Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT213
Aluno: Arthur de Sousa Vianna

Relatório do Laboratório 4 - Otimização com Métodos Baseados em População

1 Breve Explicação em Alto Nível da Implementação

1.1 *Particle Swarm Optimization*

O algoritmo PSO consiste de um método de otimização que, a partir de um conjunto de candidatos à solução, memoriza a melhor posição (que resultou no melhor valor da função de otimização) de cada partícula e considerando-se todas as partículas (global), conciliando *exploration* e *exploitation* por meio da definição de "velocidade" para a atualização da posição da partícula. No código, o PSO foi implementado primeiramente definindo-se os atributos da partícula, como a posição e a velocidade na geração atual, a melhor posição alcançada pela partícula e o valor atual e o melhor valor alcançado pela partícula em relação à função de otimização. Além disso, há uma booleana indicando se foi atribuído o valor da posição da partícula na geração atual. A inicialização da posição e da velocidade consiste de uma distribuição uniforme entre os limites para cada dimensão, que são passados como argumentos na inicialização da partícula. A inicialização do PSO consiste na declaração do conjunto de partículas e dos hiperparâmetros, passados como argumento, além dos limites. Como, nesse caso, busca-se a maximização da função de otimização, o melhor valor de cada partícula era inicilizado como menos infinito. O valor de cada partícula relativo à função de otimização era realizado fora da classe, e esse valor era obtido pela função `notify_evaluate`. Quando todas as partículas eram avaliadas, passava-se uma geração por meio da função `advance_generation`, em que a posição de cada partícula era atualizada a partir da redefinição da velocidade, conforme a Equação 1.

$$\mathbf{v}_i = \omega \mathbf{v}_i + \phi_p r_p (\mathbf{b}_i - \mathbf{x}_i) + \phi_g r_g (\mathbf{b}_g - \mathbf{x}_i), \tag{1}$$

em que ω, ϕ_p e ϕ_g são hiperparâmetros, $\mathbf{b}_i$ é a melhor posição da partícula i, $\mathbf{b}_g$ é a melhor posição global e $\mathbf{x}_i$ e $\mathbf{v}_i$ são a posição e a velocidade da partícula i, respectivamente, e r_p e r_g são parâmetros estocáticos definidos por uma distribuição uniforme entre 0 e 1. Ainda na função `advance_generation`, a posição e a velocidade de cada partícula era limitada pelos limites definidos, em que o limite para a velocidade era $[-(\mathbf{u} - \mathbf{l}), (\mathbf{u} - \mathbf{l})]$, onde $\mathbf{u}$ é o vetor limite superior para a posição e $\mathbf{l}$, o inferior.

2 Figuras Comprovando Funcionamento do Código

2.1 Teste do *Particle Swarm Optimization*

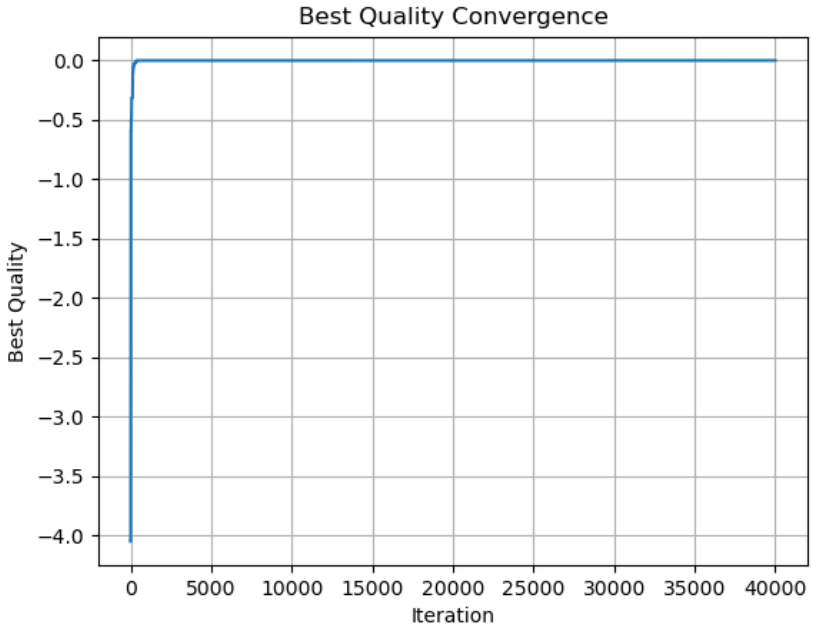

Figura 1: Convergência do melhor valor da função de otimização para o teste da implementação do PSO

