This full text paper was peer-reviewed at the direction of IEEE Instrumentation and Measurement Society prior to the acceptance and publication.

# *Particle swarm for path planning in a racing circuit simulation*

M. Bevilacqua
Cranfield University,
Cranfield, Bedfordshire, MK430AL,
United Kingdom
m.bevilacqua@cranfield.ac.uk

A. Tsourdos, A. Starr,
Cranfield University,
Cranfield, Bedfordshire, MK430AL,
United Kingdom
{a.tsourdos, a.starr,}@cranfield.ac.uk

*Abstract*—**A racing line is a trajectory that allows car of defined parameters to travel through a given track in minimum time. The growing competition in motorsports racing increases a demand for planning the vehicle path as accurately as possible, what opens a room for computerized methods. In this work, a method of race line generation basing on known optimization methods was presented and implemented using Matlab software.**
**A novel idea is to divide path optimization into several stages – optimizing curved fragments of the track independently, and then joining them by optimizing the transitions at long straight sections. This way the amount of variables being optimized at each time is reduced, what simplifies the problem and the solver is less prone to fall into local minima. The optimization task itself is stated as finding a set of control points positions, such that the interpolated path can be travelled in a minimum time. To answer this problem, three Matlab solvers are implemented and compared: Fmincon (GlobalSearch), Genetic Algorithm and Particle Swarm.**
**The algorithm was tested and presented on realistic racing circuits, that is Silverstone and Circuit De Barcelona – Catalunya. The results are encouraging for further development of a multi-stage approach, however they reveal weaknesses in interpolation approach. The resulting path is limited to the capabilities of restricted control point's positions. Moreover, traditional interpolation methods may not always create a path optimal for a real vehicle.**

**Keywords— Optimization, Race line, Trajectory Planning, Autonomous Races, Genetic Algorithm, Particle Swarm.**

## I. INTRODUCTION

In any kind of races, the ultimate aim is to get through a given track as quickly as possible. With constant car parameters, a field for manipulation is the driving technique – involving choice of optimal trajectory. A line taken during the run is called a race line. In case of racing with multiple opponents (ex. at Formula 1 type of races) it strongly depends on other vehicles position and taken strategy. A racer can, for instance, plan an overtaking maneuver or to block opponents from taking a faster line. The very basic race line, however, is always the same – the one that allows to drive through a racing circuit in a minimum time. It can be used as a default trajectory during the races (when one is not in danger of losing current position) or – more importantly – as a trajectory for laps done alone, ex. during qualification rounds.

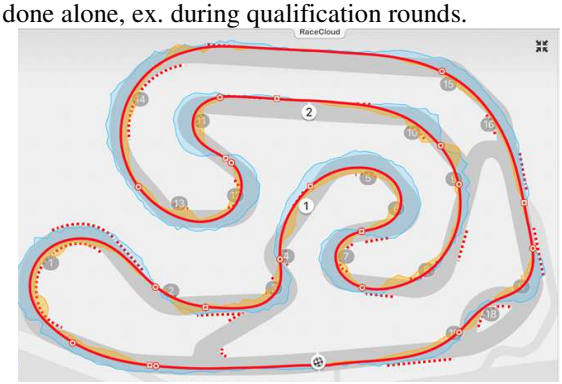


Figure 1 - An example of a racing line [1]

Racers normally choose an optimal line basing on their experience. A usual approach to a single turn is to stay close to the outside edge of the track, brake aggressively before the turn, make a smooth arc almost touching the inside border (at a point called *apex*) and then accelerate while driving to the very outside of the track – as can be seen at the turn no. 16 on Fig.1.
The general aim is to carry as much speed as possible, but an exact line depends on car capabilities – i.e. the grip and possible acceleration, due to the trade-off between the highest speed and lowest distance paths. It is also influenced by a general shape of the track, i.e. features previous to and following a given turn.
A movement of a car in a racing circuit can be divided into three basic motions:

- Accelerated/decelerated linear motion - on long straight sections
- Uniform nonlinear motion - cornering at a constant speed
- Accelerated/decelerated nonlinear motion - right before or after a turn

The first type of motion is trivial and bases mainly on car performance. The racing technique comes down to maximizing acceleration (i.e. controlling traction or gear changes) and choosing the right point to switch from accelerating to decelerating for the next corner.

