

# Algoritmo genético para resolver o problema do caixeiro viajante com o problemas de mochila para cálculo de rotas ótimas para robôs

Killdary A. Santana<sup>1</sup>

<sup>1</sup>Departamento de Engenharia Elétrica e da Computação  
Universidade Federal do Ceará (UFC) – Campus Sobral  
Caixa Postal 62010-560 – Sobral – CE – Brazil

killdary.aguiar@gmail.com

**Abstract.** *Route planning is becoming a daily task for activities that require the help of robots, such as robotic manipulators on assembly lines and transport robots in industries. The great challenge of these robots is to run a better route within their energy limitations. This paper presents the study and development of an application for the planning of beasted routes in the Traveling Salasmen Problem (TSP) combined with the Knapsack Problem (KP) using genetic algorithm (GA).*

**Resumo.** *O planejamento de rotas está se tornando a cada dia uma tarefa corriqueira para atividades que necessitem do auxílio de robôs, como manipuladores robóticos em linhas de montagem e robôs de transporte em indústrias. O grande desafio desses robôs é executar uma melhor rota dentro das suas limitações energéticas. Este artigo apresenta o estudo e desenvolvimento de uma aplicação para o planejamento de rotas baseados no Problema do Caixeiro Viajante(PVC) combinado ao Problema da Mochila(PM) por meio de algoritmo genético(AG).*

## 1. Introdução

Com o avanço da tecnologia os robôs se tornaram mais eficazes na execução de diversas tarefas, podendo executar funções que antes eram consideradas impossíveis para uma máquina, deixando de ser um componente caro e acessível apenas para grandes indústrias e passando a atuar nas mais diversas aplicações que antes eram executadas por humanos, impulsionando assim o desenvolvimento de robôs autônomos.

A finalidade da pesquisa de robôs móveis com autonomia é construir máquinas para realizar tarefas com precisão e capazes de tomar decisões adequadas frente a uma situação inesperada.

Muitos projetos podem ser observados com as mais diversas finalidades, como exploração de ambientes não estruturados como florestas[Vicente Santos et al. 2017], exploração de planetas como marte[NASA 2017], futebol de robôs[ROBOCUP 2017], carros autônomos[Google 2019], aplicações agrícolas[Tabile et al. 2011], entre muitos outros.

Um dos maiores desafios encontrados em robôs móveis se dá no planejamento de sua trajetória e tem recebido grande atenção por parte dos pesquisadores, tanto na

indústria como no meio acadêmico, pois seu desenvolvimento está diretamente relacionado com a maior autonomia dos robôs. A complexidade do problema de planejamento do movimento tem motivado o desenvolvimento dos mais diversos algoritmos. Tal complexidade advém da necessidade de integrar a navegação do robô com o sensoria-mento, a eficiência e o planejamento de rotas, como também poupar recurso importantes e preservação de suas partes mecânicas. A habilidade de planejamento de rotas deve estar presente na programação interna dos robôs para que os mesmos possam decidir quando será necessário alterar sua rota de modo a minimizar o impacto em comparação ao plane-jamento inicial.

O problema de roteirização de robôs pode ser modelado através do Problema do Caixeiro Viajante(PCV), porém o PVC não é suficiente para modelar a roteirização de rotas dentro de limitações do robô, como limitação energética ou de peso máximo a ser transportado, sendo assim o mesmo deve ser combinado ao Problema da Mochila(PM) para uma modelagem mais próxima a realidade. Este trabalho tem como objetivo desen-volver um algoritmo genético que forneça a melhor rota para um robô maximizando os ganhos que o mesmo pode realizar dentro de uma limitação energética.

Na próxima seção será descrito o PCV e sua variação síncrona, o PM e a combinação do PCV e PM. Na seção 3 será abordado a fundamentação teórica do prob- lema e da solução a ser implementada. A seção 4 abordará os métodos de implementação e suas metodologias. Na seção 5 será exposto os resultados obtidos oriundos da seção 4, enquanto a conclusão será apresentada na seção 6, e por fim na seção 7 os trabalhos futuros.

## **2. Problema do Caixeiro Viajante com o Problema da Mochila**

Para uma melhor representação do problema de roteirização de robôs com restrições será combinado o PCV e o PM. A seguir será apresentado o PCV e sua variação síncrona, usada como base do algoritmo, o PM com sua variação do Problema da Mochila Binária, para adicionar a visitação de uma cidade apenas uma vez, e por fim a combinação dos dois problemas na formulação matemática final.

