Simplified Racing Trajectory Optimization via Evolutionary Algorithms

Daniel Molina-Pérez[1st]*
ESCOM
Instituto Politécnico Nacional
Ciudad de México, México
Departamento de Computación
CINVESTAV-IPN, México City, México
dmolinap@ipn.mx

Edgar Alfredo Portilla-Flores[2nd]*
CIDETEC
Instituto Politécnico Nacional
Ciudad de México, México
aportilla@ipn.mx

Eduardo Vega-Alvarado[3rd]
CIDETEC
Instituto Politécnico Nacional
Ciudad de México, México
evega@ipn.mx

Jacobo Torres Figueroa[4th]
CIDETEC
Instituto Politécnico Nacional
Ciudad de México, México
jtorresf1900@alumno.ipn.mx

Alberto Flores Lara[4th]
ESCOM
Instituto Politécnico Nacional
Ciudad de México, México
afloresl1800@alumno.ipn.mx

Yaxem Flores Reyes Lira[5th]
ESCOM
Instituto Politécnico Nacional
Ciudad de México, México
areyesl1801@alumno.ipn.mx

***Abstract*—This study presents a computationally efficient trajectory optimization framework for racing vehicles, incorporating both dynamic stability constraints and the geometric limitations of the track. The model estimates safe velocities using local curvature and tire-road friction, enabling fast evaluations suitable for evolutionary algorithms. Genetic Algorithms (GA) and Differential Evolution (DE) are tested on the Norisring circuit. Experimental results show that DE outperforms GA in terms of solution quality and robustness, achieving lower lap times with reduced variance. The low computational cost of the proposed approach makes it particularly suitable for real-time implementation and large-scale optimization tasks, significantly outperforming simulation-based and estimation-based methods commonly reported in the literature, which often require several hours for a comparable number of evaluations.**

***Index Terms*—Trajectory optimization, optimal racing line, evolutionary algorithms, differential evolution, genetic algorithms.**

I. Introduction

Trajectory optimization in high-speed vehicles, whether in industrial, robotic, or simulation environments, presents a significant challenge due to the dynamic and geometric constraints involved. In all these contexts, vehicles are expected to perform at the limits of their physical capabilities, maximizing grip, executing aggressive maneuvers, and managing intensive braking to minimize lap times [1]–[3].

The search for the optimal trajectory on a racing circuit is a relatively recent research area. Most studies agree that this trajectory represents a trade-off between minimizing the total distance traveled and maximizing the average speed along the path [4], [5]. However, these two objectives are inherently conflicting: the shortest path typically involves tighter curves that reduce allowable speed, whereas a smoother trajectory allows for higher speeds but results in a longer distance. Consequently, determining the optimal racing line requires balancing distance and speed, while also considering the vehicle's dynamic characteristics.

The maximum speed on a given segment depends, among other factors, on the curvature of the path, tire-surface friction, and aerodynamic force. In general, lower curvature (smoother trajectories) permits higher speeds, which favors strategies that maximize straight-line segments. However, when modeling the trajectory, the chosen representation—such as parameterized curves, control points, or Bézier lines—directly affects the search space and the computational complexity of the problem. This problem can be formulated without requiring a detailed vehicle dynamics model. However, in practice, the optimal trajectory is heavily influenced by the car's dynamic capabilities, such as braking, acceleration, and mass transfer, as well as by the sequence of track segments (e.g., the approach to a curve may vary depending on whether a straight or another tight corner precedes it).

Additionally, there are two main approaches for evaluating a racing trajectory: a simulation-based approach, which provides an accurate performance estimate by running a full lap using a controller within a simulator, and a model-based estimation approach, which uses a computational model of the vehicle to calculate lap time. While the first option is more precise but computationally expensive, the second is more efficient, although its accuracy depends on the quality of the underlying model.

Several methods have been proposed for optimizing vehicle trajectories, both in simulation and real-world contexts. A classical approach was introduced by Braghin et al. [5] [6], who formulated the problem as a balance between minimizing the distance traveled and reducing the curvature of the trajectory. This approach is based on the principle that smoother paths

enable higher speeds.

