CISC455/851 Automated Tuning of Trajectory-tracking Model Predictive Controller for Autonomous Vehicle

JINGXUE JIANG, Queen's University, Canada
MAX OSWIN, Queen's University, Canada
YANAN WANG, Queen's University, Canada

ACM Reference Format:
Jingxue Jiang, Max Oswin, and Yanan Wang. 2023. CISC455/851 Automated Tuning of Trajectory-tracking Model Predictive Controller for Autonomous Vehicle. 1, 1 (April 2023), 12 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

A automated tuning method based on genetic algorithm is presented to find proper parameters for an model predictive controller. The objective of the trajectory-tracking controller is to force a vehicle to follow a predetermined reference trajectory precisely while ensuring stability and convergence.Simulation result of the tracking performance of the GA-tuned MPC controller is provided.

2 PROBLEM BACKGROUND:AN INTRO TO MODEL PREDICTIVE CONTROL

Model predictive control (MPC) is a discrete-time multi-variable control architecture. At each control interval, an MPC controller uses an internal model to predict future vehicle behavior. Based on this prediction, the controller computes optimal control actions to send to the vehicle. The objective function of a trajectory tracking MPC controller usually has the following formulation:

$$J = X_{t+N}^T \cdot S \cdot X_{t+N} + \sum_{j=1}^{N} X_{t+j}^T \cdot Q \cdot X_{t+j} + \Delta u_{t+j}^T \cdot R \cdot \Delta u_{t+j} \tag{1}$$

where:

the matrices S,Q,R are the penalty matrices that influences the performance of the controller.

The goal of the MPC controller is to find optimal control commands to the vehicle by minimizing the objective function J above. In this formulation, vehicle's deviation from the reference trajectory and the change in control inputs are minimized. In essence, the MPC will try to follow the path as precisely as possible while using as little control effort as possible.

Authors' addresses: Jingxue Jiang, Queen's University, Kingston, Canada, 17jj33@queensu.ca; Max Oswin, Queen's University, Kingston, Canada, 22bvg@queensu.ca; Yanan Wang, Queen's University, Kingston, Canada, 17yw206@queensu.ca.

3 PROBLEM DESCRIPTION

3.1 Motivation

MPC controller received significant recognition and popularity among self-driving companies because of its ability to improve vehicle responsiveness while maintaining passenger comfort and safety.

However, an improperly-tuned MPC controller can leads to convergence failure, oscillation and unstable vehicle motion. In order to follow the reference trajectory precisely while ensuring passenger safety and comfort, the parameters of the MPC controller needs to be carefully chosen to satisfy the performance objectives. To tackle this issues, an automated tuning algorithm is developed with error minimization and passenger safety in mind. The tracking behaviour of two improperly-tuned MPC controller is illustrated in the below graphs, the black path represent the trajectory the vehicle is suppose to follow and the red line represent the vehicle's actual trajectory.

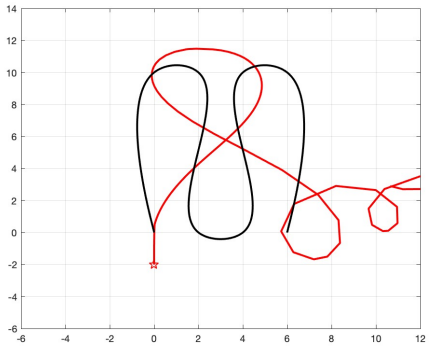

Fig. 1. Example of convergence failure: vehicle motion using an improperly tuned MPC controller

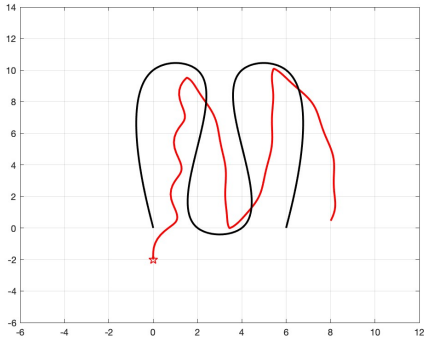

Fig. 2. Example of convergence failure: vehicle motion using an improperly tuned MPC controller

3.2 Problem Setup

The goal of this project is to use genetic algorithm to tune the penalty matrices S,Q,R, and the horizon N of the following objective function:

$$J = X_{t+N}^T \cdot S \cdot X_{t+N} + \sum_{j=1}^{N} X_{t+j}^T \cdot Q \cdot X_{t+j} + \Delta u_{t+j}^T \cdot R \cdot \Delta u_{t+j} \tag{2}$$

The termination penalty S is a 3×3 matrix, the state penalty Q is also a 3×3 matrix, the control penalty R is a 2×2 matrix, and the horizon N is an positive integer number.

