Relatório do Lab 4 de CT-213

Trabalho 4 – Otimização com Métodos Baseados em População

Controlador de um robô *Line Follower* com *Particle Swarm Optimization* (PSO)

Aluno:

Bruno Costa Alves Freire

Turma:

T 22.4

Professor:

Marcos Ricardo Omena de Albuquerque Máximo

Data:

15/04/2019

Instituto Tecnológico de Aeronáutica – ITA
Departamento de Computação

1. Implementação do PSO

A implementação do algoritmo *Particle Swarm Optimization* foi feita de modo ligeiramente diferente dos códigos apresentados em aula. Para os propósitos deste laboratório, a função objetivo do problema (também referida como *função de qualidade*) não pode ser passada como parâmetro para uma implementação iterativa do algoritmo, pois é proveniente de uma simulação.

Para adequar o algoritmo a este formato, foi criada a classe que atuará como gerenciadora do algoritmo, `ParticleSwarmOptimization`. Essa classe armazena as *partículas* do algoritmo, os hiperparâmetros, e as informações de melhor posição global do algoritmo. Os métodos `get_best_position` e `get_best_value` são a interface com o *script* "cliente" do algoritmo para comunicar as melhores soluções encontradas globalmente pelo PSO.

As partículas têm sua estrutura definida na classe `Particle`, onde são inicializadas por meio do método construtor conforme o código visto em aula.

O funcionamento do algoritmo é implementado de modo que a informação da função de qualidade é proveniente do script cliente, com o qual a comunicação se dá por meio dos métodos `get_position_to_evaluate`, e `notify_evaluation`. Através do primeiro, é passada a posição de uma partícula para o script cliente, que irá avalia-la com a função qualidade, e então retornar o resultado dessa avaliação para a classe do PSO por meio do segundo método.

Dado que o PSO funciona com várias partículas, e a cada geração todas as partículas devem ser avaliadas antes de se fazer a atualização das mesmas, foi necessário implementar um esquema de *rodízio* entre as partículas. Através de uma variável `particle_schedule`, é guardado o índice da próxima partícula a ser avaliada, o qual é atualizado toda vez que uma partícula recebe sua avaliação. Quando o índice do rodízio se iguala ao número de partículas, é hora de reiniciá-lo e realizar a atualização de geração.

Dessa forma, a implementação do método `notify_evaluation` dará conta de atualizar o registro de melhor posição da partícula que foi avaliada, e em seguida atualizar o registro de melhor posição global. Como a variável de melhor global pode ser acessada diretamente por este método, não há a necessidade de se manter um registro do melhor a cada geração, pois essa informação é imediatamente incorporada ao melhor global. Além disso, quando o contador do rodízio atinge o número de partículas, o mesmo é zerado, e invoca-se o método `advance_generation`, que realiza a atualização das posições e velocidades de todas as partículas, conforme o código apresentado em aula.

A implementação foi testada por meio do *script* `test_pso.py`, utilizando uma função quadrática simples como função de qualidade. Os resultados constam nas figuras 1, 2 e 3.