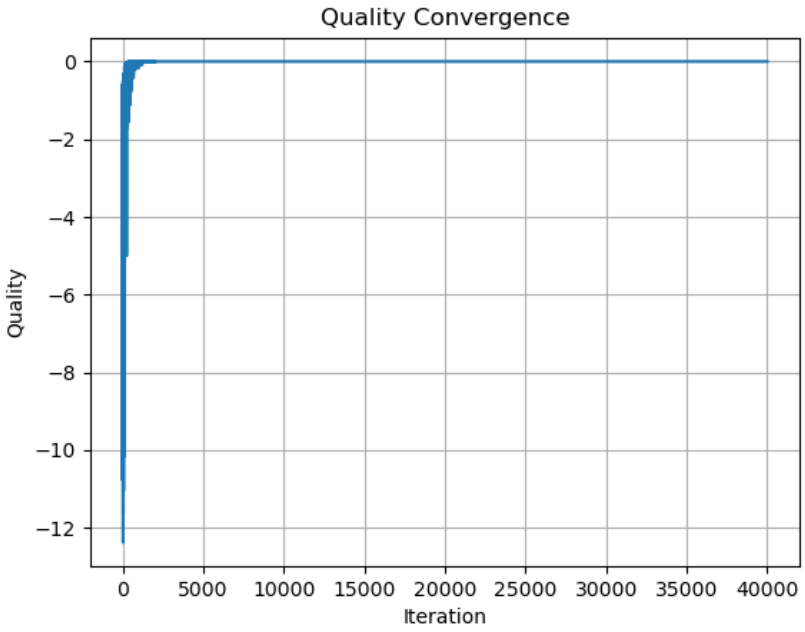

Figura 2: Convergência do valor da função de otimização para o teste da implementação do PSO

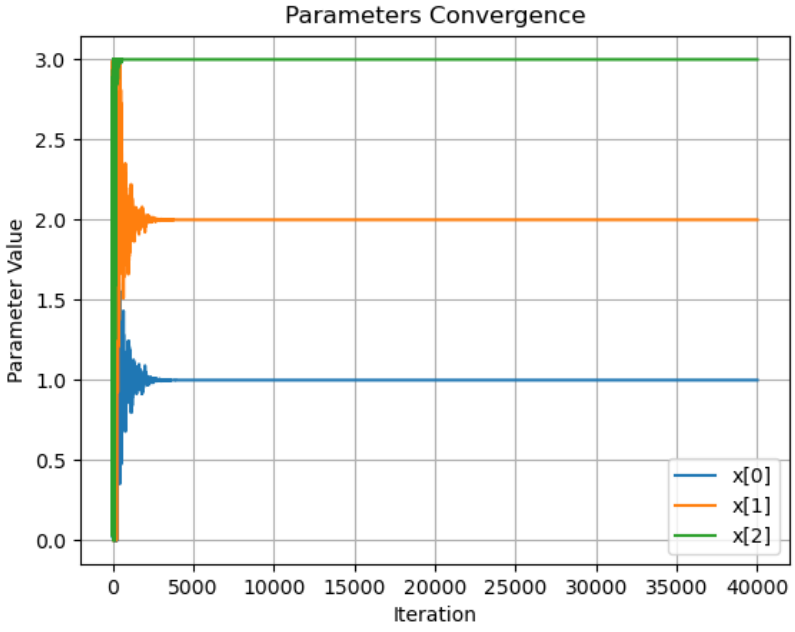

Figura 3: Convergência dos parâmetros da função de otimização para o teste da implementação do PSO