### **2.1. Problema do Caixeiro Viajante**

O Problema do Caixeiro Viajante (PCV) tem sido muito utilizado no experimento de diversos métodos de otimização por ser, principalmente, um problema de fácil descrição e compreensão, mas de grande dificuldade de solução, uma vez que é NP-difícil(Non-Deterministic Polynomial time). O PCV determina que um vendedor tem N cidades no qual o mesmo deveria visitar todas as cidades, sem repetir nenhuma, e voltar para a cidade de partida de modo que o custo da viagem seja mínimo [Singh and Lodhi 2013]. O cálculo de rotas mínimas em robôs se assemelha ao PCV, sendo o robô o vendedor, as cidades os pontos objetivos que o robô deve visitar e as arestas a distância que o robô percorrerá. O PCV é um problema de otimização combinatória, no qual o número de soluções possíveis é representado na equação 1.

$$R(n) = (n - 1)! \quad (1)$$

Onde:

- $n$ : é o numero de cidades a serem visitadas;

### 2.1.1. Formulação Matemática Para o PCV

Seja o grafo  $G(N, A)$  onde  $N$  representa o conjunto ( $|N| = n$ ) e  $A$  o conjunto de arestas. Seja, uma matriz simétrica com custos ou distâncias mínimas entre os nós da rede considerando ainda que  $c_{ij} = +\infty \forall i \in N$ . A matriz  $X[x_{ij}]$  é composta pelas variáveis de decisão do problema

$$x_{ij} = \begin{cases} 1 & , \text{ se: o arco } a_{ij} \in \text{rota} \\ 0 & , \text{ se: o arco } a_{ij} \notin \text{rota} \end{cases} \quad (2)$$

Desta forma, a formulação de Programação Linear Inteira para o problema[Golden et al. 1980] pode ser escrita como:

$$\sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (3)$$

Sujeito a:

$$\sum_{i=1}^n x_{ij} \quad j = 1, 2, \dots, n \quad (4)$$

$$\sum_{j=1}^n x_{ij} \quad i = 1, 2, \dots, n \quad (5)$$

$$X = x_{ij} \in S \quad (6)$$

$$x_{ij} = 0 \quad \text{ou} \quad x_{ij} = 1 \quad (7)$$

Os dois primeiros grupos de restrições (4) e (5) garantem que exatamente um arco  $(i, j)$  tem origem cada nó  $i$  da rota e exatamente um arco  $(i, j)$  é direcionado para um nó  $j$  da rota. A penúltima restrição (6) contém um subconjunto  $S$  que pode ser qualquer conjunto de restrições que impeça a formação de sub-rotas. Estas restrições são chamadas restrições de quebra de sub-rotas e podem ser, entre outras:

$$S = \left\{ (x_{ij}) \geq 1, \text{ para todo conjunto próprio não vazio } Q \text{ de } N \right\} \quad (8)$$

$$S = \left\{ (x_{ij}) \leq |R| - 1, \text{ para todo subconjunto não vazio } R \text{ de } \{ 2, \dots, n \} \right\} \quad (9)$$

$$S = \left\{ (x_{ij}) : y_i - y_j + n x_{ij} \leq (n-1), \text{ para } 2 \leq i \neq j \leq n \text{ para alguns números reais } y_i \right\} \quad (10)$$

O PVC pode ser classificado em simétrico, onde os custos de um caixeiro ir da cidade A para a cidade B e vice-versa são os mesmos, ou assimétrico, onde os custos de movimentação de uma cidade A para uma cidade B podem ser diferentes de ir da cidade B para a cidade A. Para este trabalho será focado no problema **simétrico**.

### 2.1.2. Problema do Caixeiro Viajante Simétrico

O PCVS possui a direção da arestas é irrelevante já que  $c_{ij} = c_{ji}$ . Uma vez que a direção não tem importância, é possível considerar que o grafo possui apenas um arco, sem direção, entre todos os pares de nós. Deste modo,  $x_j \in 0, 1$  é a variável de decisão onde  $j$  percorre todos os arcos  $A$  do grafo e  $c_j$  é o custo de percorrer cada nó. Para encontrar um circuito no grafo, deve selecionar um subconjunto de arcos, no qual todos os nós estejam contidos. Assim, o problema pode ser formulado como:

$$R(n) = \frac{(n-1)!}{2} \quad (11)$$

de modo que o número de resposta é a metade da formulação (1). De modo semelhante ao assimétrico, os sub-circuitos devem ser eliminados. O problema pode então ser formulado como [de Almeida 2019]:

