tecnia revista de educação, ciência e tecnologia do ifg

Volume 9, Número 2, 2024

Submetido 23/09/2024. Aprovado 07/10/2024
Avaliação: revisão duplo-anônimo

# Otimização do movimento de um robô móvel por meio de algoritmos evolucionários

OPTIMIZATION OF MOBILE ROBOT MOVEMENT USING EVOLUTIONARY ALGORITHMS
OPTIMIZACIÓN DEL MOVIMIENTO DE UN ROBOT MÓVIL MEDIANTE ALGORITMOS EVOLUTIVOS

**Willian Martins Leão**
Centro Federal de Educação Tecnológica de Minas Gerais (Cefet/MG)
willian.leao@cefetmg.br

**Fabiana Alves Pereira**
Universidade Federal de Uberlândia (UFU)
apereira.fabiana@gmail.com

**Resumo**

O presente trabalho aborda a aplicação de algoritmos evolucionários com o intuito de encontrar a rota e os parâmetros ótimos de um controlador de movimento, visando aproveitar melhor a dinâmica de um robô móvel. O estudo foi realizado por meio do simulador CoppeliaSim, que utiliza um controle de movimento para deslocar um robô móvel em um cenário com 16 pontos de passagem distribuídos de maneira não uniforme, sujeito a quedas e movimentos indesejados. Dessa forma, os parâmetros do controlador e a rota são otimizados a partir da combinação de um algoritmo genético e um algoritmo de evolução diferencial, com o objetivo de fazer com que o robô móvel passe nos 16 pontos no menor tempo possível. Essa combinação de técnicas de inteligência artificial propiciou uma variedade maior de soluções e, consequentemente, um tempo de deslocamento otimizado, além de eliminar soluções que resultavam em quedas ou movimentos orbitais do robô móvel.

**Palavras-chave:** algoritmo de evolução diferencial; algoritmo genético; controle; robótica evolucionária.

**Abstract**

This work explores the use of evolutionary algorithms to determine the optimal route and parameters for a motion controller, aiming to maximize the dynamic performance of a mobile robot. The simulations were carried out in Coppelia Sim, where the controller guides the robot through a scenario consisting of 16 waypoints, unevenly distributed and prone to falls or erratic movements. By combining a genetic algorithm with a differential evolution algorithm, the controller parameters and route were optimized to ensure the robot navigates all 16 points in the shortest possible time. This integration of artificial intelligence techniques not only expanded the range of potential solutions but also optimized travel time, while discarding those that result in falls or unstable robot behavior.

**Keywords:** differential evolution algorithm; genetic algorithm; control; evolutionary robotics.

**Resumen**
Este trabajo investiga la aplicación de algoritmos evolutivos para hallar la ruta y los parámetros óptimos de un controlador de movimiento, con el objetivo de optimizar el rendimiento dinámico de un robot móvil. El estudio se llevó a cabo utilizando el simulador CoppeliaSim, donde se emplea un control de movimiento para desplazar un robot móvil en un escenario con 16 puntos de paso, distribuidos de manera no uniforme, sujeto a caídas y movimientos no deseados. De este modo, los parámetros del controlador y la ruta se optimizan mediante la combinación de un algoritmo genético y un algoritmo de evolución diferencial para que el robot móvil atraviese los 16 puntos en el menor tiempo posible. Esta combinación de técnicas de inteligencia artificial proporcionó una mayor variedad de soluciones y, en consecuencia, un tiempo de desplazamiento optimizado, además de descartar soluciones que condujeron a caídas o movimientos inestables del robot móvil.

**Palabras-Clave:** algoritmo de evolución diferencial; algoritmo genético; control; robótica evolutiva.

# Introdução

A intenção de elaborar ferramentas que atuem de maneira autônoma e regular no meio em que vivemos, superando as limitações humanas, coloca a robótica como expectativa de paradigma nas tarefas industriais e domésticas do tempo presente.

Assim, a inteligência artificial tem desempenhado um papel de destaque em um novo mundo que demanda maior integração de informações e a busca por soluções eficientes para diversos problemas. Um dos ramos da inteligência artificial é a computação evolucionária, que utiliza técnicas inspiradas na evolução genética e em outros modelos de sistemas biológicos. A união dessa área com a robótica dá origem a um novo campo em pleno crescimento: a robótica evolucionária.

A Robótica Evolucionária (RE) é um método para o desenvolvimento de robôs autônomos. Diferentes abordagens de RE são apontadas em Iba (2008), o que inclui tópicos que vão desde sistemas de multi-robôs, com comportamento emergente de uma evolução ou interação, até o planejamento evolucionário de robôs e a otimização de parâmetros de controladores aplicados em robótica.

Em Tang *et al.* (2020) foi desenvolvido um planejamento de caminho de multi-robôs usando otimização autoadaptativa por enxame de partículas. A proposta era gerar um caminho livre de obstáculos e prevenir colisões de cada robô, além de garantir a chegada simultânea de cada robô ao destino, enquanto o comprimento total do caminho do sistema multi-robô é minimizado.

Os parâmetros do modelo do robô são otimizados por meio do Algoritmo Genético (AG) em Sami e Momen (2019), visando reduzir o consumo de energia de multi-robôs utilizados na tarefa de limpeza. Outros trabalhos focam nos controladores de robôs móveis, com seus ganhos sendo otimizados por algoritmos genéticos em Shim e Kim (1995), Ito, Iwasaki e Matsui (2001), e por programação genética em Walker e Messom (2002).