The penalty matrices S,Q,R and horizon N found by the genetic algorithm will dedicate the trajectory tracking performance of the vehicle. The objective of the proposed EA is to find the set of penalty matrices that minimize the error between the vehicle's current position and the path's position. In addition, other user-defined functional objectives such as minimizing the convergence time can be explicitly Incorporated into EA's fitness function to alter vehicle's tracking behaviour. However, the primary objective of a path following controller should be make the vehicle following the path as precisely as possible.

4 LITERATURE REVIEW

4.1 Topic: Evaluation of tracking performance

In this subsection, we investigated standard evaluation method for trajectory-following controller, with the goal to develop a reasonable fitness function. Several performance measures are collected and analyzed from multiple academic papers.

The most obvious and the most frequently used measure is position error. In majority of the academic papers, the position error along the x-axis and y-axis over time are used to demonstrate the effectiveness of the trajectory-tracking controller. Some researchers used distance error to evaluate the tracking performance of the controller. [Ros+09] evaluate the tracking performance by setting a maximum error (absolute value of the difference between the desired and the real trajectories, once the mobile robot has reached the geometric predefined path) of 2cm in the circular trajectory and 6cm in the eight-shaped path.These errors are very small compared to the distance between wheel axes (40cm)

4.2 2. Review of Initialisation and Crossover methods for floating point values

We took inspiration in our evolutionary algorithm's design from literature as well. In particular, we were keen to find methods which maintained the diversity of the population during early stages of evolution. [BB03] found that a Heuristic Uniform Crossover method was well suited to an exploring the breadth of search space, during an investigation into gene coding for parametric optimisation. As such, we implemented and trialled the method as part of our solution.

4.3 3. Choice of method of selection

We initially chose to implement and test tournament selection, which is a comparison (matching) between several individuals based on the fitness value of each individual. The larger the number of individuals competing, the greater the selection pressure will be. We endorsed the idea of Fuzzy Adapting tournament selection in our study, which means adjusting selection pressure according to the diversity of the population (diverse population) and accomplishing balanced development and exploration and works well on small problems. [Pra+20] This approach can be flexibly adapted, so it is suitable as an initial implementation.

5 EA DESIGN

5.1 Population Initialization

5.1.1 Random Initialization. The choice of population initialization method is random initialization. Each individual in the population is consist of the [**t**]hree diagonal matrices Q,R,S respectively and horizon N:

$$Q = \begin{bmatrix} q11 & 0 & 0 \\ 0 & q22 & 0 \\ 0 & 0 & q33 \end{bmatrix} \quad where \;\; 0 < Q_{ij} < 100$$

$$R = \begin{bmatrix} r11 & 0 \\ 0 & r22 \end{bmatrix} \quad where \;\; 0 < R_{ij} < 100$$

$$S = \begin{bmatrix} s11 & 0 & 0 \\ 0 & s22 & 0 \\ 0 & 0 & s33 \end{bmatrix} \quad where \quad 0 < S_{ij} < 10$$

$$N = n \quad where \quad n \in \mathbb{Z}^+ \quad and \quad 0 < n <= 10$$

The constraints imposed on the matrices and horizon are to ensure the stability and convergence of the MPC controller. Since aged-based elitism survivor method is used, the generated individuals are populated into a First-In-First-Out queue for age tracking purposes.

5.1.2 Random Power Initialization. A slight variation of true randomisation was also experimented with. Under this method, random floating-point values within the range $[-5, 1]$ are generated and applied as exponents to a base of 10. The results are subsequently applied as values within the diagonal matrices. This process aims to create diverse ratios between the parameters of candidate solutions.