$$\min \frac{1}{2} \sum_{j=1}^n \sum_{i=1}^n c_{ij} x_{ij} \quad (12)$$

$$s.t. \sum_{k \in J(j)} x_k = 2, \text{ para todo } j = 1, \dots, m \quad (13)$$

$$\sum_{j \in E(S)} x_j \leq |S| - 1, \text{ para todo } S \subset 1, \dots, m \quad (14)$$

$$x_j = 0 \text{ ou } x_j = 1, \text{ para todo } j \in E \quad (15)$$

onde  $J(j)$  é o conjunto de todos os arcos, não direcionados, ligados ao nó  $j$  e  $E(S)$  é o subconjunto de todos os arcos que ligam as cidades em qualquer conjunto  $S$  não vazio que contenha todas as cidades [Wikipedia 2019].

### 2.2. Problema da Mochila (PM)

O PM é um problema de programação linear inteira, classificado como NP-hard. O PM consiste de uma mochila com capacidade limitada, dentro dessa mochila deve ser colocado uma lista de itens, cada um com seu peso e valor, de modo que seja agrupado um conjunto de itens na bolsa que sua soma de peso não ultrapassem a capacidade da mochila e a soma dos valores seja a maior possível [Martello and Toth 1990]. O cálculo da rota de um robô se assemelha ao PM, sendo o robô a mochila, a capacidade da mochila as restrições impostas ao robô, como peso máximo ou energia disponível, e os itens a serem colocados na mochilas são os pontos a serem visitados.

O PM possui diversas variações, porém um robô que tem sua rota com base no PCV só deverá passar por um ponto uma única vez, sendo assim a o PM binário é a representação do PM que mais aderente ao problema.

### 2.2.1. PM binário e sua Formulação Matemática

A variação PM binária consiste de um conjunto de itens  $N$ , que é formado por uma lista de  $n$  itens  $j$  com valor  $p_j$  e peso  $w_j$ , cada item é de um tipo único, e o valor da capacidade da mochila sendo  $C$ . O objetivo é selecionar um subset de  $N$  no qual a soma dos pesos não exceda  $C$  e o valor seja o maior possível[Prof. Hans Kellerer 2004]. A lista de itens é composta por um vetor  $Y[y_j]$ , no qual

$$y_i = \begin{cases} 1 & , \text{ se: o item } y_i \in \text{rota} \\ 0 & , \text{ se: o item } y_i \notin \text{rota} \end{cases} \quad (16)$$

Desta forma, a formulação matemática para o problema[Prof. Hans Kellerer 2004][Martello and Toth 1990] pode ser descrita como: r eliminados. O problema pode então ser formulado como [de Almeida 2019]:

$$\max \sum_{i=1}^n \sum_{i=1}^n p_i y_i \quad (17)$$

$$\text{s.t.} \sum_{i=1}^n w_i y_i \leq C \quad (18)$$

$$y_i \in \{0, 1\} \quad i = 1, \dots, m. \quad (19)$$

A restrição (18) garante que a capacidade da mochila não será ultrapassada. A restrição (19) só assume valor 1 ou 0. A equação (17) garante o maior valor possível na mochila.

### 2.3. Problema do Caixeiro Viajante Combinado ao Problema da Mochila (PCVPM)

Nesta seção, o objetivo deste artigo, a saber, o Problema do Caixeiro Viajante com Problema de Mochila (PCVPM). Pode-se dizer que esse problema é a junção do PCV Simétrico com o PM binário, de modo que, para formulá-lo, foi decidido que devem ser mantidas as principais características de cada um dos dois problemas componentes:

1. Cidades visitadas geram prêmios.
2. É necessário atender uma quantidade de cidades que proporcione a soma do custo entre cidades não ultrapasse o custo máximo estabelecido.

Obviamente, a primeira característica é relativa ao PM, enquanto a segunda é uma combinação entre PCV e PM, no qual o peso do item a ser posto na mochila foi substituído pelo custo de visitação entre as cidades. Observe que, dessa forma, nem todas as cidades precisam ser visitadas. O problema consiste em minimizar a soma dos custos de viagem, no qual sua rota é formada por clientes suficientes para percorrer uma distância menor ou igual a distância preestabelecida, maximizando os prêmios coletados a cada cliente visitado, de modo que as cidades inclusas na rota sejam visitadas uma única vez.