To more accurately represent vehicle dynamics, Gritschneder et al. [7] proposed a hierarchical model predictive control scheme that uses two instances in a concurrent operation mode: a high-level instance generates trajectories by sampling 2D points, and a low-level instance computes the motion profile considering vehicle dynamics. Both instances share information over a real-time prediction horizon, enabling a continuous formulation of the optimal control problem and significantly reducing execution times.

In commercial video games, methods for automatically generating trajectories have also been explored. In Colin McRae Rally, imitation learning with neural networks is used to replicate racing lines designed by expert drivers. Meanwhile, the Forza Motorsport series applies evolutionary computation to optimize trajectories, demonstrating that these techniques are viable even in interactive entertainment environments [8].

Gerdts [9] addressed vehicle control on real racing circuits by formulating the driving task as an optimal control problem, which is solved using a direct shooting method. To overcome the high computational cost associated with long trajectories, he introduced a moving horizon strategy that divides the circuit into smaller local sectors. Each sector is optimized independently, and the solutions are then combined through consistent transition conditions, ensuring a smooth and dynamically feasible global trajectory.

An interesting alternative was presented by Vesel et al. [10], who represented the vehicle trajectory using radial coordinates relative to a common origin on the track. The complete trajectory was modeled as a smoothed periodic spline, ensuring both continuity and smoothness throughout the circuit. Minimizing lap time was achieved by employing a genetic algorithm (GA) that optimizes the control points, with evaluations performed using a predictive vehicle model. Similarly, Klapalek et al. [11] proposed a GA-based approach where each trajectory was encoded as a sequence of control points interpolated by cubic splines. To ensure that the points remain within track boundaries, they introduce the Matryoshka representation, which transforms complex geometric constraints into simple box constraints. This encoding not only facilitates the use of evolutionary algorithms but also improves efficiency by reducing the generation of infeasible solutions and accelerating convergence toward optimal paths.

Recent research on trajectory optimization using evolutionary algorithms has primarily focused on enhancing the representation of the racing line, with track segmentation emerging as a key area of exploration [11]. In this context, the present work proposes a comprehensive model that integrates the geometric representation of the circuit, simplified vehicle dynamics, and formulates the problem as a constrained optimization task. This task is solved using evolutionary techniques—specifically, GA and Differential Evolution (DE). As a real-world case study, the Norisring street circuit is used to evaluate the proposed approach. Results demonstrate that both algorithms are capable of generating optimal trajectories that satisfy the vehicle's physical stability constraints and the geometric limitations of the track. The proposed framework provides a computationally efficient solution for trajectory planning, making it a good option for real-time implementation and large-scale optimization scenarios.

II. Track Geometric Model

The geometric model is derived from a track centerline that provides the x and y coordinates of each point. Tangent and normal vectors are then used to delineate the track boundaries. To define the track boundaries, tangent and normal vectors are employed. Initially, tangent vectors are computed between consecutive points, as shown in (1):

$$\mathbf{v}_i = (dx_i, dy_i) = (x_{i+1} - x_i, y_{i+1} - y_i) \tag{1}$$

Given the tangent vector $\mathbf{v}_i = (dx_i, dy_i)$, the unit normal vector corresponding to a 90° counterclockwise rotation outward is defined as:

$$\hat{\mathbf{n}}_i = \frac{(dy_i, -dx_i)}{\sqrt{dx_i^2 + dy_i^2}} \tag{2}$$

The track boundaries are defined by displacing the center point (x_i, y_i) in the direction of the normal vector, as indicated in (3):

$$\begin{aligned} \mathbf{B}i^{\text{ext}} &= (x_i, y_i) + dE_i \cdot \hat{\mathbf{n}}_i \\ \mathbf{B}i^{\text{int}} &= (x_i, y_i) - dI_i \cdot \hat{\mathbf{n}}_i \end{aligned} \tag{3}$$

where $\mathbf{B}_i^{\text{ext}}$ and $\mathbf{B}_i^{\text{int}}$ are the coordinates of the outer and inner boundaries at point i, and dE_i and dI_i represent the distances from the centerline to the respective boundaries.