Diferentes algoritmos evolutivos são comparados em Parque e Miyashita (2020) para busca de um ajuste adequado de curvas suaves para trajetórias de robôs móveis para avaliar diferentes modos de inicialização da população, pressão de seleção, exploração e exploração durante a amostragem.

Em Freitas, Cohen e Guimarães (2023) são implementados dois tipos de planejadores de rotas baseados em splines que levam em conta a prevenção de obstáculos, o comprimento e a suavidade das rotas. Para tal, é implementado um algoritmo diferencial adaptativo, que utiliza uma redução linear da população a partir de um tamanho inicial da população, além de controlar de forma adaptativa os parâmetros

do algoritmo de evolução diferencial clássico – fator de escala e a probabilidade de cruzamento –, o que propicia uma melhoria na convergência do algoritmo.

Um algoritmo de campo potencial artificial baseado em evolução diferencial foi implementado em Zhang *et al.* (2024), a fim de otimizar a trajetória de veículos guiados automatizados para tarefas de inspeção e esterilização em ambientes internos para aliviar a escassez crítica de profissionais de saúde obtendo melhorias. Os resultados alcançados consistiram em melhorias na redução do comprimento do caminho como garantia de uma distância segura entre os obstáculos em comparação aos método de Abordagem por Janelas Dinâmicas (Dynamic Window Approach – DWA) e o algoritmo de Árvores Aleatórias de Exploração Rápida (Rapid Exploration Random Tree – RRT).

Na robótica evolucionária há duas abordagens: evolução off-line e evolução on--line. Na primeira, o algoritmo evolucionário é aplicado antes do período de funcionamento do robô. Já a segunda abordagem consiste em aplicar um algoritmo evolucionário durante a operação do robô.

Assim, o presente trabalho consiste em aplicar algoritmos evolucionários off-line, no intuito de encontrar a rota e os parâmetros ótimos de um controlador de movimento, visando reduzir o tempo de deslocamento entre os pontos de passagem de um robô móvel.

## Problema

Dado um espaço plano de 5,0 × 5,0 m, o robô deve atravessar 16 pontos de passagem, distribuídos irregularmente ao longo do cenário. Esse robô está sujeito a duas condições indesejadas: queda e movimento de órbita. A queda diz respeito a algum movimento brusco que o robô possa fazer, lançando-o para fora do cenário, como é mostrado na Figura 1. Nessa figura, a linha preta contínua, que parte da origem (centro do cenário), representa o caminho percorrido pelo robô, enquanto os pontos amarelos indicam as passagens já atravessadas, e os pontos pretos representam os pontos ainda não alcançados pela trajetória.

Uma razão para esse inconveniente são os altos ganhos do controlador que geram velocidades maiores e movimentos bruscos que podem ou não implicar em queda do robô móvel para fora do cenário. Outro motivo é a combinação de parâmetros do controlador que podem implicar em um movimento mais aberto, de tal maneira que haja a queda a depender da rota executada.

O segundo problema é o movimento de órbita ilustrado na Figura 2. O movimento de órbita é indesejado, pois o robô não consegue se aproximar de uma distância mínima em que possa ser considerado como ter cruzado o ponto de passagem, o que implicaria em ficar em tempo indefinido nessa região.

A combinação de parâmetros de controlador pode resultar ou não no movimento de órbita do robô móvel sobre um ponto de passagem, dependendo da rota executada. Portanto, a rota ótima não implica naquela com menor distância entre os pontos, como é dado pelo problema do caixeiro viajante, mas consiste na rota que possibilita o melhor aproveitamento da dinâmica do robô móvel para executar, no menor tempo possível, o deslocamento entre os pontos de passagem.

Assim, a proposta de algoritmo evolutivo aqui empregado visa obter o menor tempo de deslocamento de um robô móvel entre os 16 alvos, de forma a garantir que não ocorra movimento de órbita e queda.

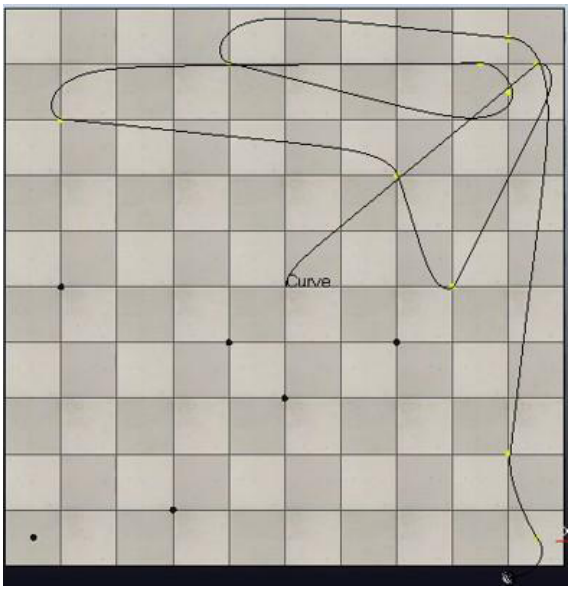


**Figura 1 - Movimento de queda de um robô móvel para fora do cenário**
Fonte: Elaborada pelos(as) autores(as).

## Controlador

O controlador adotado foi proporcional com bias (peso fixo), que corresponde a uma velocidade de rotação das rodas do robô móvel para avanço até um ponto de passagem. O controlador proporcional realiza o controle de orientação do robô móvel em direção de um ponto de passagem desejado. O ângulo desejado para o robô móvel é calculado em todos os instantes pela equação:

$$\theta_s = \arctan \frac{y_p - y_r}{x_p - x_r} \quad (1)$$

Nesse caso, sendo $y_p$ a posição em y do robô; $y_r$ a posição em y do ponto de passagem desejado; $x_p$ a posição em x do robô; $x_r$ a posição em x do ponto de passagem desejado; $\theta_s$ o ângulo de orientação desejado para o robô.