978-1-5090-3596-0/17/$31.00 ©2017 IEEE

For cornering at a constant speed, what can be often seen in central parts of the turns, formula for travelling time takes a trivial formation:

It is clear that in order to minimize time, the speed must be kept as high as possible with minimum distance to travel. For such an easy case, an optimal trajectory would be the one that combines those values in the best ratio, basing on car capabilities (i.e. maximum lateral acceleration). The distance is limited by track borders and the speed is limited by maximum friction, which must counteract the centrifugal force:

$$F_c = \frac{m \cdot v^2}{r} \leq \mu \cdot m \cdot g \quad \text{(I)}$$

Where: Fc – centrifugal force, m – mass, r – turn radius, g – gravitational acceleration, μ – friction coefficient.

Thus, the bigger a turning radius is, the higher velocity can be achieved through the corner (given a constant grip).

The most important effect on the overall speed is, however, the ability to accelerate quickly right after cornering or to decelerate in minimum time before it. Given the basic formula for distance in accelerated movement (discarding initial distance, i.e. the distance travelled in previous sections):

$$s = v_0 t + \frac{at^2}{2} \quad (2)$$

The time parameter can be obtained:

$$t = \frac{-v_0 + \sqrt{v_0^2 + 2as}}{a} \quad (3)$$

Where: a – linear acceleration, v0 – initial speed

As can be seen, in this case the time depends on: initial speed (i.e. the speed of uniform cornering before acceleration), distance to travel and possible acceleration. To visualize an importance of each of those parameters, plots of their dependence upon time were constructed. At each of them one parameter is changed, with others being constant and set to maximum values. Their range was chosen to resemble real values seen at racing circuits.

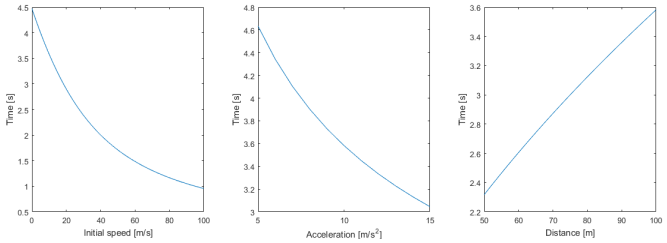


Figure 2 - Travelling time dependence on: initial speed (a), vehicle acceleration (b) and distance (c) in accelerated motion

It can be seen that that the biggest influence on overall time have the initial speed, what can make a difference of over 3 seconds on 100m track fragment. Another important parameter is longitudinal acceleration, allowing to reduce time by 1.5 seconds. The difference in distance covered is less important, as the time gain is about 1s on route shorter by 40 meters, what is a hugely exaggerated value as for a straight section.

When accelerating after the turn, initial speed is determined by turn radius, as described for uniform cornering case. Acceleration is limited by engine power or available friction. The latter is determined by car parameters (tyre grip, aerodynamic downforce) and the friction already used to counteract centrifugal force. Basing on a simple rule that summary force must stay within friction limits, that is:

$$F_c^2 + F_a^2 \leq (\mu \cdot m \cdot g)^2 \quad (4)$$

Where: Fc – centrifugal force, Fa – longitudinal friction force

It can be stated that restricting the centrifugal force (depending on a turn radius for a given speed) allows for greater longitudinal acceleration, before meeting engine limits. It is important to note that equation (4) is valid only for isotropic friction forces, what is not the case for advanced vehicle tires. Due to additional effects, like varying friction coefficient or observed sideslip angle, a usually used tire model is Pacejka's *"Magic Formula"* [2]. The basic notions, however, are still the same and a simplified model of isotropic friction is used in this work.

