Laboratório 4 - Otimização com Métodos Baseados em População

Carlos R. A. Figueiredo[1]
Instituto Tecnológico de Aeronáutica, Laboratório de Inteligência Artificial para Robótica Móvel - CT-213. Professor Marcos Ricardo Omena de Albuquerque Máximo, São José dos Campos, São Paulo, 10 de abril de 2021.

[1]E-eletrônico: carlos.figueiredo@ga.ita.br

Descrição em alto nível da implementação:

A implementação consistiu na escrita da classes `Particle` e `ParticleSwarmOptimization` e da função `evaluate` contida dentro da classe `Simulation`.

A classe `Particle` terá apenas atributos, sendo dois relativos à posição, à velocidade e ao valor(custo) da partícula e mais outros dois relativos à posição e ao valor(custo) da melhor iteração local daquela partícula.

Para a função `evaluate` foi calculado o reward e retornado seu valor. Lembrando que deve-se receber uma punição maior caso o robô saia completamente da linha. A fórmula do reward usada foi, sendo o valor utilizado para w foi de 1 :

$$reward_k = v_k * dot(r_k, t_k) - w * |e_k|$$

A inicialização da classe `ParticleSwarmOptimization` gera um vetor com *n* partículas ,definido é definido melhor global(inicialmente *None*) e seu valor (inicialmente *-inf*). Finalmente, ainda na inicialização cria-se a variável `position = 0` para marcar a posição vetor da partícula que será avaliada naquela iteração.

Nessa classe temos 5 funções:

- `initialize_particles()`: que serve para gerar as partículas durante a inicialização da classe.
- `get_best_position()`: que retorna a posição(x) da melhor partícula global.
- `get_best_value()`: que retorna o valor(custo) da melhor partícula global.
- `get_position_to_evaluate()`: que retorna a posição(x) da partícula que será avaliada naquela iteração.
- `advance_generation()`: verifica se a partícula que está na posição position é o melhor global, depois verifica se ela é a melhor iteração local e por fim é incrementado sua v e x. Após isso mudamos o valor da variável position para position + 1(dentro do range do número de

partículas), com a finalidade de que na próxima iteração seja analisada outra partícula.

- `notify_evaluation()`: recebe o valor de qualidade analisado da partícula inserido esse valor no objeto e depois chama a função `advance_generation()`.

Foram executados dois arquivos, um de teste (`teste_pso.py`) e o principal (`main.py`). No arquivo de teste, foram obtidos os gráficos referentes as figuras abaixo. Eles são consistentes, já que para o teste sabia-se que a melhor qualidade deveria convergir para 0 e a convergência dos parâmetros deveria [1 2 3], exatamente como aconteceu.

Figura 1. Best Quality Convergence

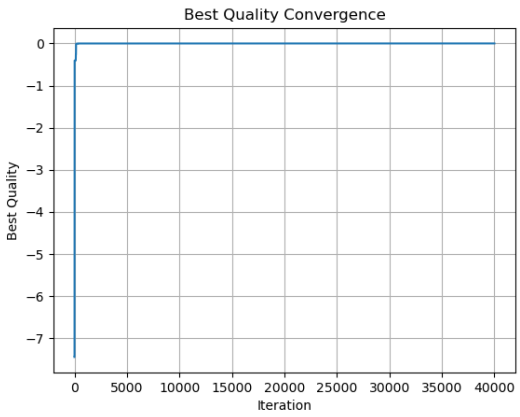


Figura 2. Quality Convergence

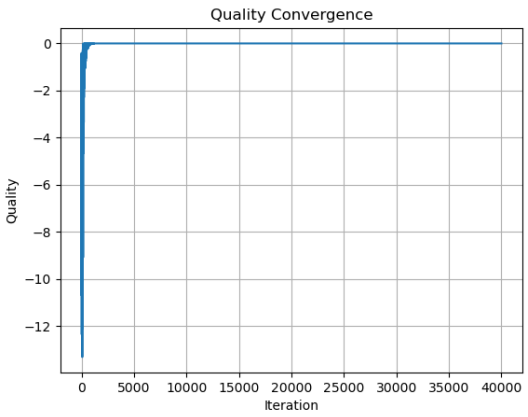
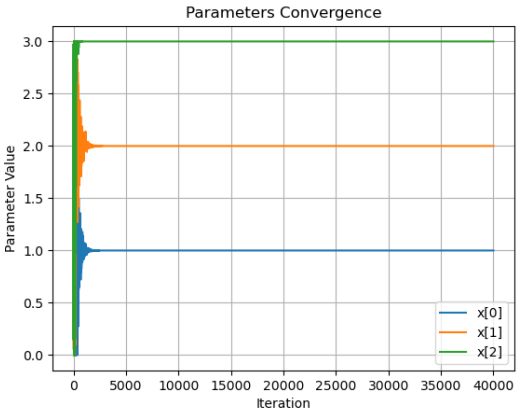


