral é a dificuldade em saber quantos e quais vértices do grafo serão os Vértices de Steiner para se encontrar a melhor solução. Esse é o motivo pelo qual o problema, nessa situação, caracterizar-se como NP árduo. Quando se sabe quais serão os Vértices de Steiner para determinada resposta, une-se esse conjunto ao conjunto dos Vértices Especiais, determinando em seguida a árvore geradora mínima da solução. A soma das arestas que formam essa árvore será o custo total da mesma.

Observe novamente, como exemplo, a Figura 1. Na imagem (a) temos os Vértices Especiais que devem ser ligados e 5 outros vértices (a citar: v2, v4, v5, v6 e v8) que poderão ser utilizados para fazer essa conexão, a custo o menor possível. Na solução (b) foram utilizados os vértices v4 e v6, que garantem o baixo custo da conexão e formam, junto com os Vértices Especiais, a árvore geradora mínima.

A origem histórica do problema remete ao século XVII, derivado do problema euclidiano de Steiner . Proposto pelo renomado matemático francês Fermat, esse problema consistia em, dados três pontos no plano, encontrar um quarto ponto tal que a soma das distâncias desse último aos demais seria mínima. Atribui-se a Torricelli a resolução dessa versão do problema. Apesar de não ter sido Steiner nem o criador nem o solucionador do problema, foi ele o responsável por sua divulgação no meio científico da época, o que acabou por lhe valer o nome no mesmo.

Existem diversas aplicações para o PASG. Dentre elas, destacam-se:

- Projeto de circuitos eletrônicos;
- Planejamento de redes de comunicação;
- Tubulações de gás e óleo;
- Distribuição de água para irrigação e redes de drenagem;
- Projetos de instalação de redes elétricas e mecânicas;
- Edificações;
- Tráfego em redes de comunicação.

Como, no caso geral, o PASG é NP, métodos convencionais de resolução mostram-se ineficientes devido ao alto custo computacional dispendido para a resolução do problema. Em geral, seria preciso analisar todas as possibilidades, uma por vez, para se identificar o valor ótimo do problema. Este artigo descreve a utilização de uma heurística conhecida como Algoritmo Genético, usada por estes pesquisadores na resolução do PASG.

## 3. Algoritmos Genéticos

Algoritmos genéticos (AG) estão entre as heurísticas mais estudadas e desenvolvidas nas últimas décadas. Desenvolvida por Jonh Holland na década de 60, tem seus principais mecanismos baseados na dinâmica natural de populações genéticas. Sua concepção lembra bastante a teoria darwinista de evolução.

AG's são implementados como uma simulação de computador em que uma população de representações abstratas de soluções para um determinado problema passam por um processo evolutivo, como o que ocorre na natureza, onde melhores soluções são buscadas a cada geração da população.

A representação de soluções em um AG normalmente é feita através de uma *string* de *bits,* chamada de cromossomo, sendo que cada *bit* é comumente chamado de gene. Esse tipo de modelagem facilita os processos de cruzamento (*crossover)* e mutação, principais operadores no processo de evolução desenvolvido pelo método.

No início da execução do AG, é necessária a geração de uma população inicial, onde cada solução será representada na forma de cromossomo.

Dada uma população de soluções em um AG, faz-se um processo de seleção entre suas representações. Estando selecionadas as soluções passarão por um processo de cruzamento, onde seus *bits* serão permutados entre si, a partir de um determinado ponto. Em seguida o processo de mutação inicia-se, existindo uma probabilidade de determinado gene do cromossomo ser permutado para outro valor. A partir dessas fases têm-se novas soluções, que farão parte da próxima geração do problema, repetindo o ciclo até um número de gerações estabelecido.

É imprescindível que os valores do operador de cruzamento e mutação sejam bem avaliados e determinados. Também, o número de gerações e o tamanho da população inicial devem ser bem selecionados, pois a eficiência do AG dependerá enormemente dos valores aqui decididos. Normalmente, cada problema tem seus valores que otimizam o desempenho do AG. Para tanto, aqueles que forem implementar esta heurística, devem conhecer razoavelmente o problema que se propõem a resolver, fazendo diferentes simulações, alterando os valores dos parâmetros aqui expostos e procurando aqueles que melhor atenderem ao problema.