Mais precisamente, consideremos um grafo  $G(N, A)$ , onde  $N$  representa o conjunto ( $|N| = n$ ) e  $A$  o conjunto de arestas. Seja, uma matriz simétrica com custos ou

distâncias mínimas entre os nós da rede considerando ainda que  $c_{ij} = +\infty \forall i \in N$ . As cidades  $x_i$  que compõe o conjunto  $N$  possuem um prêmio  $p_i$ . O custo máximo percorrido seja  $C$ . A matriz  $X[x_{ij}]$  é composta pelas variáveis de decisão do problema:

$$x_{ij} = \begin{cases} 1 & , \text{ se: o arco } x_{ij} \in \text{rota} \\ 0 & , \text{ se: o arco } x_{ij} \notin \text{rota} \end{cases} \quad (20)$$

$$x_i = \begin{cases} 1 & , \text{ se: a cidade } x_i \in \text{rota} \\ 0 & , \text{ se: a cidade } x_i \notin \text{rota} \end{cases} \quad (21)$$

Desta forma, a formulação de Programação Linear Inteira para o problema pode ser escrita como:

$$\max f(x) = \alpha_P \sum_{i=0}^n p_i x_i - \beta_D \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (22)$$

sujeito as restrições:

$$\sum_{j=2}^n x_{1j} = 1 \quad (23)$$

$$\sum_{k=1}^{n-1} x_{in} = 1 \quad (24)$$

$$\sum_{i=1}^{n-1} x_{ni} = 0 \quad (25)$$

sobre a quantidade de visitas:

$$\sum_{i=1}^n x_{ij} \quad j = 1, 2, \dots, n \quad (26)$$

$$\sum_{j=1}^n x_{ij} \quad i = 1, 2, \dots, n \quad (27)$$

sobre a distância máxima a ser percorrida:

$$\sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \leq C_T \quad (28)$$

sobre a eliminação de subrotas:

$$1 \leq u_i \leq n \quad \forall i \in T \quad (29)$$

$$i_i - u_j + 1 \leq (1 + -x_{ij})n \quad 2 \leq i \neq j \leq n \quad (30)$$

onde  $n$  é o tamanho do conjunto de cidades a serem visitadas,  $T$  é conjuntos de cidades que comõe uma rota viável,  $c_{ij}$  é o custode movimentação entre uma cidade e outra,  $p_i$  é o prêmio da  $i$ -ésima cidade,  $\alpha_P$  e  $\beta_D$  são os pesos na fução objetivo relaciona-dos a maximização da premiação e minimização do custo

Neste problema as equações (23) e (24) são restrições sobre o ponto inicial e final que devem ser inclusos na rota, a restição (25) impõe que nenhum ponto apos chegar ao ponto final seja visitado, (26) e (27) restringe que cada ponto seja visitado apenas uma única vez, a equação (28) limita a distância máxoma percorrida a  $C_T$ , subrotas são impedidas de ser geradas pelas equações (29) e (30).

O resultaddesta otimização será um conjunto de vertices ordenados a serem visi-tados pelo agente.

### 3. Algoritmo Genético

Os Algoritmos Genéticos (AGs) são técnicas de otimização inspirados no princípio da sobrevivência e reprodução dos indivíduos mais aptos, proposto por Charles Darwin[Coppin 2004][Goldberg 1989].

O Algoritmo Genético é baseado em um conjunto de soluções candidatas que representam uma solução para o problema de otimização que se deseja resolver. Uma solução é um candidato em potencial para otimizar o problema. A forma de representação desem-penha um papel importante, uma vez que determina a escolha dos operadores genéticos.

As representações geralmente são listas de valores e por muitas vezes são mais baseadas em conjuntos de símbolos. Caso o conjunto de símbolos sejam contínuos, eles serão chamados de vetores, caso eles consistem em bits, eles serão chamadas de strings de bits. Em caso de problemas combinatórios, as soluções geralmente consistem em símbolos que aparecem em uma lista. A representação pode ser aplicado ao planejamento de rotas como um percurso realizado pelo caixeiro viajante. Os operadores genéticos produzem novas soluções a partir da representação escolhida e permite a aproximação da solução. A codificação da solução, no qual pode ser aplicado o processo evolutivo, é chamada de genótipo ou cromossomo.

O algoritmo 1 mostra o pseudocódigo do AG básico. No início, um conjunto de soluções, chamado de população, é inicializada. Esta inicialização é utilizada para gerar aleatoriamente o conjunto de soluções[Kramer 2017].