Já o erro angular, ou seja, o erro na orientação do robô, é calculado por:

$$\theta_e = \theta_s - \theta_r \quad (2)$$

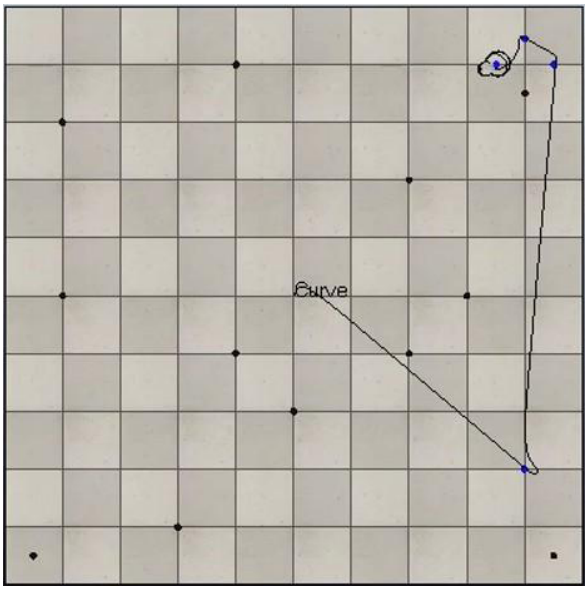


**Figura 2 - Movimento de órbita de um robô móvel sobre um ponto de passagem**
Fonte: Elaborada pelos(as) autores(as).

Assim, o $\theta_e$ corresponde ao erro de orientação do robô, entre $(-\pi, \pi]$; $\theta_s$ o ângulo de orientação desejado para o robô; $\theta_r$ o ângulo de orientação atual do robô. O controlador de movimento adotado no robô móvel é dado por:

$$\omega_{dir} = \omega_{av} + k_p \cdot \theta_e \tag{3}$$

$$\omega_{esq} = \omega_{av} - k_p \cdot \theta_e \tag{4}$$

Sendo $\omega_{dir}$ e $\omega_{esq}$ as velocidades angulares das rodas direitas e esquerdas respectivamente; $\omega_{av}$ a velocidade angular fixa das rodas para o avanço do robô móvel; $k_p$ o ganho proporcional; $\theta_e$ o erro do ângulo de orientação.

## Proposta aplicada

A proposta deste trabalho é encontrar a solução que combine uma rota e os parâmetros do controlador, resultando no menor tempo de movimento de passagem de todos os alvos, partindo o robô da origem do cenário. Para isso, será utilizado um AG para otimizar a rota e um algoritmo de evolução diferencial autoadaptativo (self adaptive differential evolution – SaDE) para otimizar os ganhos do controlador de movimento.

O algoritmo de SaDE utilizado foi proposto em Qin e Suganthan (2005), e tem como característica um método para definir a probabilidade de aplicação entre dois métodos de mutação, baseado no número indivíduos mutados permanecidos (sucesso) e não permanecidos na próxima geração (fracasso). O método é dado por:

$$p_1 = \frac{ns1 \cdot (ns2 + nf2)}{ns2 \cdot (ns1 + nf1) + ns1 \cdot (ns2 + nf2)} \tag{5}$$

Nesse contexto, sendo ns1 e ns2 o número de indivíduos dentro de um período de aprendizagem que obtiveram sucesso em permanecer na próxima geração e que são resultado de um método de mutação 1 e de um método de mutação 2 respectivamente; nf1 e nf2 são o número de indivíduos dentro de um período de aprendizagem que foram descartados para a próxima geração, sendo resultado de um método de mutação 1 e de um método de mutação 2 respectivamente. Esses métodos 1 e 2 são métodos quaisquer que foram definidos pelo usuário do algoritmo autoadaptativo.

O método autoadaptativo da Equação 5 foi empregado de três formas. Uma para definir qual método de mutação do Algoritmo de Evolução Diferencial (AED), usado para otimizar o controlador, obtém maior taxa de sucesso. O segundo caso é para definir qual método de mutação do AG, usado na otimização das rotas, oferece maior taxa de sucesso para aumentar o *fitness*. Já o terceiro caso consiste em gerenciar se há maior probabilidade de haver modificação das rotas ou dos parâmetros do controlador, favorecendo a alteração que tem maior taxa de sucesso de otimizar a população.

Para tanto, define-se a soma de todos os sucessos e fracassos para a mutação de rotas como ns1 e nf1, e a soma de todos os sucessos e fracassos para a mutação de parâmetros do controlador como ns2 e nf2. Dessa maneira, o programa nunca realiza modificações de rota e parâmetros do controlador ao mesmo tempo, o que poderia indicar uma falsa contribuição de uma das modificações.