Com o decorrer do processo evolutivo, o AG procura tornar as soluções cada vez melhores para que (caso não encontre o valor ótimo para o problema pesquisado) encontre uma boa solução em tempo e custo computacional hábeis. Como as demais heurísticas, AG's são implementados principalmente para a resolução de problemas de otimização combinatória NP, que tem um alto custo computacional para serem resolvidos por algoritmos convencionais.

Algumas aplicações onde usam-se AG:

- Processamento de imagens;
- Evolução automática de programas de computador;
- Treinamento e *design* de sistemas de inteligência artificial;
- Trajetórias espaciais.

Na próxima sessão, será descrito algumas das características

implementadas pelo AG utilizado para resolver o PASG.

## 4. Características Gerais do Algoritmo Genético Implementado

Para se implementar um AG eficaz, é necessário conhecer razoavelmente o problema proposto, de modo que seja possível selecionar os parâmetros da heurística de uma maneira que os mesmos otimizem o desempenho do algoritmo. Assim, muitas vezes, é necessário fazer testes em relação aos valores dos parâmetros selecionados e verificar aqueles que melhor contribuem para o AG.

A representação das soluções em forma de cromossomo deu-se através do uso de *string* de *bits* 0 ou 1. A *string* tem tamanho $|V - S|$, que dá o número de possíveis Vértices de Steiner da solução. Como a solução do problema deve conter $S$, não haveria motivos para colocar esses vértices como possibilidades de sofrerem cruzamentos ou mutações, incorrendo no risco de não se resolver o PASG. Na representação, quando um *bit* da *string* tem valor 1, significa que o respectivo vértice de Steiner está na solução; se for 0, ele não estará.

Como exemplo, rememore a Figura 1: os vértices dela são v1, v2, v3, v4, v5, v6, v7, v8 e v9. Os Vértices Especiais são v1, v3, v7, v9. Logo, a representação *string* de uma solução terá tamanho 5 e representará, respectivamente, v2, v4, v5, v6 e v8. No caso, a representação cromossômica da resposta ao PASG, visualizada como (b), será 01010, pois apenas v4 e v6 fazem parte da resposta, enquanto v2, v5 e v8 não.

Também deve-se preocupar com o tamanho da população inicial e o número de gerações. Nesse caso, foram testadas populações de tamanho inicial 10 e 40 e número de gerações de 10 e 50. Como o programa executou de forma rápida, ficou estabelecido o tamanho da população em 40 e o número de gerações em 50, proporcionando maior diversidade genética à população.

Logo após, precisa-se definir de que maneira será gerada a população inicial. O AG recebeu testes em que a população inicial foi gerada aleatoriamente e, em outro, a população inicial foi completamente gerada pela solução que compreende $V' = V$ (ou seja, todos os vértices do grafo $G$ foram utilizados; todos os possíveis Vértices de Steiner foram incluídos na solução). A melhor maneira, segundo os experimentos, foi a geração aleatória, que disponibiliza uma maior diversidade genética para a população, tornando mais diversa as populações subseqüentes, geradas a partir de cruzamento e mutação.

Em seguida, é necessário uma maneira de saber o custo de cada solução encontrada pelo AG. Com a *string* formada, junta-se a ela os Vértices Especiais e computa-se a árvore geradora mínima para a solução analisada. O algoritmo usado para esse passo é o algoritmo de Prim. Esse passo também é importante porque ele acusa quando uma *string* gerada não é uma solução para o problema; talvez, por exemplo, ela não seja conexa. Assim o algoritmo de Prim garante que os indivíduos em cada população sejam realmente soluções para o PASG.

Após a formação de cada geração, o AG salva a solução que melhor responde ao PASG. Se, em uma próxima geração, existir um indivíduo que resolva o PASG e tenha seu custo de conexidade menor que o do indivíduo anteriormente salvo, este substituirá aquele, visto que é uma melhor solução.

Com o desempenho de cada solução computado, o AG guarda-as e passa para o processo de seleção. Neste passo, o algoritmo selecionará duas soluções para sofrerem o processo de cruzamento e mutação. O método de seleção usado é o método da roleta, que consiste na geração de um número aleatório entre 0 e $\sum[custos\,de\,conexidade\,da\,população]$ . A partir daí soma-se os custos das soluções, em ordem crescente, até aquele que alcance ou passe primeiramente o valor aleatório gerado. A solução por ele representada é então selecionada.