In summary, a perfect racing line through a turn would be an optimal combination of those three objectives: to start accelerating with a maximum initial speed, to finish the turn with as straight a line as possible (allowing for maximal acceleration) and to reduce the distance. Those often conflicting lines are shown on Fig. 1.

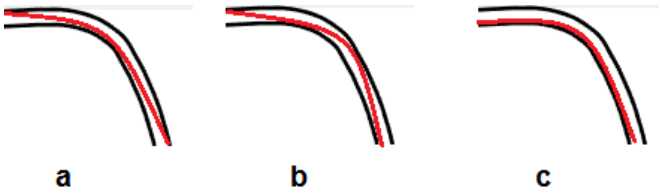


Fig. 1 Lines of highest velocity (a), highest exit acceleration (b) and lowest distance

Analogues approach is taken for deceleration before taking a turn, with the main difference observed in initial speed. A driver wants to keep the maximum speed as long as possible, however a reduction is needed before entering the turn in order to not exceed maximum centrifugal force limit. As a result, maximum deceleration is preferred to accomplish this maneuver in minimum time. The deceleration depends on vehicle braking power (coming from brakes and engine) and maximum friction, limited similarly as in the case of acceleration.

## II. LITERATE REVIEW

The concept of automatic path planning for racing cars, although relatively new and not widely used, has already been touched in a number of research papers. Various methods of optimization or artificial intelligence were investigated to solve the problem of optimizing a trade-off between path of the shortest lap distance with highest velocity one. Two main attitudes dealing with this issue can be distinguished – using optimal control theory to minimize a cost function (i.e. time of doing a full lap) and artificial intelligence techniques (f.e. genetic algorithms). In this chapter, already proposed solutions will be presented and assessed.

A basic approach is described by Braghin et al. in [5]. The problem is formulated as finding the optimal compromise between the line of minimum distance and the one of minimum curvature (maximum speed). These targets are usually conflicting, as shown on Fig.1 below:

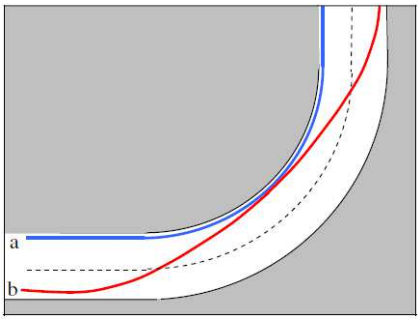


Fig.1 Comparison between shortest space (a) and lowest curvature (b) trajectories.

The track of a constant width is represented by its centerline, with car's position given as the fraction of the track's width. The shortest and lowest curvature paths may be found by purely geometrical algorithms solving a constrained minimization problem. The optimal path is defined by a linear combinations of these two (distinguished by a weighting coefficient) and depends on car dynamics. For each possible trajectory (represented with a discretized weighting coefficient), a speed profile is created, thus allowing to choose the one with minimal lap time ( Fig. **2**).

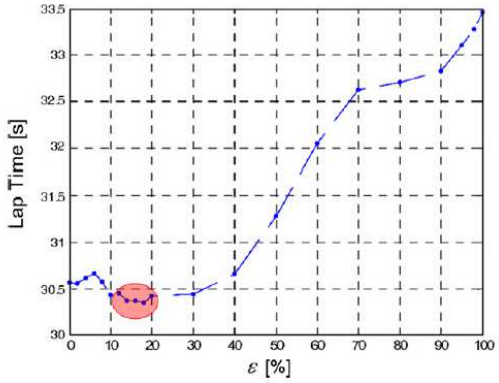


Fig. 2 Lap time on the test track as function of the weighting coefficient ε [5]

Those drawbacks were later conquered by Cardamone at al. [6] by using genetic algorithms and choosing different weights for each section of the track. The starting points of those sections are identified by intersections between the shortest and the minimum curvature path, as seen at Fig. **3**, which also clearly shows the difference between these two lines of different geometric objectives. Another improvement on original algorithm is that Cardamone decided to use real simulations to obtain fitness value of each possibility, instead of evaluating it basing on a vehicle model. The proposed solution outperformed original strategy at every simulation, however it raises a concern if the transition between lines basing on different weighting coefficients will always be fluent and manageable with a real car.