2.2 Otimização do controlador do robô seguidor de linha

2.2.1 Histórico de Otimização

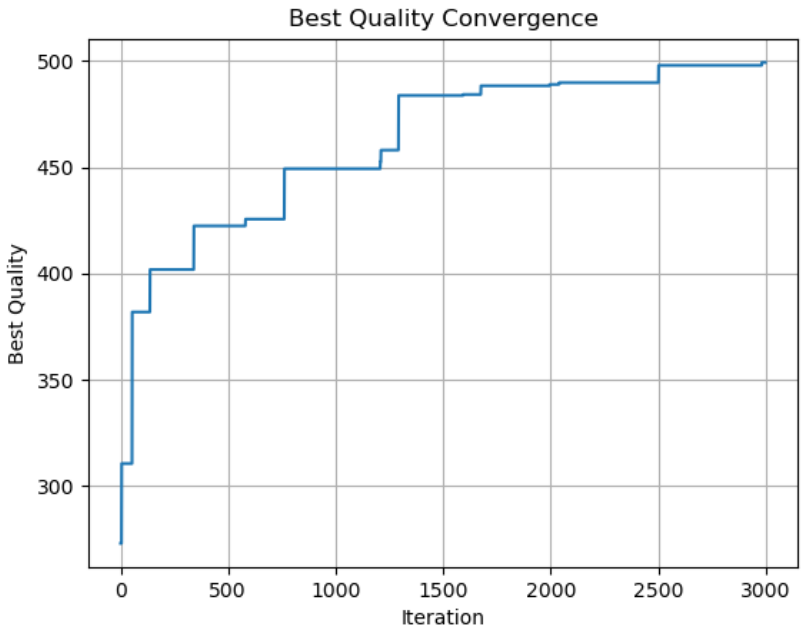

Figura 4: Convergência do melhor valor da função de otimização para a otimização do robô seguidor de linha

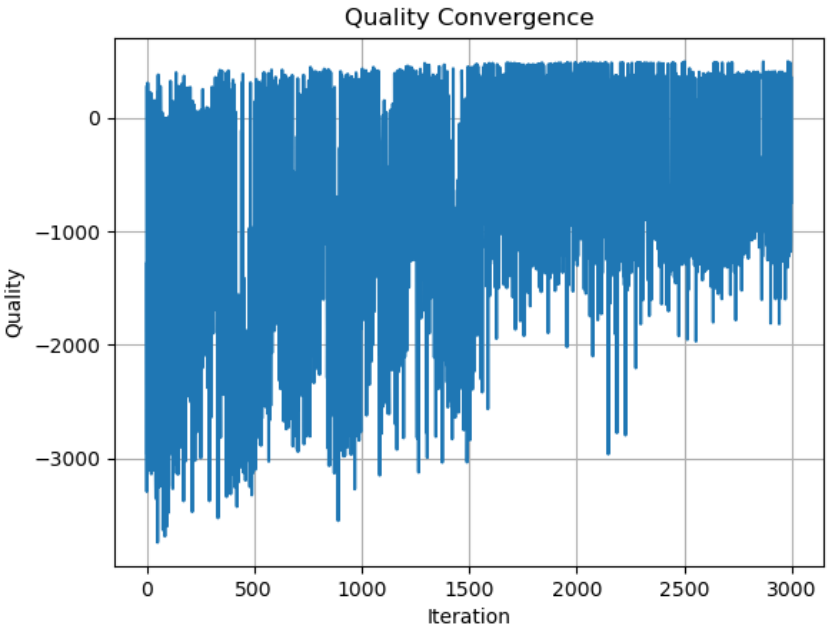

Figura 5: Convergência do valor da função de otimização para a otimização do robô seguidor de linha