O método de seleção por sobrevivência foi aplicado nas rotas e nos parâmetros do controlador, com a diferença que o método foi utilizado com o intuito de conservar os maiores *fitness*, o que o difere do método clássico, em que o problema é de minimização.

## Definição dos hiperparâmetros em comum

Em ambas as abordagens (AG e AED), o tamanho da população foi de dez indivíduos, sendo cada um composto de uma rota e dois parâmetros do controlador. O critério de parada do programa foi 200 gerações. O cálculo de *fitness* adotado foi:

$$f_i = \frac{1000}{1 + t_i} \tag{6}$$

Assim, sendo $f_i$ e $t_i$ o *fitness* e o tempo gasto para que o robô móvel passe nos 16 pontos de passagem, dada uma rota e uma combinação de parâmetros do controlador definidos pelo indivíduo $i$ da população do algoritmo evolucionário aplicado. Com essa definição de *fitness*, o problema de otimização é de maximização.

Os casos em que há problemas de queda do robô móvel para fora do cenário ou em que há movimentos de órbitas sobre pontos de passagem foram penalizados com a definição de tempo $t_i$ igual a 240 segundos. Portanto, todas as rotas e os parâmetros de controlador são definidos com o mesmo valor de *fitness*.

## Definição dos hiperparâmetros do SaDE

Os hiperparâmetros do SaDE foram definidos com um intervalo de busca para as velocidades de avanço de 5 a 40 rad/s e um intervalo de busca para o ganho proporcional de 10 a 60 s−1. Esses valores foram escolhidos de forma a garantir velocidades baixas, que não necessariamente contribuiriam para o melhor tempo, e velocidades altas, nas quais o robô móvel certamente cairia para fora do cenário. Dessa forma, o algoritmo SaDE encontrará o limite para obter um menor tempo de deslocamento do robô sem gerar problemas de movimentação. Os métodos utilizados para a mutação são os:

$$\text{DE/best/1: } V_{i,G} = X_{best,G} + F \cdot (X_{r1,G} - X_{r2,G})$$

$$\text{DE/rand/1 : } V_{i,G} = X_{r1,G} + F \cdot (X_{r2,G} - X_{r3,G})$$

Nesse caso, sendo que, para cada vetor alvo $X_{i,G}$ de uma geração G, está associado um vetor mutante $V_{i,G} = \{\omega_{avi,G}, k_{pi,G}\}$, no qual $\omega_{avi,G}$ e $k_{pi,G}$ são a velocidade de avanço e o ganho proporcional de um indivíduo $i$ de uma geração $G$ respectivamente.

O primeiro método de mutação garante uma velocidade maior de convergência, e o segundo uma diversificação maior nas variáveis dos indivíduos da população. As probabilidades de mutação entre os métodos foi definida pelo algoritmo autoadaptativo proposto por Qin e Suganthan (2005), conforme a Equação 5, com probabilidade inicial de 50% para ambos os métodos.

O fator de evolução ($F$) é escolhido aleatoriamente, dentro de uma distribuição gaussiana dada por N ($\mu = 0,7$ $e$ $\sigma = 0,3$) $\in (0, 2]$ para cada variável no indivíduo da população, garantindo uma diversidade maior da população quanto aos parâmetros do controlador.

O método de recombinação utilizado é a recombinação discreta e a probabilidade de recombinação para SaDE ($p_r$, $sade$) foi definida por uma distribuição gaussiana N ($\mu = 0,5$ $e$ $\sigma = 0,1$), que tem seu valor atualizado a cada cinco gerações.

Nas soluções que obtiveram parâmetros do controlador fora do domínio de busca, aplicou-se o operador de reflexão.

## Definição dos hiperparâmetros do AG

Os hiperparâmetros do AG são representados pela mutação, sendo cada indivíduo uma lista de número inteiros no intervalo de 1 a 16, em que cada número corresponde a ponto de passagem no cenário do problema dado na Seção 2. Em outras palavras, cada lista representa uma rota pela qual o robô móvel passará, sendo que o cruzamento não intencional do robô móvel em um ponto de passagem não é considerado para a execução do movimento.

A partir deste ponto do texto, o uso da palavra “rotas” será aplicado em vez de “AG”, com o objetivo de enfatizar para qual hiperparâmetro ou análise cada procedimento está sendo realizado. No entanto, os métodos de mutação e recombinação que serão apresentados para as rotas derivam da teoria de algoritmos genéticos (AGs). Os métodos de mutação para as rotas são de inserção e inversão para representação por permutação, conforme disposto em Eiben e Smith (2015).

As probabilidades de mutação entre os métodos foi definida pelo método do algoritmo autoadaptativo proposto por Qin e Suganthan (2005), conforme definido no início da Seção 4.1, com probabilidade inicial de 50% para ambos os métodos. Assim,

também o método de seleção por sobrevivência típico do AED foi utilizado para garantir que as melhores soluções fossem conservadas.