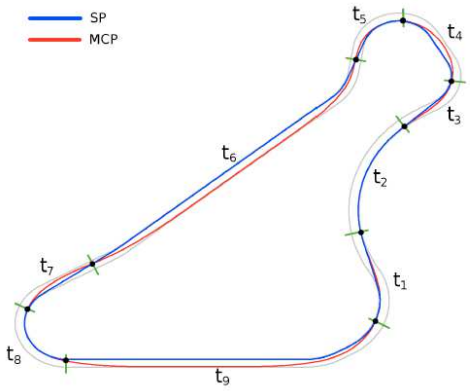


Fig. 3 Shortest path (SP) and the minimum curvature path (MCP) intersections [6]

The use of genetic algorithms was further developed to evolve a racing line from a scratch while representing the track with a set of Bezier curves [7].

| Track | Simplix | Cardamone et al. | Simulation-Based Evaluation | Estimation-Based Evaluation |
|---|---|---|---|---|
| A-Speedway | 27.56 | **24.70 ± 0.00** | **24.76 ± 0.04** | 24.89 ± 0.00 |
| CG Speedway 1 | 40.12 | **39.37 ± 0.00** | **39.55 ± 0.13** | 40.38 ± 0.08 |
| Ruudskogen | 63.20 | **62.73 ± 0.00** | **62.85 ± 0.10** | 63.52 ± 0.01 |
| Alpine 2 | 94.33 | **92.53 ± 0.01** | 94.16 ± 0.09 | 96.56 ± 0.10 |

Table1 Comparison of the best performance achieved with different algorithms

Another approaches base on standard optimization methods. Optimal Control problem may be defined as finding the control history which minimizes a performance index, ex. a time integral of a function of: time, state vector and a control signal. Casanova et al. [8] used a parallel-shooting method to evaluate the minimum time to complete a double-lane change maneuver. A special attention is placed on transient motion of car dynamics, i.e. the yawing moment evoked by steering wheels at the very beginning of a turn, along with the basic aerodynamic forces and lateral load transfer of a seven degrees of freedom vehicle model. The track is discretized by a communication grid (control changes may occur only at the nodes of this grid) and divided into a few segments (depending on the complexity of the track). Because the vehicle state trajectories may be univocally evaluated for any set of values of the augmented control array, the cost function to be minimized may be defined as a function only of control variables. This was done using the Sequential Quadratic Programming algorithm implemented in the Matlab Optimization Toolbox. The results are shown at Fig.4 and proved to be highly accurate, although time consuming – calculations for a set of three bends took about 2 hours on a computer used in the study. The effect of yaw inertia on the minimum lap time was shown at Fig.**5**. It is clearly seen that, within reasonable limits, its influence is not significant and so the exact modelling of this parameter is not crucial.

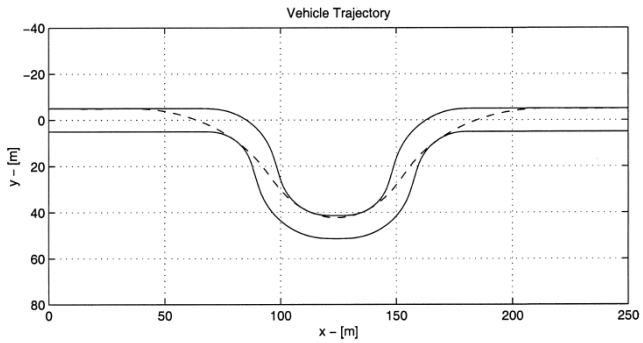


Fig.4 Optimized vehicle trajectory [8]