O principal ciclo geracional do Algoritmo Genético gera uma nova prole de soluções candidatas, através do cruzamento e da mutação, até a população estar completa.

Para a metodologia de de eleição foi optado peloa forma de *ranking*. Quanto ao cruzamento foi utilizado 3 métodos: *one point*, *two points* e *sinusoidal*[?]. Já para as mutações foram realizados 4 tipos: *Swap*, *Scramble*, *Worst gene with random gene muta-tion* e *Worst gene with nearest neighbor mutation* [Hassanat et al. 2016].

---

**Algoritmo 1:** Algoritmo Genético

---

```
1  iniciar população; repeat
2    repeat
3      cruzamento;
4      mutação;
5      avaliação;
6      seleção;
7    until;
8    população completa;
9    atualização teste de Parada
10 until;
11 finalização
```

---

## 4. Resultados

Nesta seção serão apresentados os resultados da simulação do cálculo de rotas para um agente para os seguintes cenários:

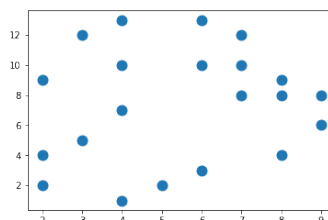
- Cenário 1 - Rota para 20 pontos, sem restrições de custo;
- Cenário 2 - Rota para 20 pontos, com limitação de custo;
- Cenário 3 - Rota para 30 pontos, sem restrições de custo;
- Cenário 4 - Rota para 30 pontos, com limitação de custo;

Para cada cenário foi gerado um conjunto de pontos a serem visitados, e para cada ponto foi atribuído uma recompensa. Para o cálculo de custos entre os pontos foi utilizado o cálculo da distância euclidiana.

A seguir serão mostrados os resultados das simulações para cada um dos 8 cenários apresentados.

### 4.1. Cenário 1 - Rota para 20 pontos, sem restrições de custo

O cenário 1 relacionado com a simulação do cálculo de rota onde o agente não possui restrições de custo, e pode percorrer todas as cidades livremente. Para o cenário 1 foi utilizado os seguintes pontos.

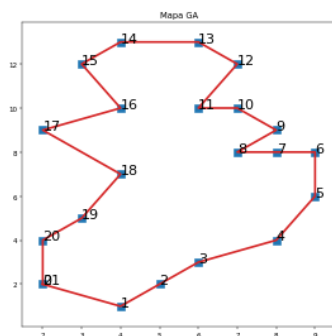


**Fig. 1. 20 pontos**

Após a execução do algoritmo a seguinte rota foi encontrada:

Juntamente com a distância mínima de 37,38 metros e a pontuação máxima de 125 pontos.

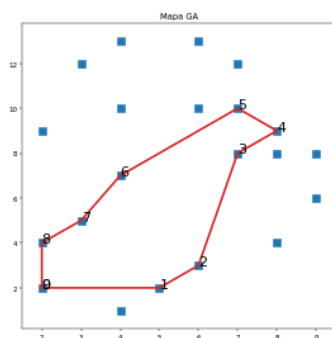




**Fig. 2. Rota 1**

#### 4.2. Cenário 2 - Rota para 20 pontos, com limitação de custo

O cenário 2 é estabelecido em cima dos mesmos pontos estabelecidos no cenário 1(1), porém agora ao algoritmo foi imposta a restrição de percorrer no máximo 23 metros, sendo assim o seguinte resultado foi encontrado.

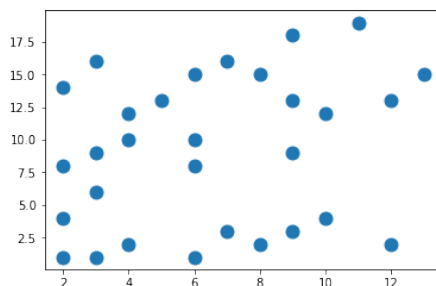


**Fig. 3. Rota 1**

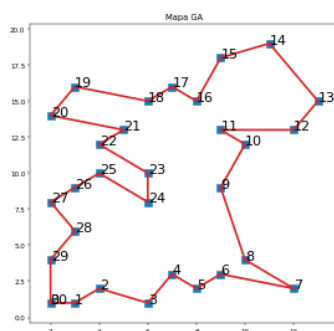
Para o Cenário 2 o algoritmo foi executado um total de 30 vezes, sendo a 22,23 metros com uma pontuação de 58,0 pontos.