5.1.3 Heuristic initialization. A heuristic initialization method is developed to generate dissimilar solutions to make the initial population more diverse. It utilizes ideas include eigenvalues of the matrix and the dynamic among the variables to generate the initial population.

5.2 Termination

The genetic algorithm will terminate once the iteration counts exceeds a user-defined threshold. In practice, the average fitness of the population stabilizes after approximately 40 iterations.

5.3 Fitness Evaluation

The vehicle's deviation from the reference trajectory is quantified by the following mathematical formula:

$$e_{dev} = e_x + e_y + e_\theta + e_{cte} + e_v + e_\omega$$

where:

e_{dev}: is the deviation error
e_x: is the horizontal error
e_y: is the vertical error
e_θ: is the heading error
e_{cte}: is the cross track error
e_v: is the deviation from the reference linear velocity
e_ω: is the deviation from the reference angular velocity

The deviation error e_{dev} captures how well the controller can force the vehicle to follow the reference trajectory and the negation of this deviation error forms the fitness function:

$$fitness = -k * e_{dev} \quad where \quad k \in \mathbb{R}^+$$

5.4 Parent Selection

5.4.1 tournament selection. Tournament selection is used to select parents to enter the mating pool and replacement is not allowed. In practice, the tournament size is set to 20 percent of the population size and the mating pool size is set to 10.

5.4.2 stochastic Universal Sampling Selection. Stochastic universal sampling not only maintains population diversity but also avoids the problem of premature convergence. In this approach, the cumulative fitness of the population is divided into equally spaced slots and a parent is selected from each slot using multiple pointers.

5.5 Crossover

5.5.1 Wright's crossover. One choice of crossover method we explored is Wright's Heuristic crossover, which takes into account the fitness of the parents for generating the offspring. This crossover method hones in on excelling individuals, leading the search process towards the most promising zones.

5.5.2 Heuristic Uniform Crossover. Heuristic Uniform Crossover (HUX) is great at maintaining a diverse population. Two children are initialised as clones of their respective parents. Next, the number of different genes between these two children is calculated, before half of these differences are randomly swapped. We refer to this method as 'random-HUX'. To mitigate the chaotic nature of this crossover method, we experimented with only returning the fittest of the two offspring. This makes the method much more effective, at the cost of increased execution time, due to the doubled fitness evaluations. Furthermore, this shifts HUX away from exploration of the search space, and towards exploitation. We refer to this version as 'best-HUX'.

5.6 Mutation

The choice for mutation is an altered version of Non-uniform Mutation. If an individual is selected, all parameters are mutated. Each parameter is a randomly incremented by a small value, drawn from a Gaussian distribution. The mutated value is bounded to a range dependant on the parameter's respective matrix. Furthermore, the effect of mutation is inversely weighted by the generation count; as the generations increase, mutations make smaller changes to a selected individual. We chose this approach in order to hone in on top-performing individuals when approaching the end of generation.

5.7 Survivor Selection

5.7.1 Age-based Elitism Selection. We select survivors based on a combination of age-based and elitism, and we will compare the experimental data with a fitness-based strategy. First, we replace the oldest individuals with offspring while retaining the best individuals in the population. We also tried replacing the oldest ones with all the better children. This approach sometimes yields better results but is more likely to converge too quickly with some uncertainty.

5.7.2 Fitness-based Selection. In fitness-based replacement, individuals with high fitness produce offspring that replace those parents with low fitness. This approach helps improve the population's overall fitness over time. In addition, a fitness-based replacement can help guide the search toward better solutions and improve the algorithm's efficiency.

6 EA RESULTS

This section is divided into three subsections: Relevant Terminology, Functional requirements and results. First subsection discusses the definitions of several error measures and their relationship with the tracking performance. The second subsection provides a list of functional requirements the EA must satisfy. The third subsection presents the final results and EA's compliance with the proposed functional requirements.

6.1 Relevant Terminology

To quantitatively analyze the tracking performance of the MPC controller tuned by the designed genetic algorithm, following performance measures are used:

(1) cross track error (lateral error)
(2) angular error
(3) x-component of the position error
(4) y-component of the position error
(5) convergence time

The cross track error is the lateral distance error measured from the centre of the front axle to the nearest point on the path. An illustration of the cross track error is given in the below figure.