O propósito, portanto, foi aproveitar as características típicas da abordagem de computação evolucionária com o AG, sendo o último já bem aceito para a representação por permutação. Assim, no intuito de explorar mais o mecanismo do AED, o AG aqui aplicado teve sua ordem de execução alternada, passando a realizar a mutação de cada indivíduo da população primeiramente e, em seguida, de acordo com a probabilidade de recombinação das rotas, as soluções são recombinadas com uma rota de solução escolhida da população por torneio de cinco indivíduos. Isso garante que a recombinação seja sempre realizada com uma rota mutada, aumentando a diversidade das rotas para soluções.

O método de recombinação utilizado foi do tipo por ordem, de acordo com Eiben e Smith (2015). A probabilidade de recombinação é dada por uma distribuição gaussiana N ($\mu = 0, 2$ e $\sigma = 0, 1$), que tem seu valor atualizado a cada  gerações. Esse valor reduzido foi escolhido para que haja pouca troca de informações entre os indivíduos e a diversidade na população no decorrer das gerações.

## Simulação

No intuito de validar a proposta, no software CoppeliaSim® foi simulada a operação do robô móvel Lumibot, posicionado em um cenário 5,0 x 5,0 m com 16 pontos de passagem fixos e distribuídos de forma irregular, conforme pode ser visualizado na vista superior apresentada na Figura 3.

O teste realizado foi configurado com o passo de tempo de simulação de 50 ms. O robô tem informação da sua posição e do ponto de passagem a ser alcançando a todo instante.

O desenvolvimento deste trabalho foi realizado por meio das linguagens Lua e Python. Em Lua, foram desenvolvidos os algoritmos por meio de scripts embarcados no software CoppeliaSim: os scripts de customização e scripts filhos (ou child scripts), a partir de API Regular em um child script sem thread, e os códigos em Python utilizaram uma Application Programming Interface (API) Remota do CoppeliaSim. A estrutura do software CoppeliaSim é resumida na Figura 4.

O primeiro foi utilizado para executar ações independentemente da simulação estar ou não em andamento, como a reinicialização do cenário, o gerenciamento da medição dos tempos de realização e da definição das rotas e dos parâmetros do controlador de cada teste (indivíduo) dentro de uma geração. Já o segundo associa um algoritmo para um objeto do cenário, nesse caso é o controlador proporcional com bias no robô móvel.

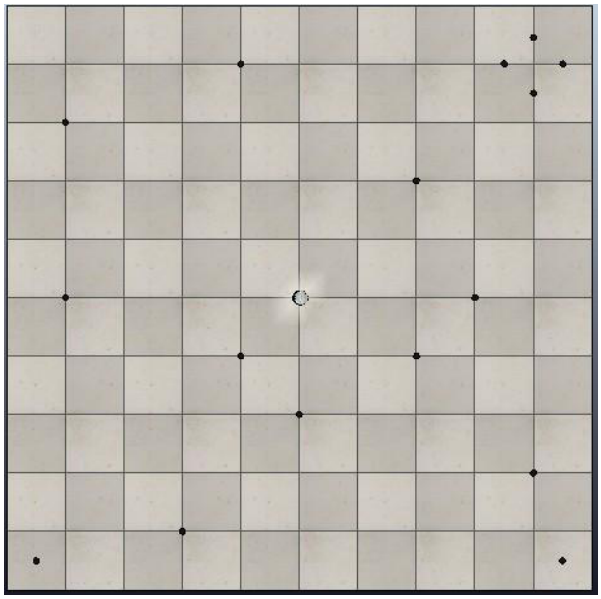

**Figura 3 - Cenário de simulação do robô móvel**
Fonte: Elaborada pelos(as) autores(as).

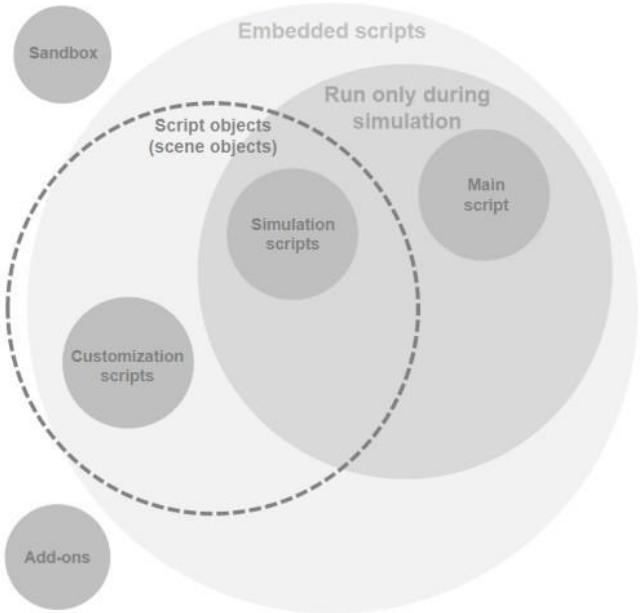


**Figura 4 - Estrutura de organização do software CoppeliaSim**
Fonte: Coppelia Robotics.

O algoritmo de evolução diferencial proposto foi elaborado em Python, que comunica com o CoppeliaSim por meio de uma API remota. A adoção do algoritmo de evolução ser escrito em Python deveu-se ao fato de as funções disponíveis em Lua no CoppeliaSim serem muito limitadas.