Given the tangent vector $\mathbf{v}_i = (dx_i, dy_i)$, the unit normal vector is computed as a 90° counterclockwise rotation, as shown in (4),

$$\hat{\mathbf{n}}_i = \frac{(dy_i, -dx_i)}{\sqrt{dx_i^2 + dy_i^2}} \tag{4}$$

The track boundaries are then constructed by displacing the center point (x_i, y_i) along the normal direction. The exterior and interior border coordinates are given by:

$$\begin{aligned} \mathbf{B}_i^{\text{ext}} &= (x_i, y_i) + dE_i \cdot \hat{\mathbf{n}}_i \\ \mathbf{B}_i^{\text{int}} &= (x_i, y_i) - dI_i \cdot \hat{\mathbf{n}}_i \end{aligned} \tag{5}$$

where $\mathbf{B}_i^{\text{ext}}$ and $\mathbf{B}_i^{\text{int}}$ represent the coordinates of the outer and inner boundaries at point i, respectively, and dE_i and dI_i denote the distances from the centerline to the corresponding boundaries.

III. Simplified Vehicle Model for Skid Prevention

To estimate optimal trajectories on racing circuits, a simplified vehicle model is proposed, considering the stability conditions required to prevent skidding. This is achieved by maintaining a balance between the centrifugal force and the available lateral grip. The vehicle is modeled as a point mass m moving over a two-dimensional plane representing the track. The model focuses on the lateral forces acting on the vehicle during cornering, assuming that stability is maintained as long as the centrifugal force does not exceed the lateral adhesion provided by the tire–surface interaction.

A. Lateral Force Equilibrium

In a curved section, the vehicle experiences a centrifugal force expressed as:

$$F_c = \frac{mv^2}{r} \tag{6}$$

where v is the tangential velocity of the vehicle and r is the local curvature radius of the path. On the other hand, the maximum lateral grip that the tires can provide is given by:

$$F_{\text{adh}} = \mu m g \tag{7}$$

where μ denotes the lateral friction coefficient between the tire and the track surface, and g is the gravitational acceleration. To ensure vehicle stability and avoid skidding, the following condition must be satisfied at every point along the trajectory:

$$F_c \leq F_{\text{adh}} \quad \Rightarrow \quad \frac{mv^2}{r} \leq \mu m g \quad \Rightarrow \quad v \leq \sqrt{\mu g r} \tag{8}$$

Thus, a safe maximum velocity v_{max} can be defined for each point, based solely on the local curvature and available grip, as in (9):

$$v_{\text{max}} = \sqrt{\mu g r} \tag{9}$$

Equations (6) through (9) define a simplified model that captures the fundamental relationship between vehicle dynamics and lateral stability constraints.

B. Curvature Radius Estimation

The curvature radius r at a given point along the trajectory is estimated using three consecutive points: $P_1 = (x_{i-1}, y_{i-1})$, $P_2 = (x_i, y_i)$, and $P_3 = (x_{i+1}, y_{i+1})$. The value of r corresponds to the radius of the circumcircle passing through these three points, and is computed as:

$$r = \frac{abc}{4A} \tag{10}$$

where $a = |P_2 - P_3|$, $b = |P_1 - P_3|$, $c = |P_1 - P_2|$, and A is the area of the triangle formed by the three points. The area can be obtained by:

$$A = \frac{1}{2}|x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)| \tag{11}$$

This method provides an efficient computational strategy for evaluating local curvature along a discretized path.

C. Lap Time Estimation

Once the safe maximum velocity v_{max} is determined for for the i-th point ($v_{\text{max,i}}$), the time required to travel from P_i to P_{i+1} can be estimated as:

$$\Delta t_i = \frac{d_i}{v_{\text{max,i}}} \tag{12}$$

where the segment length d_i is given by:

$$d_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \tag{13}$$

The total lap time estimate is obtained by summing all segment times Δt_i along the trajectory.

D. Additional Considerations

The goal of this model is to provide a fast and practical evaluation of trajectory optimality, making it suitable for use as an objective function in evolutionary optimization frameworks. Additionally, the model can be extended to account for factors such as aerodynamic loads, weight distribution, or variations in the friction coefficient along the track.