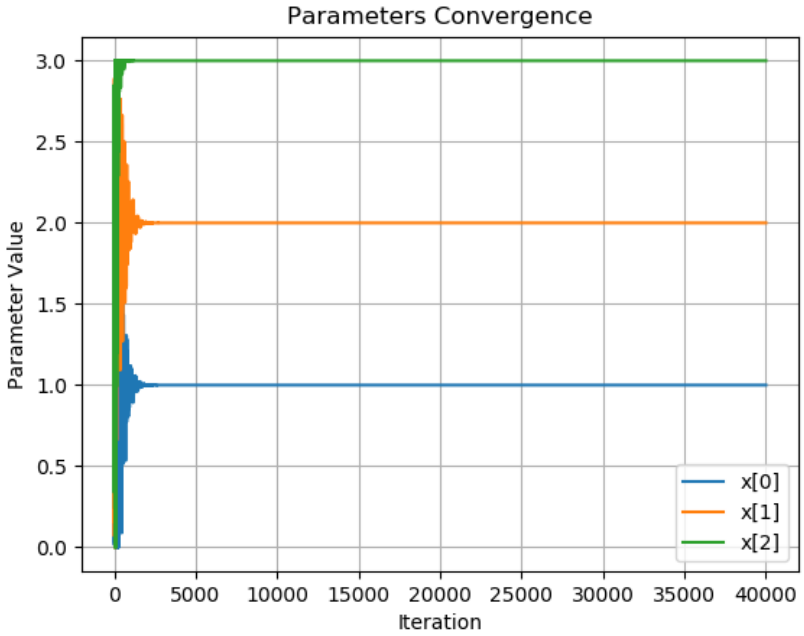

Figura 1: Convergência dos parâmetros para a função teste.

A função a ser otimizada nesse teste era

$$f(x_1, x_2, x_3) = -[(x_1 - 1)^2 + (x_2 - 2)^2 + (x_3 - 3)^2],$$

cujo único e óbvio máximo global é o ponto $(x_1, x_2, x_3) = (0, 0, 0)$, com valor de qualidade igual a 0. Pela figura 1, podemos ver que o PSO encontrou as coordenadas do ótimo global corretamente.

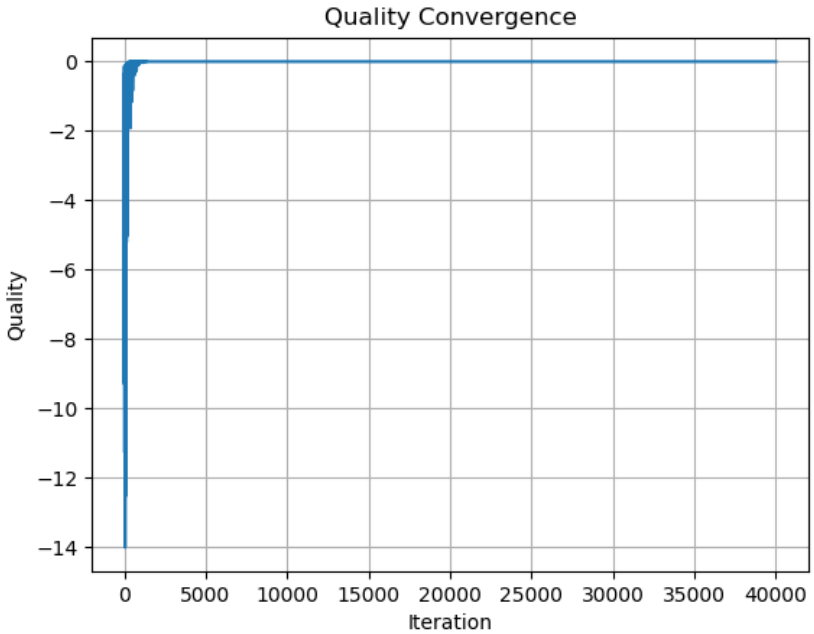

Figura 2: Convergência do valor de qualidade das partículas para o ótimo da função teste.