O comportamento do robô móvel é esquematizado pelo autômato finito apresentado na Figura 5. O robô móvel partirá da origem, seguindo até o primeiro ponto de passagem. Uma vez que ele atinja uma distância mínima de $dist_{min}$=0,03m, parte para o segundo ponto de passagem, e assim sucessivamente, até passar pelos 16 pontos de passagem. Após isso, o cenário é reiniciado, com o robô móvel na origem

novamente, e ele começa a completar a nova rota e os novos parâmetros do controlador, definidos pela solução do algoritmo evolutivo.

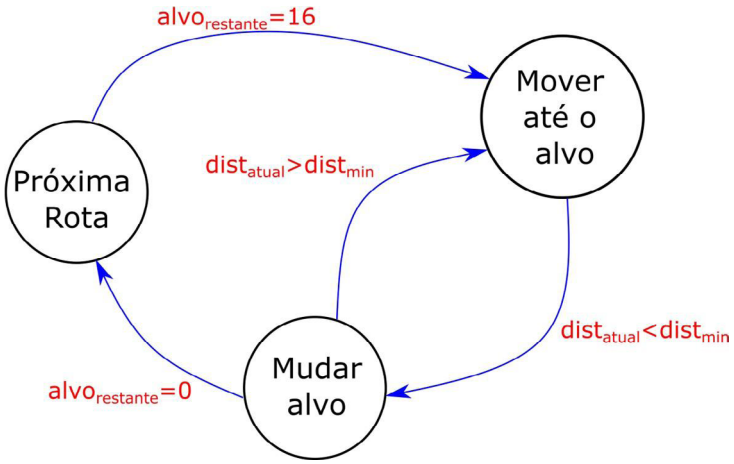


**Figura 5 - Autômato finito da rotina do robô móvel**
Fonte: Elaborada pelos(as) autores(as).

## Resultados e análises

As Figuras 6 e 7 apresentam a evolução do tempo gasto e a evolução do *fitness*, respectivamente, para deslocar nos 16 pontos de passagem definidos pelo problema.

A Figura 6 indica que o pior tempo (maior tempo) é igual a 240 segundos entre a 25ª e 50ª geração, o que corresponde a problemas de queda ou de órbita. Esse inconveniente é eliminado ao longo das gerações por meio de mudanças nas rotas ou nos parâmetros do controlador combinados com a seleção por sobrevivência.

Na Figura 7, há a mudança contínua da média do *fitness*, logo há sempre mudanças de rotas e/ou parâmetros do controlador que garantem melhorias nas soluções no decorrer das gerações. Essa característica permite concluir que o período de aprendizagem adotado – de dez gerações – é adequado para a análise do emprego do método autoadaptativo nas probabilidades de mutação aplicadas aos algoritmos evolutivos.

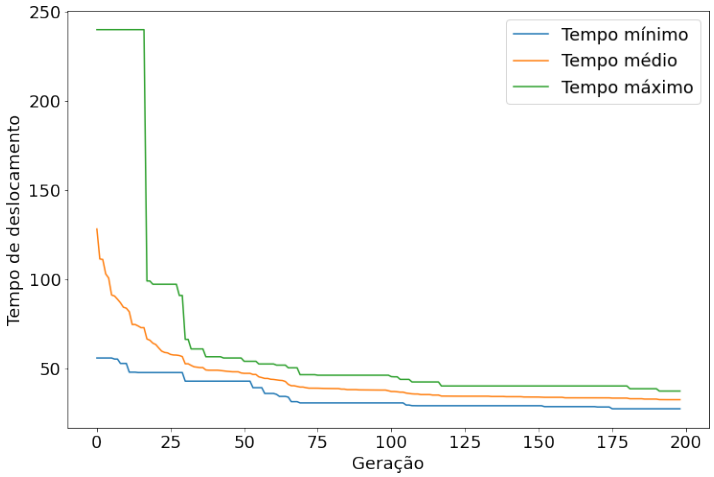


**Figura 6 - Evolução do tempo de deslocamento**
Fonte: Elaborada pelos(as) autores(as).

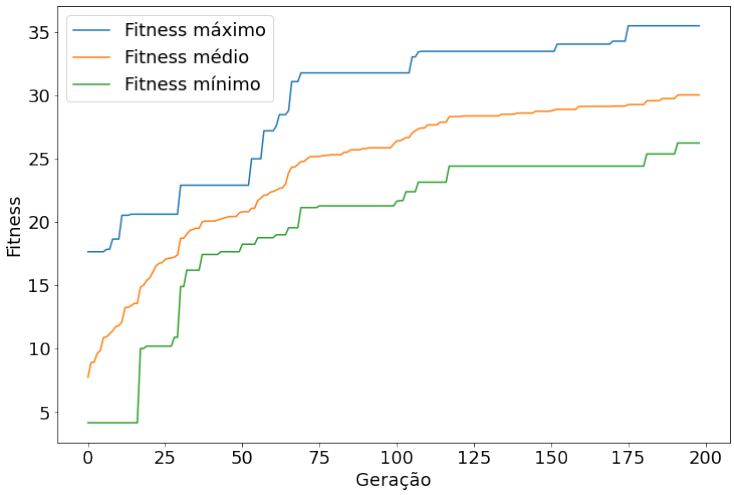


**Figura 7 - Evolução do *fitness***
**Fonte:** Elaborada pelos(as) autores(as).

A Figura 8 indica alternância de maior probabilidade entre os métodos de mutação dos parâmetros do controlador, portanto não há uma predominância entre os métodos DE/rand/1/bin e DE/best/1/bin ao longo das atualizações das probabilidades.