Na fase do cruzamento, foram testadas três maneiras: a primeira, o ponto de cruzamento acontecia na metade do cromossomo; na segunda, acontecia no ponto correspondente à 90% do tamanho do cromossomo; e a terceira, gerava-se aleatoriamente o ponto que sofreria cruzamento. Aparentemente, a terceira maneira proporcionou a geração de soluções melhores que as demais.

A mutação ocorre da seguinte forma: após o cruzamento, gera-se um número aleatório para cada gene do cromossomo; se esse número for menor que o valor do operador de mutação, esse gene será permutado. No caso, se seu valor for 0, ele tornar-se-á 1; do contrário, se ele for 1, terá seu valor permutado para 0. Caso o número gerado aleatoriamente seja maior que o valor do operador de mutação, nada ocorrerá.

Para o valor do operador de mutação, foram testadas duas possibilidades: em uma, ele valia 2%; em outra, 5%. Convencionou-se utilizar 5%, pois aparentemente soluções melhores foram formadas com esse valor. Após o processo, a solução será computada pelo algoritmo de Prim, indicando seu valor e verificando sua conexidade.

O tamanho da população para cada geração é o mesmo tamanho da população inicial. Na formação de cada geração, foram experimentadas duas maneiras: na primeira, as 10% melhores soluções de cada geração passavam para a próxima geração, sendo o restante da população constituído apenas pelos processos de cruzamento e mutação. A esse processo dá-se o nome de elitismo. De outra maneira, toda a população de cada geração era formada apenas por cruzamento e mutação; não havia elitismo de qualquer espécie. Essa última maneira produziu melhores soluções que a primeira.

É importante também ressaltar que foi adicionado um processo de seleção de soluções, que consiste em aceitar indivíduos para a próxima geração apenas se ele tiver seu custo associado menor que, ou igual a, o maior custo associado da geração anterior.

Esse método, aparentemente, faz com que a convergência para um valor comum a todos os indivíduos de uma população seja mais rápido no decorrer das gerações; porém garante que a próxima geração será melhor ou igual à geração anterior. Apesar do temor da convergência rápida levar a resolução a um ótimo local imprestável como solução ao problema, esse critério fez com que o AG tivesse um desempenho de soluções melhor do que sem ele.

Acabada a demonstração e discussão dos parâmetros escolhidos para o referido AG, será agora exposto os conjuntos usados como teste e os resultados dos experimentos.

## 5. Resultados

O AG foi testado no conjunto de problemas B, da OR-library de Beasley. Esse conjunto compreende 18 grafos, que vão de estruturas com 50 vértices, 63 arestas e 9 Vértices Especiais até 100 vértices, 200 arestas e 50 Vértices Especiais. Maiores detalhes do conjunto, assim como os os melhores valores encontrados na execução do AG estão na Tabela 1.

Para cada problema testado, o algoritmo foi executado 10 vezes. Os parâmetros utilizados das execuções aqui expostas foram aqueles discutidos e escolhidos na seção anterior desse artigo.

Analisando os resultados, é perceptível que o AG implementado conseguiu bons resultados para os diferentes conjuntos. Mesmo não sendo um algoritmo exato, ele conseguiu encontrar soluções ótimas para alguns dos problemas. Nos quais ele não encontrou o ótimo, aproximou-se, obtendo soluções altamente aceitáveis, sob o ponto de vista da complexidade dos problemas propostos.

Também é importante registrar que não houve qualquer tipo de adaptação do AG para a solução de um problema particular. Todos os PASG's apresentados foram resolvidos utilizando-se do mesmo parâmetro em todas as execuções de todos os conjuntos.