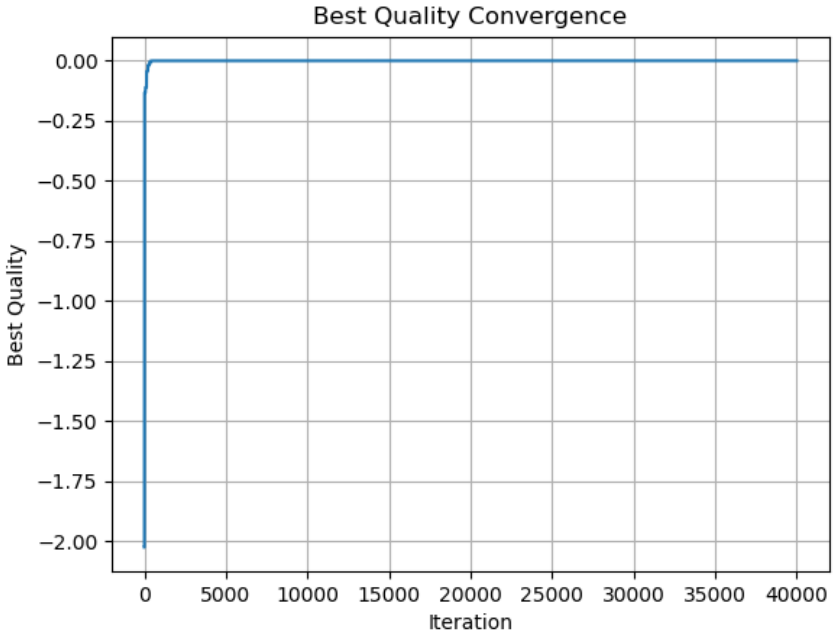

Figura 3: Convergência do melhor valor global do PSO para o ótimo da função teste.

Das figuras 2 e 3 podemos observar a convergência do valor de qualidade das posições encontradas pelo PSO para o valor ótimo da função de qualidade.

Após verificado o funcionamento do algoritmo para este teste, vamos ao objetivo do laboratório: otimizar os parâmetros do controlador de um robô seguidor de linha.

2. Treinamento do robô *Line Follower*

Para treinar o robô *Line Follower* em seu nobre objetivo, precisamos definir a função de qualidade que o PSO tentará otimizar. No arquivo `simulation.py`, são definidos os métodos a serem invocados pela classe `Simulation`, tais como atualizar a simulação, desenhar o estado atual, etc. Dentre estes, o método de interesse para o PSO é a função `evaluate`, a qual irá aplicar, a cada instante da simulação, a função de qualidade do desempenho do robô. Essa função, quando avaliada para uma sessão de treino inteira, consiste no somatório:

$$\sum_{k=1}^{N} v_k \cdot \langle r_k, t_k \rangle - w \cdot |e_k|$$

Neste somatório, o índice *k* é relativo ao número da iteração numa sessão de treino com *N* iterações. No termo geral, v_k é a velocidade linear do robô, r_k e t_k são respectivamente os vetores tangentes à trajetória do robô e à curva do caminho, (logo o produto interno indica a concordância entre a direção do robô e o caminho a ser seguido), e_k é o erro em relação à linha e w é um fator de peso para ponderar os objetivos de completar o caminho rapidamente e fazê-lo sem fugir muito da linha.

Quando o robô está distante da linha, ele não é capaz de mensurar seu erro. Por isso, quando a linha não é detectada, deve ser colocado um valor padrão para $|e_k|$, que no caso foi de *100 vezes o valor máximo* para o erro detectável. O objetivo era "punir" com severidade soluções que levassem o robô a se afastar muito do caminho.

Uma vez implementados todas as funções e classes relacionadas ao algoritmo, vamos ao treinamento. O robô foi deixado treinando durante um total de 126241 iterações. Como foram utilizadas 40 partículas no PSO, isso dá um total de 3156 gerações do algoritmo. A convergência total do algoritmo não foi observada, no entanto, a partir da 40000ª iteração, já eram observadas soluções boas muito próximas do melhor global obtido ao longo de todo o treinamento. Podemos entender melhor esse aspecto por meio das figuras 4 e 5, que mostram a convergência da melhor qualidade global encontrada pelo PSO, e da qualidade de cada sessão de treinamento, respectivamente.

Os parâmetros ótimos encontrados foram listados na tabela 1, e a convergência do PSO para encontra-los é mostrada na figura 6. Por fim, a figura 7 mostra o trajeto executado pela solução ótima.