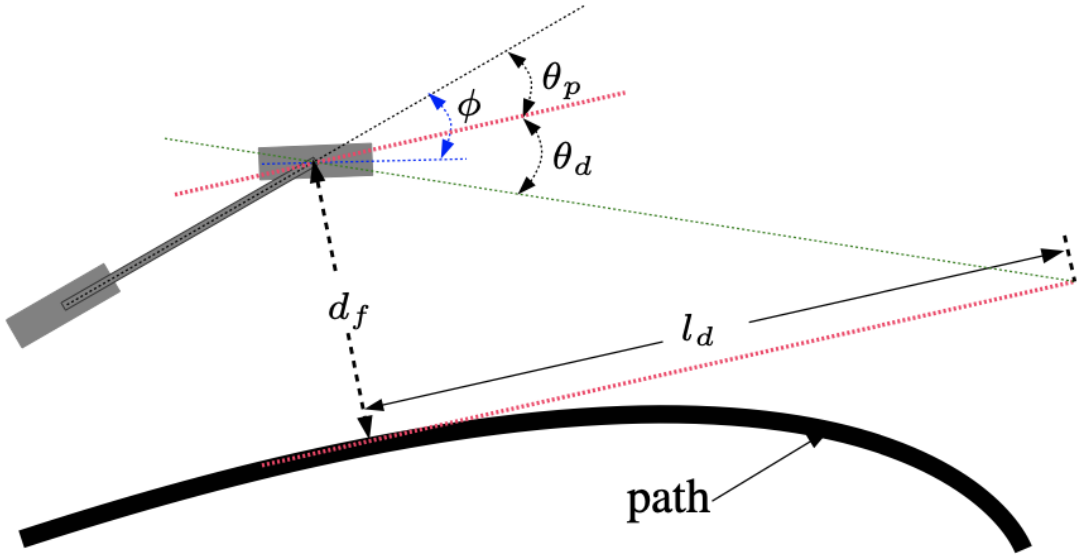

Fig. 3. illustration of cross track error d_f

If the vehicle is perfectly aligned to the path, the cross track error should equal to zero. The second error measures is the angular error, which is the angle the vehicle has to steer to align itself with the heading of the path. In other words, angular error is the difference between the vehicle's current heading and the path's heading. The position error

Table 1. Functional requirements

Functional Requirements	Value
Max cross track error	20cm
Average cross track error	5cm
Average position error along x-axis	10cm
Average position error along y-axis	15cm
Average angular error	10 deg
Convergence time	5 sec

is the error between the vehicle's position and the path's position. The term path is used loosely here to simplify things. Strictly speaking, a path is just a sequence of way-points. The path the vehicle has to follow really just means the next way-point the vehicle has to follow. The last performance measures is the convergence time, which is the first time instant when the cross track error is under 2cm.

6.2 Functional requirements

EA's functional requirements are described in table 1.

6.3 Results

The average cross track error over the entire path is approximately 2.5 cm. The average angular error and the maximum theta error is 2.39 and 6.5 degree respectively. The average position error along the x-axis is 6cm and the average position error along the y-axis is 17cm. The max cross track error is 22cm and the convergence time is 2.5 seconds.

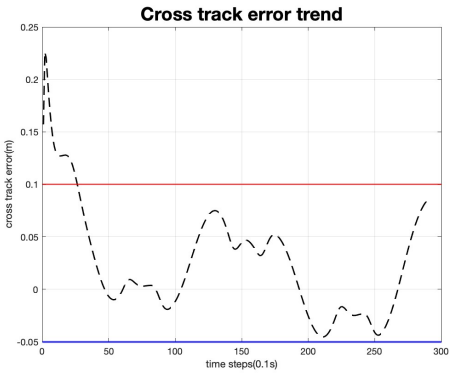

Fig. 4. cross track error

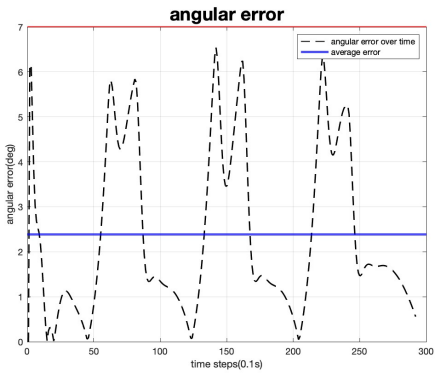

Fig. 5. angular error

The reference trajectory that the vehicle is suppose to follow is represented by the black line and the vehicle's actual trajectory is indicated by the red line and the red star represent the initial state of the vehicle. It is evident from the below graphs that the genetic algorithm can properly tune the controller to force the vehicle to follow the reference trajectory while ensuring stability and convergence. In addition, the GA-tuned MPC controller results in less tracking error compared to a manually-tuned MPC controller.