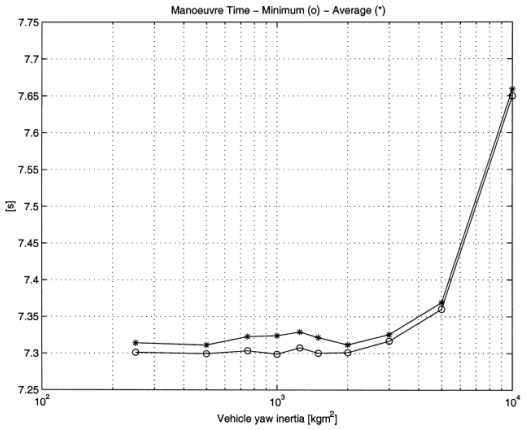


Fig.5 Vehicle maneuver time in relation to yaw moment of inertia [8]

This idea was further developed by Rucco in [9]. A vehicle mass transfer effect was incorporated into the model and its dynamics were distinguished to longitudinal and transverse. The optimal control problem is, similarly, restated with a distance along centerline as an independent variable, what ensures the fixed integral horizon – i.e. the track length (instead of a varying lap time).

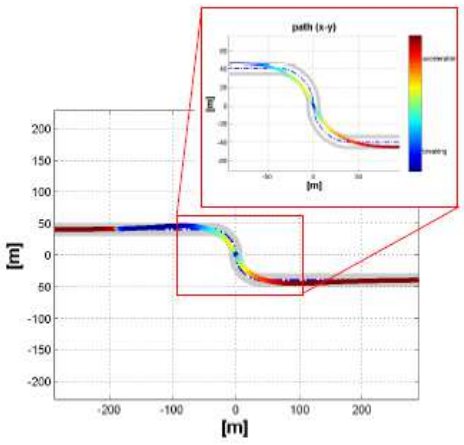


Fig.6 Trajectory and speed profile from Projector Operand Newton method [9]

Timings and Cole in [10] aim to avoid the need to resort to nonlinear optimization techniques by linearizing the cost function around a current vehicle state. The task is stated as maximizing the distance travelled in a fixed amount of time, and so a cost function is defined as the sum of distances travelled in all time steps. Those are represented using lateral displacement and heading angle errors (referring to the track centerline) and linearized to form a direct output of systems state equation (basing on small angles approximation). To make this possible, however, a constant speed of a vehicle is assumed what poses a significant restriction on usefulness of this method. The accuracy of the algorithm was checked by calculating the optimal path for a vehicle with no dynamics (treated as a point mass) and comparing it to the shortest distance path, as seen on Fig.7. A slight discrepancy may be noticed, resulting from a big difference between vehicle path angle and track's centerline. Although this issue is not significant and doesn't exist in majority of instances, a track geometry must be chosen judiciously for the solution to remain feasible.

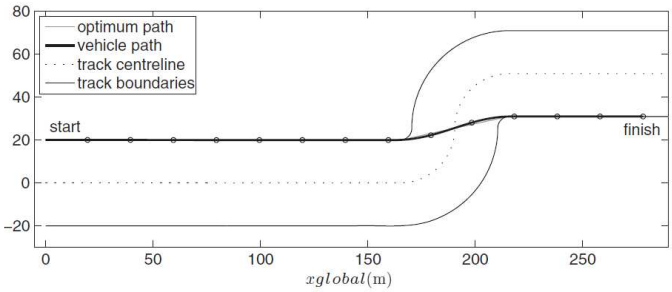


Fig.7 Optimized vehicle path simulation results for a point mass [10]

## III. METHODOLOGY

### *1) Track encoding*

According to the initial assumptions, only planar (2D) tracks of a constant width are investigated. This simplification allows to represent the track by its centerline - encoded in a form of *(x,y)* coordinates matrix. This is a simplifying assumption as some real tracks may have sections of different widths, however it is sufficient for algorithm performance evaluation at the current stage of development. The width must be provided as an external parameter, preferably the modal value of measured widths along the track.