IV. Optimization Problem

The main objective is to minimize the lap time while ensuring that the vehicle does not exceed the available tire-road adhesion (stability constraint) and remains within the track limits (geometric constraint). Each trajectory is defined by a set of N control points selected along the track centerline. At each point i, the vehicle's final position is obtained by displacing the center point in the direction of the unit normal vector using a scalar variable u_i:

$$\boldsymbol{p}_i(u_i) = \boldsymbol{c}_i + u_i \cdot \hat{\mathbf{n}}_i \tag{14}$$

where $\boldsymbol{c}_i$ is the centerline position at point i, $\hat{\mathbf{n}}_i$ is the unit normal vector, and u_i is the lateral displacement from the centerline (positive toward the outer edge, negative toward the inner edge).

Given a displacement vector $\boldsymbol{u} = [u_1, \ldots, u_N]^T$, a continuous curve is reconstructed using Makima (a modified version of the Akima) interpolation [12], [13], and the total lap time is computed based on the velocity constraints imposed by the local curvature, as defined in (9). The choice of Makima is motivated by its capability to reduce oscillations and preserve the shape of the data, especially in regions with rapid transitions. Unlike spline methods, Makima avoids overshooting and ensures local continuity.

The mathematical formulation of the optimization problem is as follows:

$$\begin{aligned} \min_{\boldsymbol{u} \in \mathbb{R}^N} \quad & f(\boldsymbol{u}) = \sum_{j=1}^{T} \frac{|\boldsymbol{p}_{j+1} - \boldsymbol{p}_j|}{v_{\text{max,j}}} \\ \text{s.t.} \quad & -dI_i \leq u_i \leq dE_i, \quad \forall i = 1, \ldots, N \end{aligned} \tag{15}$$

where T is the total number of interpolated samples, $\boldsymbol{p}_j$ is the position of the j-th interpolated point, $v_{\text{max,j}}$ is the maximum allowable velocity at that point, dI_i and dE_i are

the maximum lateral displacements toward the inner and outer track boundaries, respectively, at control point i.

This formulation ensures that the vehicle remains within the track boundaries at the control points. However, violations may still occur at interpolated locations if the control points are not appropriately selected.

V. Experimental Design

As a case study, the urban circuit of Norisring, located in Germany, is employed. This relatively simple layout features two tight corners and a chicane. The total track length is set at 2.3 kilometers. The track centerline data are obtained from the repository in [14]. In this scenario, 28 control points are established, resulting in an optimization problem with 28 decision variables, as illustrated in Fig. 1. These points are placed more densely in the winding sections to minimize the risk of violating the track's geometric constraints. In the simulations, a tire-road friction coefficient of 0.8 is used, and the maximum vehicle speed is set to 45 m/s.

The optimization problem is addressed using GA and DE to perform a comparative performance analysis between the two methods. The GA implementation employs deterministic tournament selection, Simulated Binary Crossover (SBX), and polynomial mutation. A generational replacement strategy with elitism is used to retain the best individual in each generation. For DE, the classical `DE/rand/1/bin` scheme is adopted.

Each algorithm is executed independently 20 times, with a computational budget of 100,000 function evaluations per execution. The parameter settings for both methods are consistent with recommendations from the literature [15]–[17]. Specifically, both algorithms use a population size of 50. The GA uses a crossover probability of 0.9 and a mutation probability of 0.03. The distribution index for SBX is set to $N_c = 5$, and for polynomial mutation, $N_m = 20$. For DE, the scaling factor is set to $F = 0.5$, and crossover is performed with probability zero. All algorithms were executed using Intel Core i7-4790 CPU @3.6 GHz, 32 GB RAM, with Windows 10 Enterprise N, and MATLAB 2024b served as the programming platform.

VI. Results

The results are summarized in Table I, reporting the best, mean, and worst objective values, along with the standard deviation for the 20 independent runs for each algorithm.

DE consistently outperformed GA, yielding the best individual lap time of 56.9442 seconds and the lowest average lap time of 57.0375 seconds, whereas GA achieved 56.9631 and 57.2007 seconds, respectively. These results indicate that DE not only produces more efficient solutions but also exhibits greater robustness, as evidenced by its lower standard deviation (0.0492 for DE vs. 0.1746 for GA). This stability is further reflected in the worst-case outcomes, where DE recorded 57.1573 seconds, compared to 57.5636 seconds for GA. A Wilcoxon rank-sum test at a 5% significance level confirmed that the differences between DE and GA are statistically significant within the context of this problem.