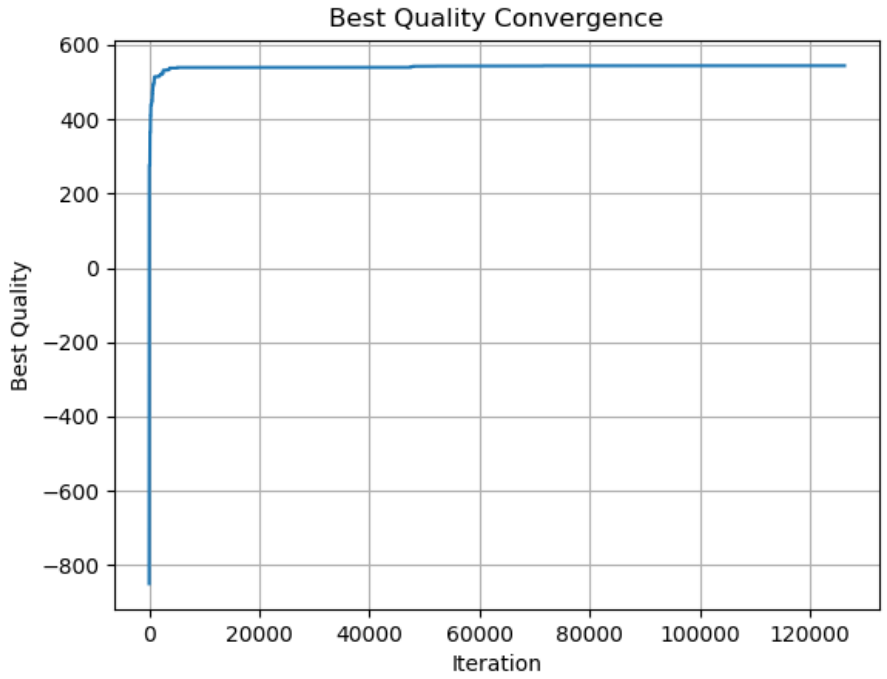

Figura 4: Convergência da qualidade da melhor solução global do PSO para o treinamento do robô seguidor de linha.

Pela figura 4 podemos constatar que a qualidade da melhor solução evoluiu pouco após cerca da 3000ª iteração, em torno da qual o algoritmo parece ter encontrado uma solução satisfatória. Uma pequena evolução pode ser constatada por volta da 47000ª iteração. Nesse ponto, foi observado que num momento quase todas as partículas exibiam um comportamento muito parecido, muito próximo da então melhor solução. Alguns milhares de iterações depois, a maior parte das partículas apresentavam soluções de qualidade menor, sendo notados várias sessões de treino consecutivas em que o robô fugia do trajeto em algum ponto. Apesar dessa constatação, os dados da figura 5 mostram que durante todo o treinamento, a qualidade das partículas sempre teve uma amplitude considerável, com pouca evolução após as primeiras 3000 iterações.