**Tabela 1: Características do conjunto B da OR-library e melhor resultado encontrado pelo AG**

| **Problema** | $\|V\|$ | $\|E\|$ | $\|S\|$ | **Custo Ótimo** | **Melhor Valor Encontrado** |
|---|---|---|---|---|---|
| **1** | 50 | 63 | 9 | 82 | 83 |
| **2** | 50 | 63 | 13 | 83 | 83 |
| **3** | 50 | 63 | 25 | 138 | 140 |
| **4** | 50 | 100 | 9 | 59 | 62 |
| **5** | 50 | 100 | 13 | 61 | 61 |
| **6** | 50 | 100 | 25 | 122 | 126 |
| **7** | 75 | 94 | 13 | 111 | 111 |
| **8** | 75 | 94 | 19 | 104 | 104 |
| **9** | 75 | 94 | 38 | 220 | 226 |
| **10** | 75 | 150 | 13 | 86 | 86 |
| **11** | 75 | 150 | 19 | 88 | 88 |
| **12** | 75 | 150 | 38 | 174 | 176 |
| **13** | 100 | 125 | 17 | 165 | 165 |
| **14** | 100 | 125 | 25 | 235 | 238 |
| **15** | 100 | 125 | 50 | 318 | 319 |
| **16** | 100 | 200 | 17 | 127 | 131 |
| **17** | 100 | 200 | 25 | 131 | 131 |
| **18** | 100 | 200 | 50 | 218 | 218 |