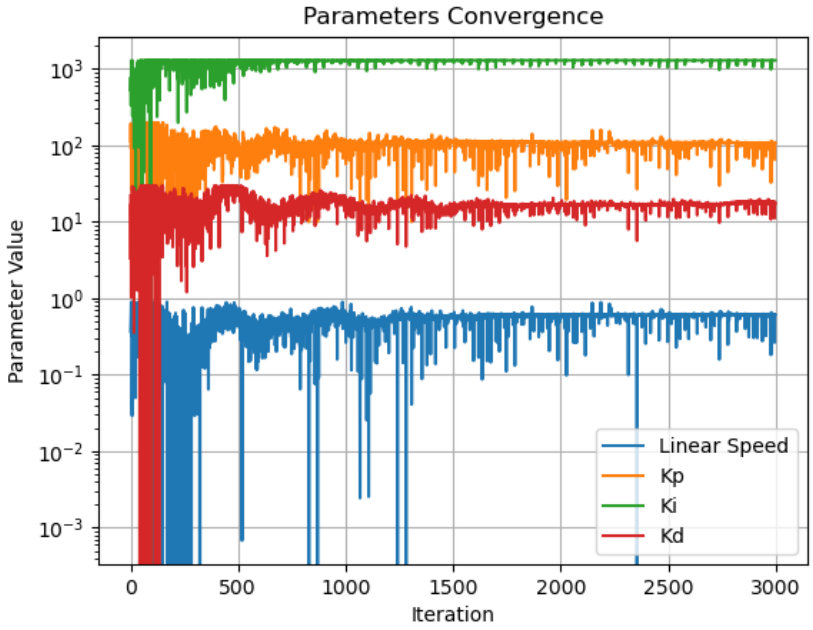

Figura 6: Convergência dos parâmetros da função de otimização para a otimização do robô seguidor de linha

2.2.2 Melhor Trajetória Obtida Durante a Otimização

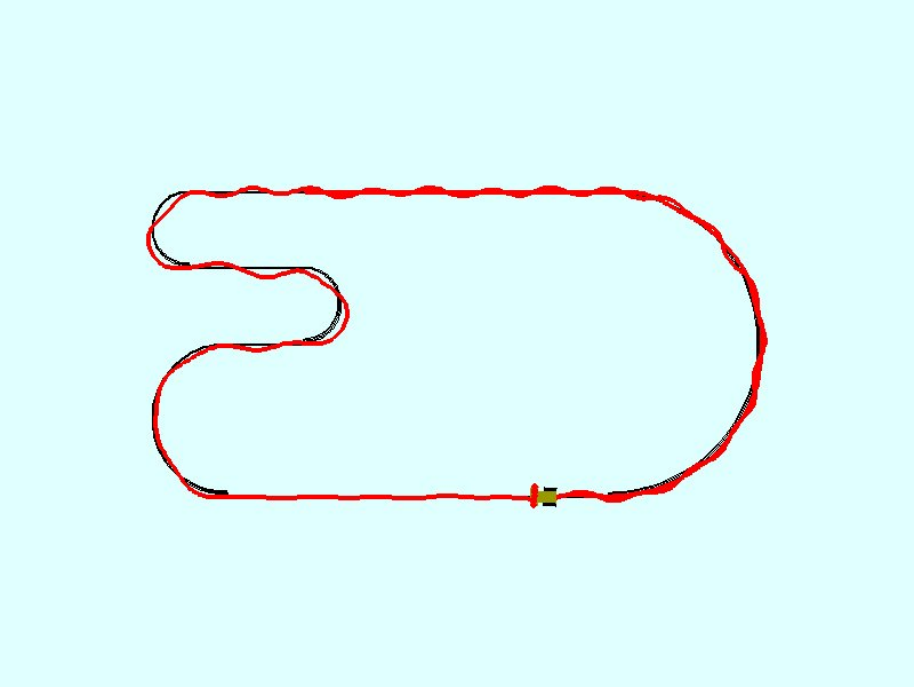

Figura 7: Solução encontrada para o melhor percurso do robô pela otimização

3 Discussão sobre o observado durante o processo de otimização

Para a otimização do controlador do robô seguidor de linha, fixaram-se os hiperparâmetros do PSO e ajustavam-se os parâmetros w e o valor definido para o erro quando a linha não era detectada, conforme a equação de recompensa, vista na Equação 2.

$$reward_k = v_k \cdot < r_k, t_k > -w \cdot |e_k| \tag{2}$$

O valor do erro nesse caso deve ser alto o bastante para penalizar quando o robô não segue a linha, mas não deve ser demasiadamente alto para não enviesar os valore da função de qualidade. Ajustando-se os valores, notou-se que para valores baixos do erro, um dos parâmetros k_i não convergia ao longo de 3000 iterações e o robô não conseguia realizar a primeira curva, executando um arco com muita curvatura. Para erros muito grandes, a convergência era lenta e 3000 iterações não eram suficientes. Por fim, atribuindo-se o erro para o valor 15 quando a linha não era detectada e $w = 0,6$, geraram-se as imagens anteriores.