Figura 3. Parameters Convergence



Para o arquivo principal aguardou-se um pouco mais de 20.000 iterações, e conforme pode ser visto pelos gráficos abaixo, houve convergência para um ótimo global. O valor dos parâmetros encontrados no ótimo global foram [0.685382, 102.829336, 651.360836, 16.090761], obtendo-se assim uma qualidade de 517.07.

Figura 4. Best Quality Convergence

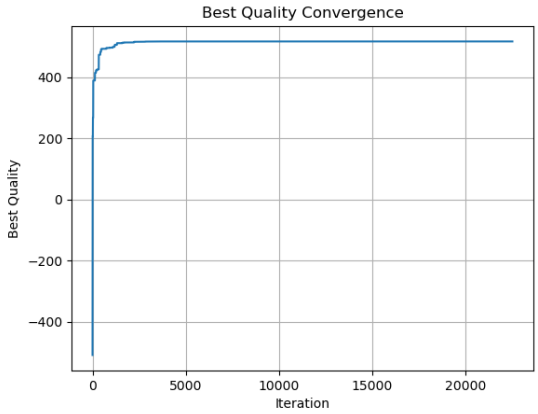


Figura 5. Quality Convergence

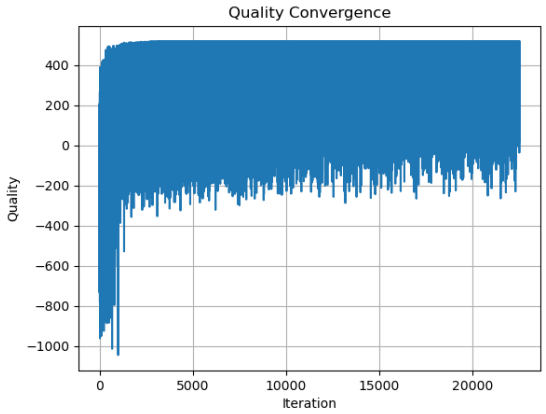
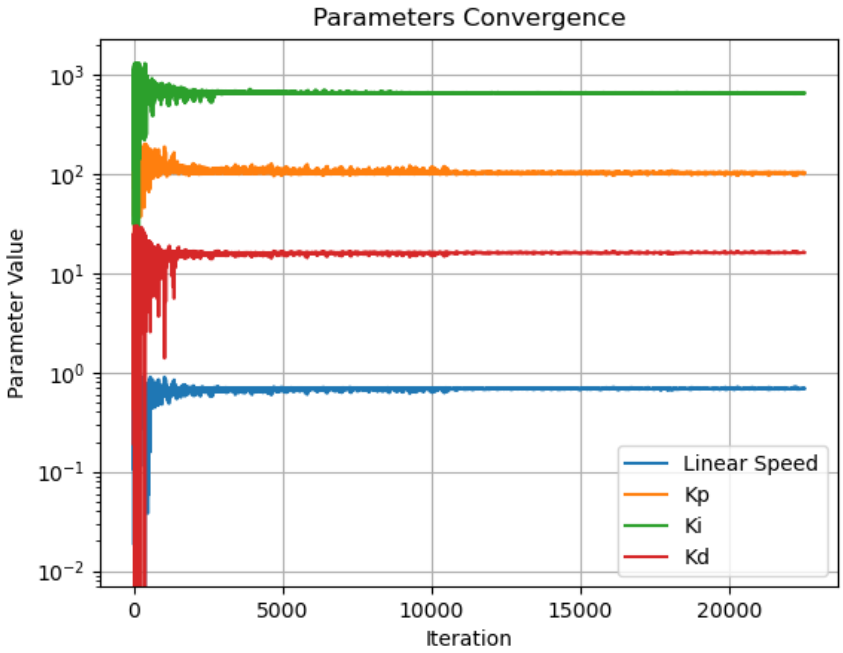


Figura 6. Parameters Convergence



Pode-se dizer que a simulação foi satisfatória, já que, observando na figura abaixo a trajetória feita pelo carrinho com os parâmetros do ótimo global, o carrinho seguiu a linha.

Figura 7. Trajetória do carrinho com os parâmetros do ótimo global