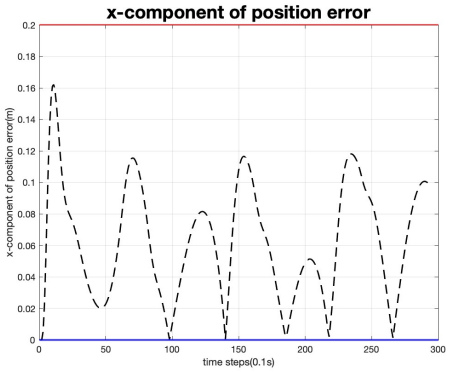

Fig. 6. x-component of position error

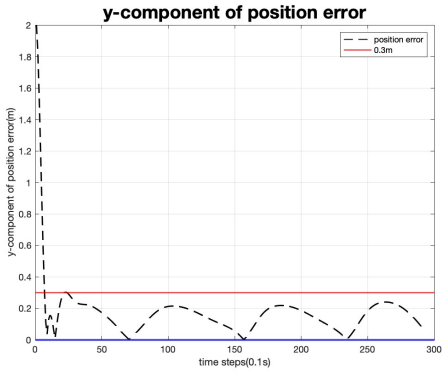

Fig. 7. y-component of position error

The best results from our experiments yielded a fitness of around -52. We found that our solutions adhere to the path's horizontal coordinate better than the vertical - this may be due to the complexity of our test path, and due to the simulated vehicle starting some distance from the path.

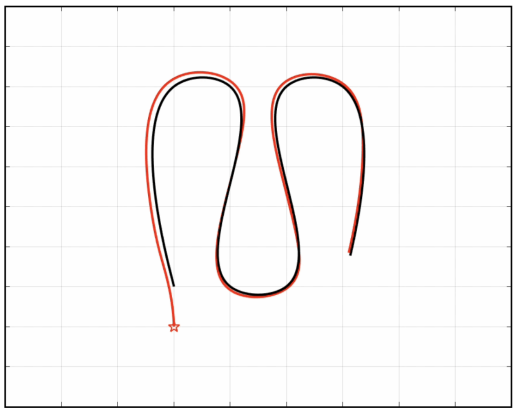

Fig. 8. Tracking performance using MPC controller tuned by trial-and-error method.

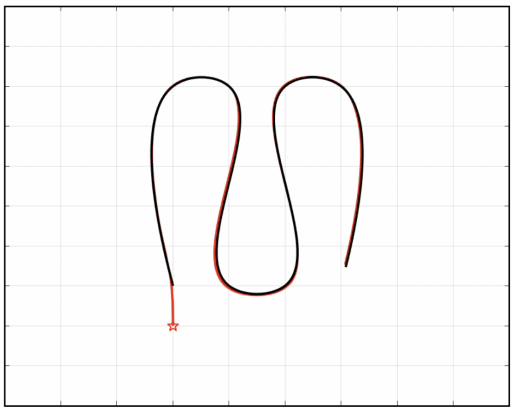

Fig. 9. Tracking performance using MPC controller tuned by genetic algorithm.

7 COMPARISON

Each author must be defined separately for accurate metadata identification. Multiple authors may share one affiliation. Authors' names should not be abbreviated; use full first names wherever possible. Include authors' e-mail addresses whenever possible.

7.1 Initialization Comparison: random vs heuristic

In this section, the diversity and fitness of the initial population generated from three different initialization methods are compared and analyzed. The three initialization methods include two random initialization methods and one heuristic initialization method.

Table 2. Results from initial populations, across 20 generations each, by True-Random, Random-Power, and Heuristic methods.