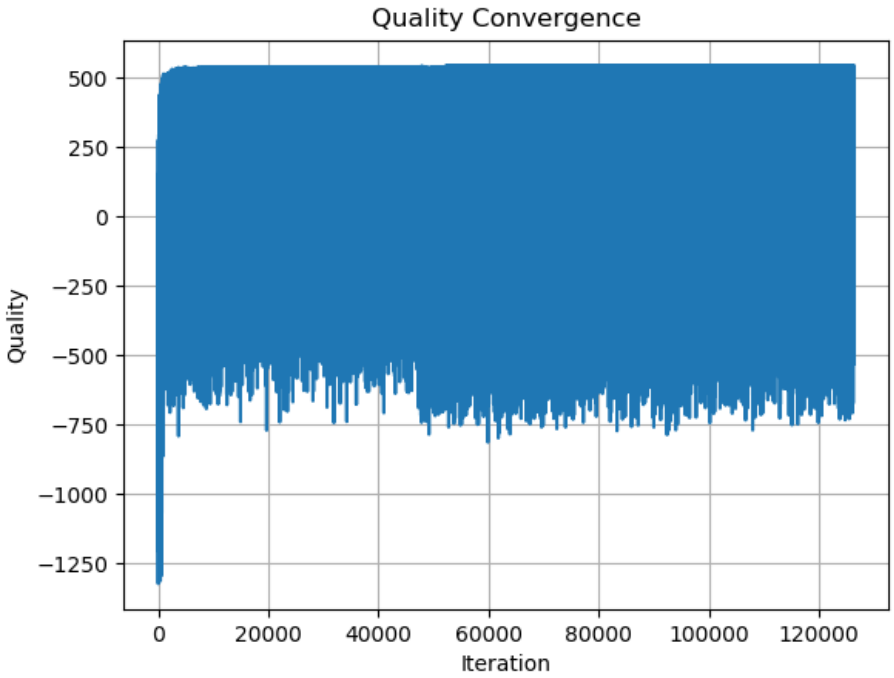

Figura 5: Histórico da qualidade das partículas durante o treinamento do robô seguidor de linha.

É possível notar um aumento sutil na amplitude da qualidade das partículas após cerca de 47000 iterações. Ou seja, as piores soluções ficaram ainda piores, embora a melhor solução tenha ficado um pouco melhor.

Tabela 1: Valores ótimos para os parâmetros encontrados pelo PSO ao longo do treinamento do robô *Line Follower*

Parâmetro	**Valor**
Velocidade Linear	0.677840
Ganho Proporcional (K_P)	135.625910
Ganho Integral (K_I)	810.913490
Ganho Derivativo (K_D)	16.136381
Qualidade	544.0238375699511

Na tabela 1 podemos ver os valores finais otimizados dos parâmetros de controle do robô. Nota-se um valor relativamente alto para o ganho integral, em contraposição a um valor baixo para o ganho derivativo. A convergência dos parâmetros para estes valores é mostrada na figura 6.

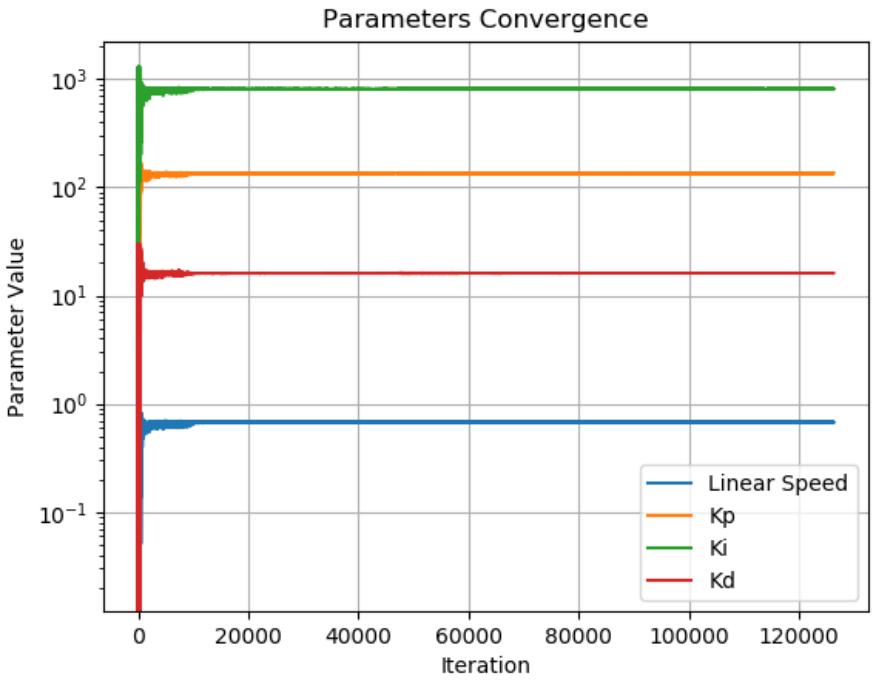

Figura 6: Convergência dos parâmetros do controlador do robô seguidor de linha, encontrados pelo PSO.