**Figura 2: Representação gráfica do Custo Ótimo e Melhor Valor Encontrado**

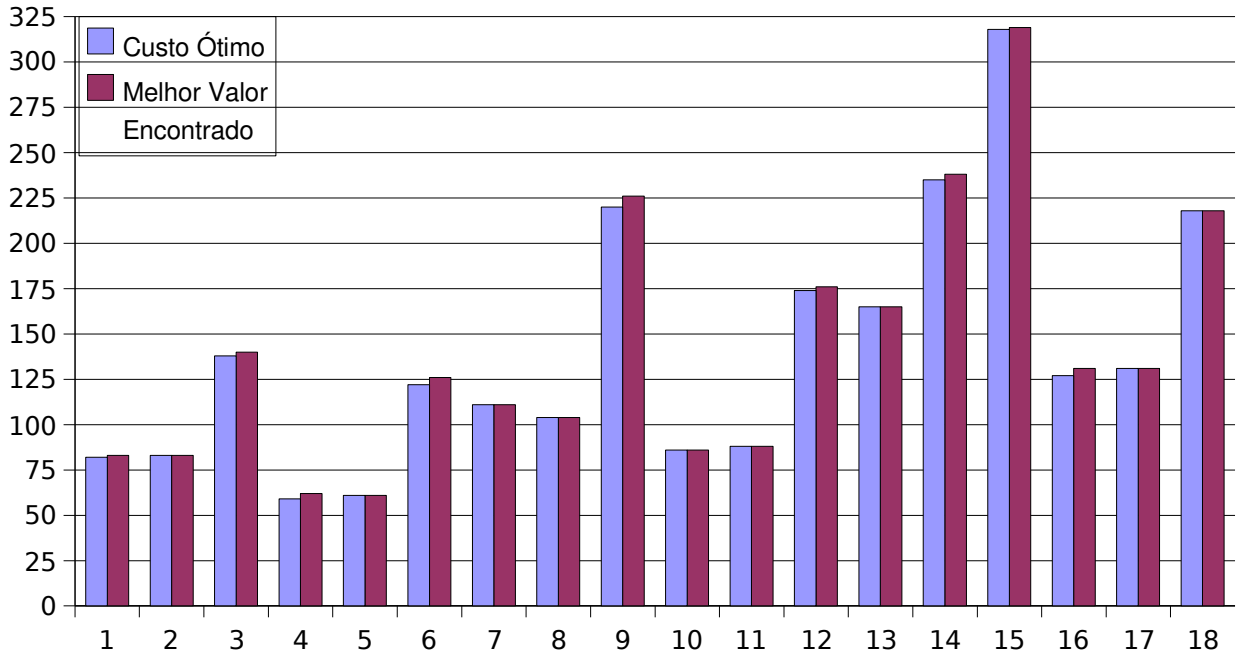


**Tabela 2: Resultado das 10 execuções do AG para cada problema analisado**

| **Problemas** | **Número de Execuções** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | **1ª** | **2ª** | **3ª** | **4ª** | **5ª** | **6ª** | **7ª** | **8ª** | **9ª** | **10ª** |
| **1** | 84 | 83 | 83 | 84 | 83 | 83 | 83 | 84 | 83 | 84 |
| **2** | 88 | **83** | **83** | **83** | **83** | **83** | **83** | **83** | **83** | **83** |
| **3** | 140 | 140 | 140 | 140 | 140 | 140 | 140 | 140 | 140 | 140 |
| **4** | 63 | 63 | 63 | 62 | 63 | 62 | 62 | 62 | 63 | 63 |
| **5** | **61** | 64 | **61** | **61** | 63 | **61** | **61** | **61** | **61** | **61** |
| **6** | 126 | 126 | 126 | 126 | 126 | 127 | 126 | 126 | 127 | 127 |
| **7** | **111** | **111** | **111** | **111** | **111** | **111** | **111** | **111** | **111** | **111** |
| **8** | 106 | 106 | 106 | 106 | 106 | 106 | 106 | **104** | 106 | 106 |
| **9** | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 | 226 |
| **10** | **86** | **86** | **86** | **86** | **86** | **86** | **86** | **86** | **86** | **86** |
| **11** | 90 | 91 | 91 | 89 | 90 | 89 | **88** | 91 | 90 | **88** |
| **12** | 179 | 180 | 180 | 179 | 176 | 179 | 176 | 180 | 180 | 180 |
| **13** | **165** | 170 | **165** | **165** | **165** | **165** | 170 | **165** | 170 | 170 |
| **14** | 238 | 238 | 238 | 238 | 238 | 238 | 238 | 238 | 238 | 238 |
| **15** | 319 | 319 | 319 | 319 | 319 | 321 | 319 | 319 | 323 | 323 |
| **16** | 133 | 133 | 133 | 131 | 142 | 138 | 133 | 134 | 133 | 135 |
| **17** | 134 | 133 | 135 | 134 | 133 | 135 | 135 | 135 | **131** | 133 |
| **18** | 220 | 220 | 220 | 220 | 220 | 225 | 220 | 220 | **218** | 220 |

***Obs.: Os números em negrito correspondem à solução ótima do problema encontrada.**

## 6. Conclusão

O AG apresentado neste artigo conseguiu desempenhar seu papel satisfatoriamente. Em um tempo e custo computacional hábeis, resolveu positivamente o PASG para o conjunto B da OR-library, encontrando boas respostas para alguns dos problemas e soluções ótimas para outros.

Diante da complexidade de alguns problemas de otimização combinatória, a exemplo do PASG, torna-se necessário o desenvolvimento do estudo das heurísticas. Um algoritmo convencional levaria um tempo demasiado longo para encontrar a solução de tais problemas, tornando sua implementação irrealizável.

A resolução de problemas desse tipo encontram cada vez mais aplicação nas mais diferentes áreas. O desenvolvimento, estudo e melhorias de heurísticas capazes de resolvê-los, a exemplo do AG, tornam-se então uma contribuição ao conhecimento e motivação suficiente para o contínuo aprofundamento sobre o tema.

## 7. Bibliografia


GOLDBARG, M. C.; LUNA, H. P. L. *Otimização Combinatória e Programação Linear,* Editora Campus, Rio de Janeiro 2005;

FURTADO, Antonio L. *Teoria dos Grafos: Algoritmos*, Livros Técnicos e Científicos Editora S.A., Rio de Janeiro, 1973;

COLEY, David A. *An Introduction to Genetic Algorithms for Scientists and Engineers*, World Scientific Publishing, 1999;

REEVES, Colin R. Modern Heuristic Techniques. *Modern Heuristic Search Methods*, pág.1 - 25, 1996;

VERHOEVEN, M.G.A.; SEVERENS, M.E.M.; AARTS, E.H.L. Local Search for Steiner Trees in Graphs, *Modern Heuristic Search Methods*, pág.117 - 129, 1996;

KAPSALIS, A.; RAYWARD-SMITH, V.J.; SMITH, G.D. Solving the Graphical Steiner Tree Problem using Genetic Algorithm, *Journal of the Operational Research Society,* vol. 44, nº. 4, pág. 397 – 406, 1993;

BEASLEY, J.E. An SST-Based Algorithm for the Steiner Problem in Graph, *Network*, vol. 19, pág. 1 – 16, 1989;

ESBENSEN, Henrik Computing Near-Optimal Solutions ti the Steiner Problem ins a Graph using a Genetic Algorithm, *Network*, vol. 26, pág. 173 – 185, 1995.