Category	True-Random	Random Power	Heuristic
Best Fitness	-88	-66	-62
Weighted Average Fitness*	-543.175	-733.451	-729.379
True Average Fitness Fitness	-1583.426	-4231.574	-1192.910
Total Min-fitness** Individuals	44	151	20
Average Min-fit** Individuals per Population	2.2	7.55	1

*: Weighted mean fitness is calculated by removing all individuals of -10000 fitness from a population, then taking the mean.

**: Min-fitness individuals all have a fitness of -10000, representing an unstable solution.

7.1.1 *Diversity analysis of initialization strategies.*

The diversity index is computed by running each population initialization methods 20 times and averaging the results. The diversity index indicates the particular method's ability to generate dissimilar individuals. The following figures shows the diversity of the initial population over 20 runs as well as the average diversity. From experiment,

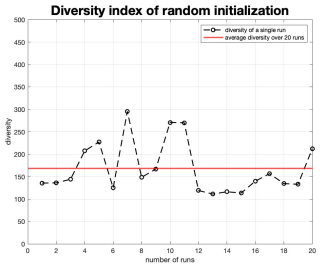

Fig. 10. Population diversity index using random initialization

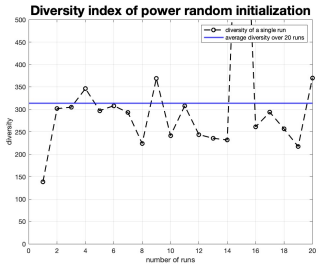

Fig. 11. Population diversity index using power random initialization

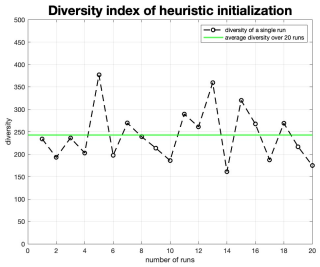

Fig. 12. Population diversity index using heuristic initialization

random initialization has an diversity index of 174, followed by a variant of random initialization with diversity index of 313, and finally the heuristic initialization with diversity index of 247. Although the variant random initialization can generate more diverse initial population, it is prone to generate significantly more invalid solutions/individuals compared to other initialization methods and on average it generates 4 invalid solution in each run. An invalid solution represents an solutions cause convergence failure. When it comes to the fitness of the initial population, the population generated from heuristic initialization has the highest average fitness over 20 runs, with a fitness value of -271. Random initialization method generates population with the lowest average fitness, with fitness value being -413.

7.1.2 *Fitness analysis of initialization strategies.*

Table 2 shows the average results from our three initialisation methods. All algorithms generated twenty initial populations.

We found that the True-Random method produced far fewer individuals of the minimum fitness value, -10000, than the Random-Power method. Furthermore, True-Random had a greater mean fitness than Random-Power, even when minimum-fitness individuals were controlled for. While a large portion of the initial population being unstable does not

necessarily harm the final result of evolution, we deem that a diverse initial population of semi-capable individuals is nonetheless preferable.

Our Heuristic method proved to overall be the best initialisation method, for a few key reasons. Whilst it achieved a similar weighted fitness as Random-Power, the true fitness was the greatest by far. On average, only a single unstable individual was generated per initialisation. This stability enables a fast and reliable evolutionary stage. Furthermore, the Heuristic method is capable of generating individuals with a very high fitness, similarly to Random-Power. Overall, this method acts as the best of both True-Random and Random-Power.

7.2 Tournament Selection vs Stochastic Universal Sampling Selection

We used tournament and stochastic universal sampling selection for parent selection and compared their results. Tournament selection works better in general as shown in Figure 8-13, which use the same saved initialization data set.

Tournament selection works by selecting a random subset of individuals from the population and selecting the most suitable individuals from the subset as parents. This process is repeated to determine multiple parents. The advantages of tournament selection include its ease of implementation, computational efficiency, ability to handle noisy fitness functions, and ability to maintain population diversity. However, tournament selection can also lead to premature convergence if the tournament size is too small or the selection pressure is too high and needs to consider the overall distribution of fitness values in the population. In our project, tournament selection works better with age-based elitism, as seen in Figure 9-11.