Additionally, the average runtime for 100,000 function evaluations was 89.01 seconds for DE and 82.66 seconds for GA, demonstrating a low computational cost for both algorithms. This efficiency becomes especially notable when compared to previous studies such as [5], where 90,000 evaluations require between 12 and 36 hours for simulation-based approaches, and 3 to 5 hours for estimation-based methods. These comparisons highlight the significant computational advantages of the proposed model, addressing one of the primary challenges reported in the literature: the high computational burden associated with objective function evaluations in racing optimization problems. Consequently, the model is a good option for large-scale experimentation or integration into real-time optimization frameworks.

Fig. 2 shows the best solution obtained by DE. As observed, the trajectory tends to follow straight lines along the straight sections of the track and approaches each corner from the opposite edge, minimizing steering angles and reducing the need for sharp turns that would otherwise lower the vehicle's speed. This driving behavior closely resembles how professional drivers maneuver in real racing scenarios to maintain momentum and optimize corner exit velocity.

Notably, the detailed view of the chicane demonstrates how

TABLE I
Performance statistics for each algorithm

Indicator	DE	GA
Best	56.9442	56.9631
Mean	57.0375	57.2007
Worst	57.1573	57.5636
Standard deviation	0.0492	0.1746
Average time(s)	89.0115	82.6574

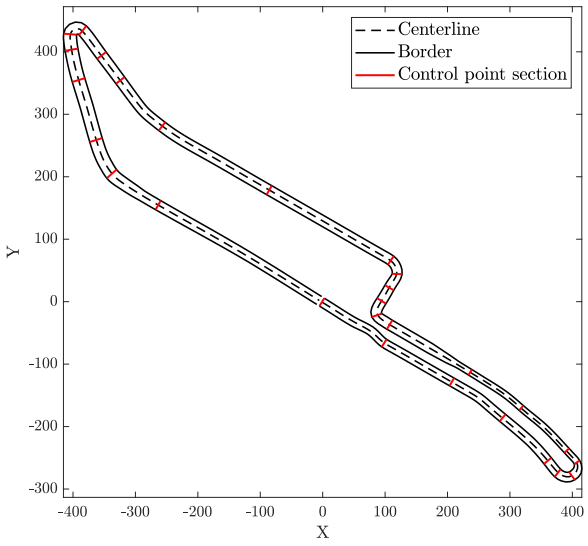

Fig. 1. The urban Norisring circuit in Germany.

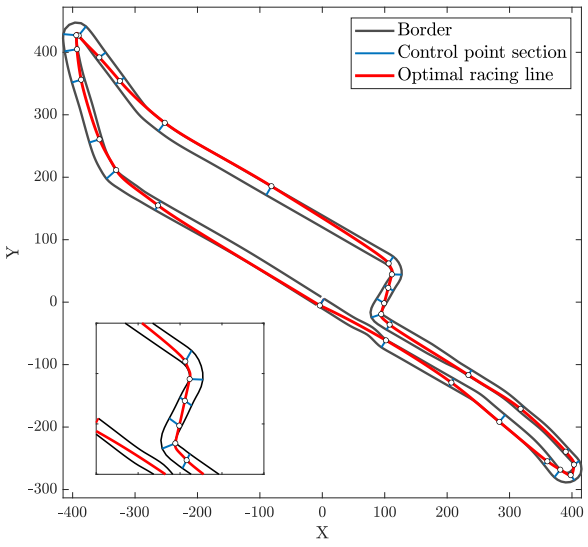

Fig. 2. Best trajectory found by DE, achieving a lap time of 56.9442 seconds. A detailed view of the chicane illustrates the curvature adaptation.

the algorithm adapts to complex sections of the track by modulating curvature and positioning the vehicle for optimal transitions. This behavior highlights DE's capability to not only optimize localized decision variables but also generate globally effective trajectories that comply with physical constraints such as tire grip and lateral forces.

VII. Final considerations

This work presents a trajectory optimization framework for racing scenarios, integrating both the vehicle's dynamic stability constraints and the geometric limitations imposed by track boundaries. The proposed model relies on a simplified representation that enables the estimation of safe velocities based on local curvature and tire-road friction. This formulation allows the use of optimization algorithms while ensuring feasibility throughout the trajectory.