The first point *(0,0)* of coordinates matrix is taken as a default starting point and the direction is oriented along the growing number of rows. It can, however, be changed in the program by flipping the centerline matrix upside-down. The centerline matrix may be supplied directly in a form of *csv* file or derived by other means.

An exemplary photo of a racing circuit is shown on Fig.11.

Fig.11 Silverstone racing circuit–real photo (left) and simplified plan (right) [11]

*2) Provided data*

The data provided by partner company comes in a form of **.kml* files [12]. This is a format used by *Google* to display geographic data in their software. It contains GPS data of track borders described by the set of points of a given longitude, latitude and height. Those values are easily transferred to Cartesian axes with *Matlab* function *lla2flat()* [13].

Although the data is well prepared for graphical presentation, it is difficult to find the centerline analytically due to the noise and mixed points' order. That is why additional function was created to extract the centerline from a black-white image. Provided track can be easily represented as an image (**Error! Reference source not found.**) with use of *Matlab* software and then supplied to the program.

## IV. SIMULATIONS RESULTS AND DISCUSSION

Three solvers were investigated: *Fmincon, Genetic Algorithm, Particle Swarm*. Firstly, their crucial parameters, lap time and maximum speed, were optimized to assure the best performance. Simulations were run on a track resembling the *Silverstone* racing circuit, as provided by the company. The size of Particle Swarm, as well as population in Genetic Algorithm, was set to 200. This was chosen to be optimal combination of proper convergence and computational time, however greater values are recommended for increased accuracy in further use. Those two solvers are basing on stochastic procedures, so the results may vary slightly with each run. To compensate for this randomness, each setting was tested 5 times and the mean from results was taken as a final value. The major criterion of effectiveness is the objective function score that is the lap time. However computations time is also presented, as it has to stay within reasonable limits and cannot be completely ignored.

*1) Fmincon*

**Parameter:** *FiniteDifferenceStepSize*

It is the size of the steps that the solver takes to calculate finite differences, what has an influence on steps size in optimizing point's positions. This parameter affects the tendency to falling into local minima – too big steps result in aggressive changes of overall line, what may be treated as detrimental for the objective value (even if it is the right change in a bigger perspective). On the other hand, too small changes will have almost no effect on the result and solver will treat it as a converged state, instead of searching for a better alternative.

Parameter was changed in range: [$10^{-9}$, $10^{-8}$, … , $10^{-1}$, $10^{0}$], results are shown in **Table** and **Fig. 1**12.

Table 2 FMINCON parameters optimization

| FinDiffRelStep | Lap Time [s] | Elapsed [min] |
|---|---|---|
| 0.000000001 | 164.29 | 31.07 |
| 0.00000001 | 165.11 | 26.03 |
| 0.0000001 | 164.47 | 35.61 |
| 0.000001 | 164.27 | 35.1 |
| 0.00001 | 164.5 | 26.37 |
| 0.0001 | 164.33 | 31.01 |
| 0.001 | 164.83 | 28.24 |
| 0.01 | 166.03 | 25.42 |
| 0.1 | 167.76 | 16.37 |
| 1 | 170.39 | 12.11 |

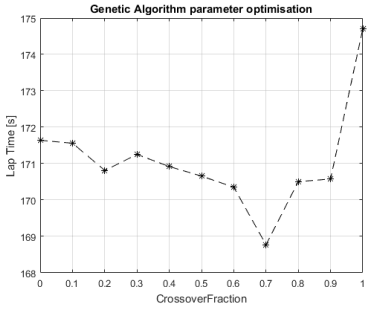


Fig. 12 Fmincon parameter optimization – step of finite differences

It can be seen that keeping this parameter in the middle of proposed range gives the best results. The best value, that is $10^{-6}$, was chosen to be used in the algorithm. The value of $10^{-5}$ gives a similar lap time with much shorter calculations, however such an unexpected decrease in calculations time may suggest that solver has found local minimum, what should be treated as a statistical error. A general trend in convergence time can be stated as decreasing with growing step size, what is intuitional as the solver moves faster through possible solutions with bigger steps.