Stochastic Universal Sampling Selection involves dividing the population into equally spaced segments based on the cumulative probability distribution of fitness values and selecting parents within each segment using multiple randomly generated pointers. Compared to tournament selection, Stochastic Universal Sampling Selection considers the overall distribution of fitness values in the population. As a result, it provides smoother selection pressure, which is why it only works well with fitness-based survivor selection, as seen in Figure 13. As in Figure 11-12, cross track error, x-error, and y-error have too much fluctuation when working with age-based survivor selection.

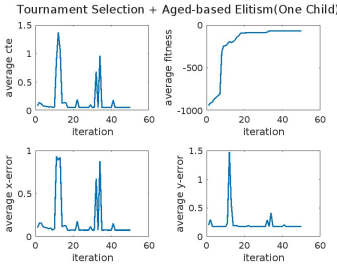

Fig. 13. The trend of tournament selection and age-based elitism(One child).

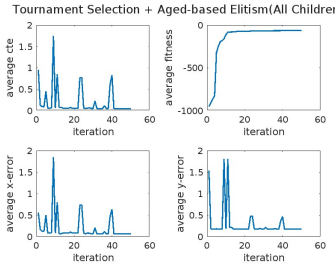

Fig. 14. The trend of tournament selection and age-based elitism(All children).

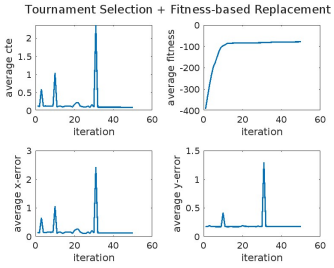

Fig. 15. The trend of tournament selection and fitness-based replacement.

We experimented with two main crossover methods: Wright Heuristic and Heuristic Uniform Crossover (HUX). Both crossover methods were used across five iterations of evolution, and the average fitness across the population was collected. This data is shown in Figures 15 and 16 respectively, with the mean average fitness highlighted in red. For fairness, the initial population was kept consistent; all testing started with the same group of individuals. The survivor selection method was fitness based, which contributes to the initially steep gradient of the graphs.

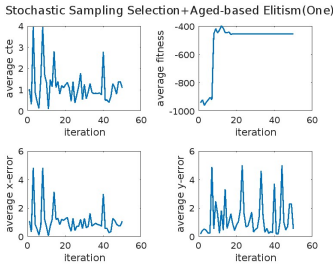

Fig. 16. The trend of Stochastic universal sampling selection and age-based elitism(One child).

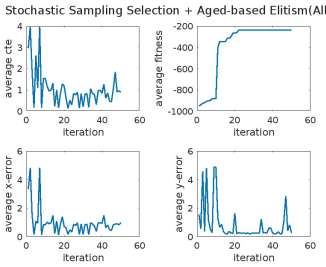

Fig. 17. The trend of Stochastic universal sampling selection and age-based elitism(All children).

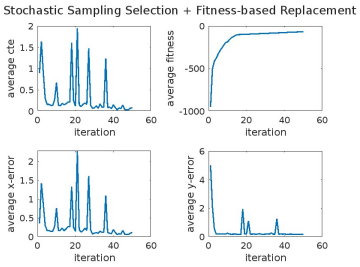

Fig. 18. The trend of Stochastic universal sampling selection and fitness-based replacement.

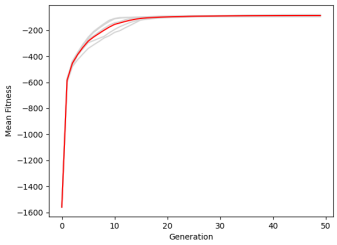

Fig. 19. The average fitness across 5 iterations, utilising best-HUX.

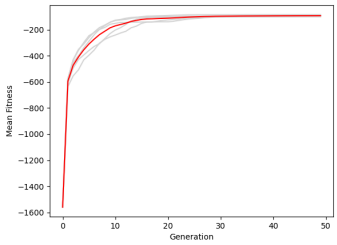

Fig. 20. The average fitness across 5 iterations, utilising Wright crossover.