Entretanto, em relação ao emprego do método autoadaptativo de Qin e Suganthan (2005), verificou-se que este atribui indefinições matemáticas às probabilidades calculadas quando há casos de dominância de um dos métodos – não há sucesso por parte de um dos métodos – e em situações nas quais nenhum método apresenta sucesso dentro do período de aprendizagem. Por conseguinte, a indefinição matemática que advém da atualização das probabilidades ocorre por causa da ausência de fracassos e sucessos de um ou de ambos os métodos.

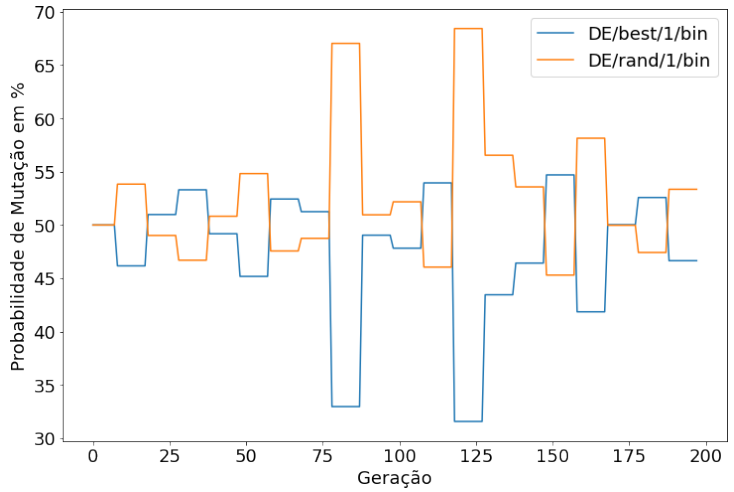


**Figura 8 - Evolução da probabilidade de mutação para os parâmetros do controlador**
Fonte: Elaborada pelos(as) autores(as).

A solução foi adotar duas estratégias: se o denominador da Equação 5 for igual a zero e seu numerador for maior que zero, então a probabilidade $p_1$ é igual a 1; se o denominador e o numerador da Equação 5 forem iguais a zero, a probabilidade $p_1$ é igual a 0,5, como se o método de adaptação da probabilidade se reiniciasse. Em resumo, quando há dominância de um método, este permanece como único por um

período de aprendizagem, sendo reequilibrado na próxima atualização, a fim de que haja oportunidade para que ambos os métodos otimizem novas soluções.

As Figuras 9, 8 e 10 mostram a evolução da probabilidade, respectivamente, para os tipos de métodos de mutação dos parâmetros do controlador, para os tipos de métodos de mutação das rotas e para modificação por rotas ou parâmetros do controlador.

Na otimização da rotas dada pela Figura 9, vemos que na 140ª geração houve a dominância do método de mutação por inserção. No entanto, quando observamos a Figura 10, notamos que houve também a dominância de modificação por variação dos parâmetros do controlador, logo, entre a 130ª a 139ª geração, não houve nenhuma mutação nas rotas que obteve sucesso, sendo que, no período de atualização na 160ª geração, houve uma reinicialização da probabilidade de modificação de rotas e modificação dos parâmetros. Além disso, na 170ª a 179ª geração houve novamente uma dominância na modificação dos parâmetros do controlador. É interessante ressaltar que, ao reiniciar as probabilidades, surgiram sucessos com a modificação nas rotas.

A explicação desse fenômeno é que as modificações nas rotas apresentam decrescimento de oportunidade para melhorar o tempo de deslocamento do robô móvel, pois, ao longo das gerações menores, há chances de que alguma modificação resulte em uma melhoria. Contudo, as modificações nas rotas são responsáveis pela geração de grandes mudanças nos *fitness*, o que pode ser observado por meio da Figura 7, em que há uma grande variação do *fitness* da melhor solução (maior *fitness* entre a 170ª a 179ª geração), que corresponde ao instante da Figura 10 em que houve maior probabilidade (maior sucesso) de modificações de rotas.