*2) Genetic Algorithm*

**Parameter:** *CrossoverFraction*

For every new generation, part of potential solutions are simply changed with each iteration (what is called a *mutation*), and part of them are mixed amongst themselves (*crossover*). This parameter changes a fraction of points to be mixed, what can have an important effect on finding the global solution. Having too many generations produced from mixing good results may result in solver converging too early in a local minimum. However making most of the generations evolve individually would increase randomness of the process.

Investigated range: [0 : 0.1 : 1], comparison presented in Table IV and Fig. 3.

Table IV Genetic Algorithm parameters optimization

| CrossoverFraction | Lap Time [s] | Elapsed [min] |
|---|---|---|
| 0 | 171.64 | 8.49 |
| 0.1 | 171.56 | 7.58 |
| 0.2 | 170.81 | 6.66 |
| 0.3 | 171.25 | 6.54 |
| 0.4 | 170.92 | 7.38 |
| 0.5 | 170.66 | 6.56 |
| 0.6 | 170.35 | 7.14 |
| 0.7 | 168.77 | 7.02 |
| 0.8 | 170.51 | 6.37 |
| 0.9 | 170.58 | 5.33 |
| 1 | 174.7 | 1.39 |

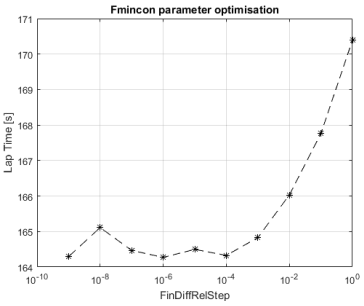


Fig. 13 Genetic algorithm parameter optimization – crossover fraction

Indeed, the results show that best times are achieved when both modes are mixed in a similar ratio, with a slight tendency towards the bigger weight of random mutation. The best score was achieved with a crossover fraction of 0.5, what means that equal parts of new generations were created by both means. Elapsed time is very similar in almost all trials and thus it was not taken into consideration, although slightly prominent values at the extremes suggest that solver basing on mixed solutions converges much faster than keeping to random evolution.

*3) Particle Swarm*

**Parameter:** *SelfAdjustmentWeight* and *SocialAdjustmentWeight*

The speed of every particle is updated basing on its inertia, self-adjustment weight and social-adjustment weight. The two

latter weights are responsible for social behavior of the particle and those are going to be reviewed. The first one reflects a tendency to move towards the best place a particle has already visited, while the second makes the particle move towards the best place in current neighborhood.

Change range: [0 : 0.3 : 3], see Table1 and Fig. **1** below.

Table 4 Particle Swarm parameters optimization

| Social AdjWeight | SelfAdjWeight | LapTime | Elapsed |
|---|---|---|---|
| 0 | 3 | 177.46 | 2.1 |
| 0.3 | 2.7 | 164.49 | 4.73 |
| 0.6 | 2.4 | 164.3 | 4.24 |
| 0.9 | 2.1 | 164.48 | 3.86 |
| 1.2 | 1.8 | 165.4 | 3.83 |
| 1.5 | 1.5 | 165.86 | 3.93 |
| 1.8 | 1.2 | 166.12 | 3.69 |
| 2.1 | 0.9 | 165.01 | 3.7 |
| 2.4 | 0.6 | 166.22 | 3.93 |
| 2.7 | 0.3 | 166.68 | 3.59 |
| 3 | 0 | 169.41 | 2.57 |