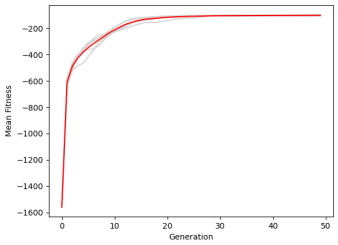

Fig. 21. The average fitness across 5 iterations, utilising random-HUX.

We found that best-HUX performed roughly the same as Wright crossover. It converged a tiny bit faster than Wright, and had a very slight improvement in average best fitness, coming to -86.78 and -90.81 respectively. However, given the sample size, this small difference is unlikely to be significant. While slightly under-performing, Wright executes with much greater speed than best-HUX, leaving the two methods more or less equally comparable.

We also experimented with random-HUX, the results of which can be seen in Figure 17. Compared to best-HUX and Wright, this method converged at a significantly slower rate. The average best fitness for this method was -99.73, which is understandably lower due to the more exploratory nature of random-HUX. A combination of both HUX methods was implemented, utilising random-HUX for the first 25 generations, before switching to best-HUX, but no overall improvement in fitness compared to best-HUX and Wright was found.

7.3 Aged-based elitism vs fitness-based survivor selection

Regarding tournament parent selection and stochastic sampling parent selection, selecting survivors based on age and elitism are better performers. Because our tournament parent selection already exerted sufficient pressure, age-based elitism helped maintain population diversity while ensuring that the fittest individuals were retained over time. As in Figure 8-10, age-based elitism produces better fitness than fitness-based survivor selection when working with tournament selection. Its most significant advantage is balancing the competing goals of maintaining diversity and selecting the most suitable individuals.

However, fitness-based survivor selection works better with stochastic universal sampling selection than age-based elitism, as in the comparison between Figure 11-13. It produces more minor cross track errors, x-error, and y-error and much better maximum fitness. The randomness of the stochastic universal sampling selection method combined with the high survival pressure of fitness-based replacement yields good results.

8 DISCUSSION

8.1 Limitations

Our project is limited in a handful of ways. One such weakness is the applicability of our simulation to hardware. We had initially planned to execute the evolutionary algorithm on board an MPC-controlled differential drive robot. However, several factors prevented from doing so, such as the unpredictability of the initialised population, as well as training time. Our current solution serves as a good method to evaluate parameters for use on hardware, but further research should be done in order to create an evolutionary solution which is both efficient and safe enough to run on-board a robotic system whilst it is in use.

Another limitation is a lack of outside comparison data; the parameters we are optimising are typically hand-tuned by engineers until the MPC controller is performing decently well. As such, there is no one specific method which we can compare our evolutionary approach with. Whilst this limits our ability to evaluate our findings, it does highlight the potential of our project. In the absence of optimal approaches, evolutionary algorithms, can prove powerful solutions.

8.2 Practical Applications

GA can be used reasonably for lane-keeping, obstacle avoidance, path tracking, and speed control. However, real-world automatic driving practice is tricky as it needs high-precision results. Moreover, considering many factors like road friction, fuel volume, etc., inaccurate production may lead to accidents or vehicle damage. Therefore, the integration of real-time sensory data into MPC algorithms and the need for efficient and accurate simulation of the environment are challenges that need to be faced in this field.

REFERENCE

[BB03] Radu Belea and Liviu Beldiman. “A New Method Of Gene Coding For A Genetic Algorithm Designed For Parametric Optimization”. In: *Annals of Dunarea de Jos* 2003 (Dec. 2003).

[Ros+09] Andrés Rosales et al. “Trajectory tracking of mobile robots in dynamic environments—a linear algebra approach”. In: *Robotica* 27.7 (2009), pp. 981–997. DOI: 10.1017/s0263574709005402. URL: https://www.cambridge.org/core/journals/robotica/article/trajectory-tracking-of-mobile-robots-in-dynamic-environmentsa-linear-algebra-approach/EC17FF60D222CE14D339B3CA3E45C898.

[Pra+20] S Prayudani et al. “Analysis Effect of Tournament Selection on Genetic Algorithm Performance in Traveling Salesman Problem (TSP)”. In: *Journal of Physics: Conference Series* 1566.1 (June 2020), p. 012131. DOI: 10.1088/1742-6596/1566/1/012131. URL: https://doi.org/10.1088/1742-6596/1566/1/012131.