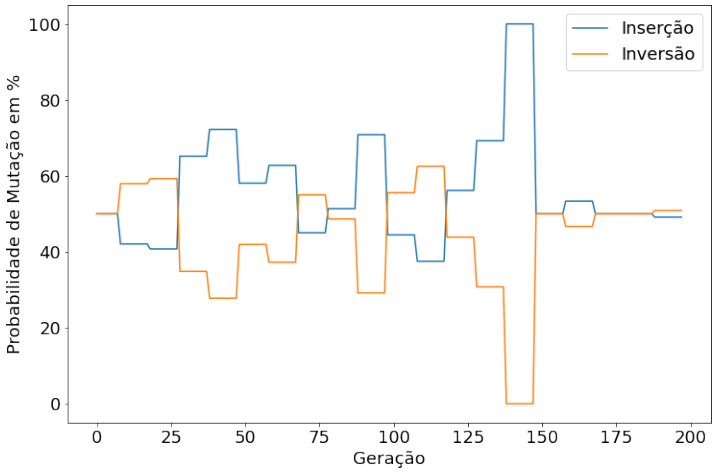


**Figura 9 - Evolução da probabilidade de mutação para as rotas**
Fonte: Elaborada pelos(as) autores(as).

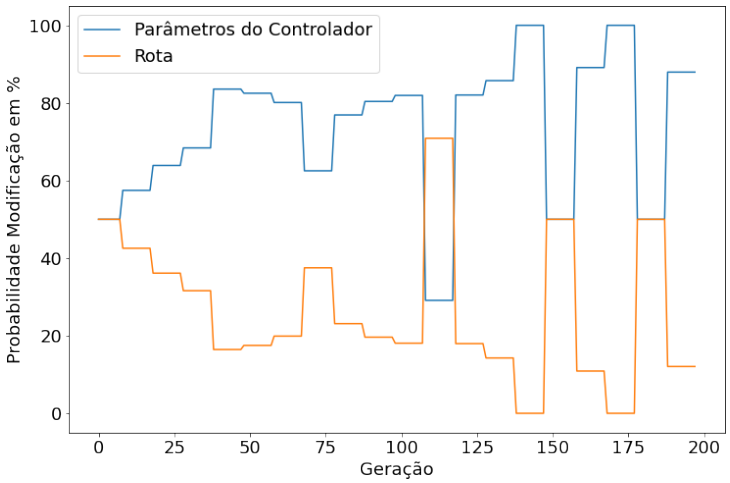


**Figura 10 - Evolução da probabilidade de mudança de rota e mudança nos parâmetros do controlador.**
Fonte: Elaborada pelos(as) autores(as).

## Considerações finais

O uso do método de seleção por sobrevivência, típico do algoritmo DE, garantiu a sobrevivência das melhores rotas e dos melhores parâmetros do controlador, o que possibilitou, ao longo das gerações, a eliminação de soluções que levavam a quedas ou movimentos de órbitas do robô móvel.

O uso do método autoadaptativo do SaDE permite direcionar a escolha dos melhores métodos de mutação tanto para as rotas quanto para a definição dos parâmetros do controlador. Além disso, a extensão do mesmo método autoadaptativo, aplicado ao sucesso e às falhas gerais da modificação de rotas e parâmetros, permite definir qual dessas mudanças favorece a otimização da solução do problema apresentado ao longo das gerações.

Por fim, o trabalho permitiu apontar uma limitação e propor uma estratégia para superar situações de dominância de probabilidade, nas quais o método autoadaptativo atribui indefinições matemáticas às probabilidades.

## Referências

EIBEN, A.; SMITH, J. *Introduction to Evolutionary Computing*. [*S. l.*]: Springer Berlin Heidelberg, 2015. (Natural Computing Series).

COPPELIA Robotics. *Scripts*. Disponível em: https://manual.coppeliarobotics.com/en/scripts.htm.

FREITAS, E. J. de R.; COHEN, M. W.; GUIMARÃES, F. G. *An autonomous mobile robot path planner using spline curves and differential evolution*. Recife: IEEE, 2023.

SHIM, H., S.; KIM, J. Robust control of non-holonomic wheeled mobile robot based on evolutionary programming for optimal motion. *In*: Proceedings of 1995 IEEE

International Conference on Evolutionary Computation. [*S.l.: s.n.*], 1995. v. 2, p. 625–630 vol.2.

IBA, H. *Frontiers in Evolutionary Robotics*. 1. ed. [S. l.]: Tech Education and Pu- blishing, 2008.

ITO, K.; IWASAKI, M.; MATSUI, N. Ga-based practical compensator design for a motion control system. *IEEE/ASME Transactions on Mechatronics*, [*s. l.*], v. 6, n. 2, p. 143-148, 2001.

PARQUE, V.; MIYASHITA, T. Smooth curve fitting of mobile robot trajectories using differential evolution. *IEEE Access*, [*s. l.*], v. 8, p. 82855-82866, 2020.

QIN, K.; SUGANTHAN, P. Algoritmo de evolução diferencial autoadaptável para otimização numérica. *In*: CONGRESSO IEEE SOBRE COMPUTAÇÃO EVOLUTIVA, 2005, Edimburgo, Reino Unido. *Anais* […]. Edimburgo, Reino Unido: CEC, 2005. v. 2, p. 1785-1791.

SAMI, T. A.; MOMEN, S. Optimization of energy consumption in swarms of robots. *In*: INTERNATIONAL CONFERENCE OF COMPUTER SCIENCE AND RENEWABLE ENERGIES, 2019, [*s. l.*]. *Anais* […]. [*S. l.*]: ICCSRE, 2019. p. 1-5.

TANG, B.; XIANG, K.; PANG, M.; ZHANXIA, Z. Multi-robot path planning using an improved self-adaptive particle swarm optimization. *International Journal of Advanced Robotic Systems*, [*s. l.*], v. 17, n. 5, 2020.

WALKER, M.; MESSOM, C. H. A comparison of genetic programming and genetic algorithms for auto-tuning mobile robot motion control. *In*: PROCEEDINGS First IEEE Internati- onal Workshop on Electronic Design, Test and Applications 2002. [*S. l.: s. n.*], 2002. p. 507-509.

ZHANG, C.; ZHAI, B.; DING, L.; LIANG, Z. Research on path planning of inspection robots for epidemic prevention. *In*: 5TH INTERNATIONAL CONFERENCE ON MECHA- TRONICS TECHNOLOGY AND INTELLIGENT MANUFACTURING, 2024, [*s. l.*]. *Anais* […]. [*S. l.*]: ICMTIM, 2024. p. 493-497.