Da figura 6 podemos notar que a convergência dos parâmetros se deu próxima à 10000ª iteração, havendo pouca variação dali em diante. É curioso observar que, mesmo havendo pouca variação nos parâmetros a partir de um determinado número de iterações, ainda havia uma discrepância notável entre as qualidades das partículas. Isso mostra que a função de qualidade do treinamento do robô possui uma geometria complicada (além de uma natureza estocástica intrínseca), certamente merecendo o termo “mal comportada”.

O aspecto da melhor solução encontrada pelo PSO, caracterizada pelos parâmetros na tabela 1, é registrado na figura 7.

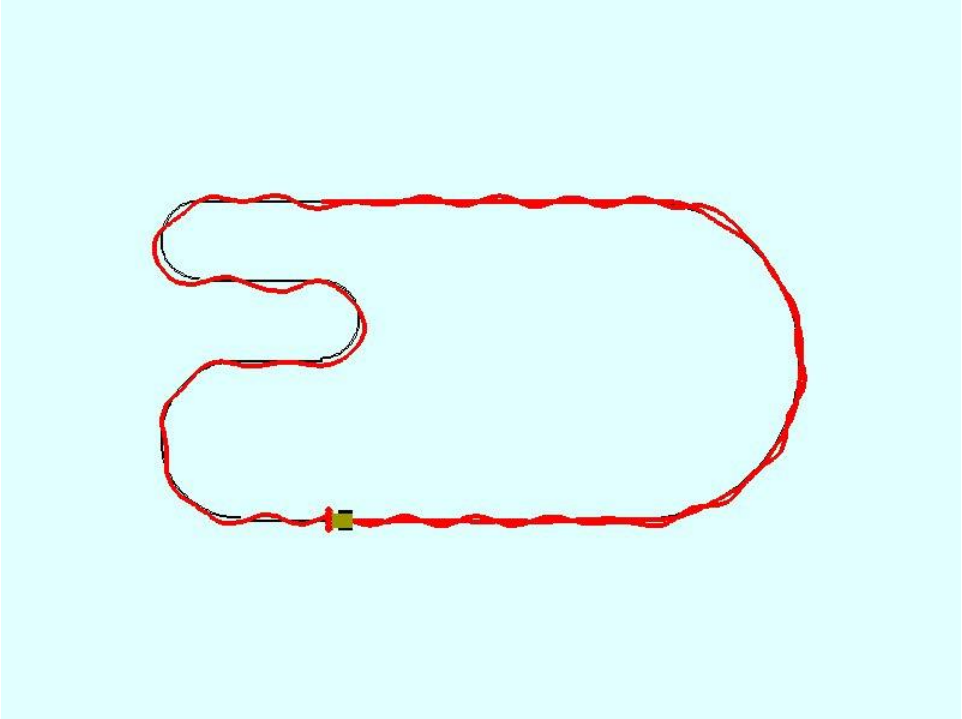

Figura 7: Trajetória resultante da melhor configuração de parâmetros encontrada pelo PSO para o robô seguidor de linha.

A partir da figura 7, podemos inferir que o robô conseguiu completar uma volta inteira e mais um trecho de metade do circuito no tempo de uma sessão (15 segundos). É possível observar que, na primeira passagem pelo trecho reto de baixo, o robô oscila bastante, no entanto, é capaz de realizar as curvas sem se perder do caminho. Na segunda passagem pelo trecho curvo longo, o robô executou a curva com perfeição, mantendo inclusive boa estabilidade ao passar pela segunda vez no trecho reto de baixo.

Contudo, não dá pra saber se o robô seria capaz de completar o circuito quantas vezes fosse necessário mantendo um bom desempenho. Para isso, seria necessário treiná-lo com sessões mais longas, ou esperar o PSO encontrar uma solução que desse mais voltas com mais estabilidade. Dado que foram rodadas mais de 120000 iterações, ao longo das quais a qualidade não aumentou muito significativamente, vamos tomar essa solução como boa o suficiente.