#### 4.3. Cenário 3 - Rota para 30 pontos, sem restrições de custo

O cenário 3 relacionado com a simulação do cálculo de rota onde o agente não possui restrições de custo, e pode percorrer todas as cidades livremente. Para o cenário 3 foi utilizado um conjunto de 30 pontos a serem visitados, como mostrado a seguir.



**Fig. 4. 30 pontos**



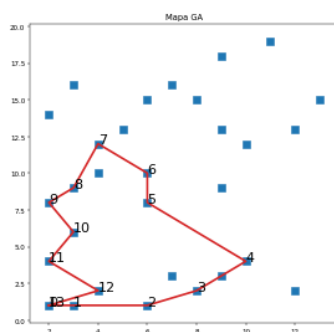
**Fig. 5. Rota 3**

Após a execução do algoritmo a seguinte rota foi encontrada:

Juntamente com a distância mínima de 71,24 metros e a pontuação máxima de 149 pontos.

#### **4.4. Cenário 4 - Rota para 30 pontos, com limitação de custo**

O cenário 4 foi estabelecido em cima dos mesmos pontos utilizados no cenário 3(1), porém agora ao algoritmo foi imposta a restrição de percorrer no máximo 35 metros, sendo assim o seguinte resultado foi encontrado.



**Fig. 6. Rota 3**

Para o Cenário 4 o algoritmo foi executado um total de 30 vezes, sendo a 33,66 metros com uma pontuação de 71,0 pontos.

## **5. Conclusão**

O algoritmo proposto neste trabalho, visa principalmente a otimização e a busca por soluções satisfatórias, em ambientes que possuam ou não alguma restrição no agente. os testes executados mostraram resultados promissores para a busca de rotas otimizadas com restrições, porém ainda inconclusivas já que para uma real avaliação será necessário um número maior de variabilidade de pontos e premiações para estudar melhor o comportamento do algoritmo.

Como proposta futura, tem-se a intenção de adaptar o algoritmo a utilizar rotas geradas pelo TSPLIB, e alterar possíveis formas de mutação e crossover, e o claculo de rotas com restrições para múltiplos agents.

## References

- CASS, S. (2018). The 2018 top programming languages.
- Coppin, B. (2004). *Artificial Intelligence Illuminated*. Jones and Bartlett Publishers, 1st ed edition.
- de Almeida, T. A. (2019). Método do subgradiente para a resolução da relaxação lagrangeana do problema do caixeiro viajante simétrico.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1 edition.
- Golden, B., Bodin, L., Doyle, T., and W. Stewart, J. (1980). Approximate traveling salesman algorithms. *Operations Research*, 28(3-part-ii):694–711.
- Google (2019). Just press go: designing a self-driving vehicle.
- Hassanat, A., Alkafaween, E., Alnawaiseh, N., Abbadi, M., Alkasassbeh, M., and Alhasanat, M. (2016). Enhancing genetic algorithms using multi mutations: Experimental results on the travelling salesman problem. *International Journal of Computer Science and Information Security*, 14:785–801.
- Jones, E., Oliphant, T., Peterson, P., et al. (2001–). SciPy: Open source scientific tools for Python. [Online; accessed 2017-12-21].
- Kramer, O. (2017). *Genetic Algorithm Essentials*. Springer.
- Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc., New York, NY, USA.
- Mitchell, R. (2017). Battle of the sbcs: Beaglebone black, raspberry pi 3, and asus tinker board.
- NASA (2017). Curiosity overview.
- Prof. Hans Kellerer, Prof. Ulrich Pferschy, P. D. P. a. (2004). *Knapsack Problems*. Springer-Verlag Berlin Heidelberg, 1 edition.
- ROBOCUP (2017). Robocup about.
- Singh, S. and Lodhi, E. A. (2013). Study of variation in tsp using genetic algorithm and its operator comparison.
- Tabile, R. A., Godoy, E. P., Pereira, R. R. D., Tangerino, G. T., Porto, A. J. V., and Inamasu, R. Y. (2011). Design and development of the architecture of an agricultural mobile robot. *Engenharia Agrícola*, 31:130 – 142.
- Vicente Santos, A., De Góes, E., Miranda, F., Freitas, G., Almir De Sena, J., Robinson, N., Dos Reis, S., Eduardo, P., Panta, G., Carvalho Ferreira, R., and Curty Cerqueira, R. (2017). Robô ambiental híbrido: um novo conceito em locomoção e monitoramento de áreas inóspitas na floresta amazônica.
- Wikipedia (2019). Wikipedia.