The experimental results confirm that both GA and DE are capable of producing high-quality solutions with robust performance across multiple runs, with DE consistently outperforming GA across all metrics. The best trajectory obtained by DE closely resembles professional driving behavior, minimizing steering effort in corners and preserving speed on straights.

A key strength of the proposed framework lies in its low computational cost. The model achieves significantly faster evaluations than simulation-based or estimation-based approaches reported in the literature, which often require several hours of computation. This positions the proposed method as a viable and scalable alternative for real-time or large-scale optimization tasks.

As future work, the model can be extended to incorporate more complex vehicle dynamics and environmental factors, such as aerodynamic loads, weight distribution, and spatial variations in the friction coefficient along the circuit.

Acknowledgment

The authors acknowledge the support from the Secretaría de Ciencias, Humanidades, tecnología e Innovación (SECIHTI) and its support in Mexico through the institutions ESCOM-IPN, CINVESTAV-IPN, and CIDETEC-IPN.

References

[1] Z. Wang, T. Taubner, M. Schwager, Multi-agent sensitivity enhanced iterative best response: A real-time game theoretic planner for drone racing in 3d environments, Robotics and Autonomous Systems 125 (2020) 103410.

[2] A. Remonda, N. Hansen, A. Raji, N. Musiu, M. Bertogna, E. Veas, X. Wang, A simulation benchmark for autonomous racing with large-scale human data, Advances in Neural Information Processing Systems 37 (2024) 102078–102100.

[3] C. Weaver, R. Capobianco, P. R. Wurman, P. Stone, M. Tomizuka, Real-time trajectory generation via dynamic movement primitives for autonomous racing, in: 2024 American Control Conference (ACC), IEEE, 2024, pp. 352–359.

[4] M. Bevilacqua, A. Tsourdos, A. Starr, Particle swarm for path planning in a racing circuit simulation, in: 2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), IEEE, 2017, pp. 1–6.

[5] M. Botta, V. Gautieri, D. Loiacono, P. L. Lanzi, Evolving the optimal racing line in a high-end racing game, in: 2012 IEEE conference on computational intelligence and games (CIG), IEEE, 2012, pp. 108–115.

[6] M. Bevilacqua, A. Tsourdos, A. Starr, Particle swarm for path planning in a racing circuit simulation, in: 2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), IEEE, 2017, pp. 1–6.

[7] F. Gritschneder, K. Graichen, K. Dietmayer, Fast trajectory planning for automated vehicles using gradient-based nonlinear model predictive control, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 7369–7374.

[8] L. Cardamone, D. Loiacono, P. L. Lanzi, A. P. Bardelli, Searching for the optimal racing line using genetic algorithms, in: Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, IEEE, 2010, pp. 388–394.

[9] M. Gerdts, A moving horizon technique for the simulation of automobile test-drives, ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik: Applied Mathematics and Mechanics 83 (3) (2003) 147–162.

[10] R. Vesel, Racing line optimization@ race optimal, ACM SIGEVOlution 7 (2-3) (2015) 12–20.

[11] J. Klapálek, A. Novák, M. Sojka, Z. Hanzálek, Car racing line optimization with genetic algorithm using approximate homeomorphism, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2021, pp. 601–607.

[12] H. Akima, A new method of interpolation and smooth curve fitting based on local procedures, Journal of the ACM (JACM) 17 (4) (1970) 589–602.

[13] H. Akima, A method of bivariate interpolation and smooth surface fitting based on local procedures, Communications of the ACM 17 (1) (1974) 18–20.

[14] Institute of Automotive Technology, Racetrack database, accessed: 2025-06-20. Specific path: racetrack-database/tree/master (2024).
URL https://github.com/TUMFTM

[15] K. V. Price, Differential evolution: a fast and simple numerical optimizer, in: Proceedings of North American fuzzy information processing, IEEE, 1996, pp. 524–527.

[16] K. Deb, K. Sindhya, T. Okabe, Self-adaptive simulated binary crossover for real-parameter optimization, in: Proceedings of the 9th annual conference on genetic and evolutionary computation, 2007, pp. 1187–1194.

[17] K. Deb, D. Deb, Analysing mutation schemes for real-parameter genetic algorithms, International Journal of Artificial Intelligence and Soft Computing 4 (1) (2014) 1–28.