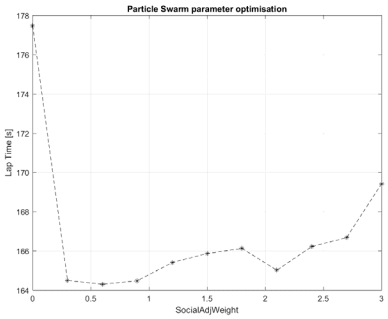


Fig. 14 Particle Swarm parameter optimization – weight of social behavior

The results show that the higher Self Adjustment Weight is, the better is solver's performance. Similarly as in Genetic Algorithm, it may be caused by the tendency to fall in local minima found by other particles when too much pressure is placed on Social Adjustment. The values chosen for this algorithm are respectively: 0.6 and 2.4. Elapsed times are again similar, despite at extreme values where the computational burden is lowered significantly by zeroing one parameter. The swarm size and population size for *Particle Swarm* and *Genetic Algorithm* was increased to 500 to get more accurate results. Rest of important options was left to default values, as no modification was found to give significantly better results.

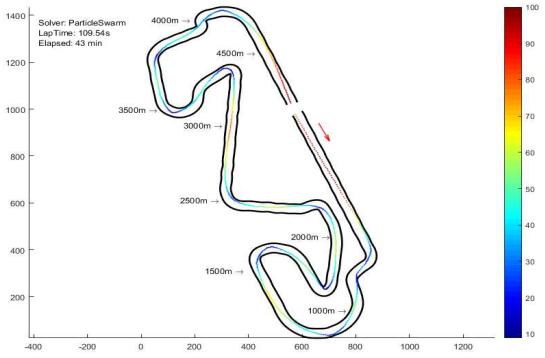


Fig. 15 Particle Swarm result on Catalunya circuit

Particle Swarm gave result 0.6% better than GlobalSearch with much smaller convergence time, and Genetic Algorithm proved to be the worst. Increasing the size of solutions range in case of Genetic Algorithm and Particle Swarm should help to improve this result. Particle Swarm, as the best performing solver, is used as a reference. The time differences of two other solvers are plotted for comparison of how they performed at the same places on track, Fig. 16.

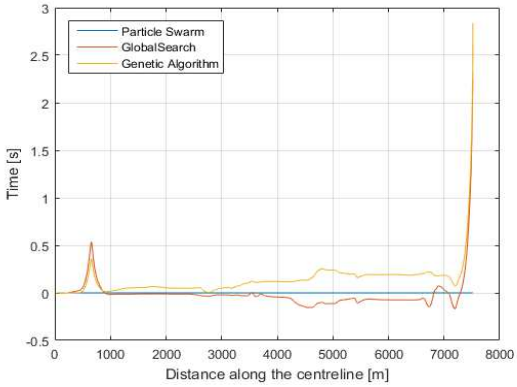


*Fig. 16 Solvers comparison – time vs distance (Silverstone)*

## CONCLUSION AND FUTURE WORK

The algorithm proved to be a working method for finding a sub-optimal trajectory on realistic racing circuits. The convergence time varies from 0.5 up to 2 hours for typical tracks (depending on geometry and used solver), what is a reasonable score.

*Differences between solvers*

Three Matlab solvers were tested to solve the presented problem. Best results were usually obtained with Particle Swarm, which also converges in the minimum time (about 30 – 45min). Genetic Algorithm, although popular for solving nonlinear problems, failed to meet the result of others in most of the simulations. Recommendations for future development and testing are: Path interpolation in a way more resembling the behavior of real car and Usage of realistic simulation for lap time calculation.

This solution can be used to train the driver for new circuit or make simulation in case of obstacles along the track so the driver can manage during the training unpredicted situations.

## REFERENCES


[1] „RaceCloud Blog," [Online]. Available: http://www.racecloud.net/blog/.

[2] „Pacejka Magic Formula," [Online]. Available: http://www.racer.nl/reference/pacejka.htm.

[3] „Roborace website," [Online]. Available: http://roborace.com/.

[4] „Matlab - Mathworks website," Mathworks, [Online]. Available: http://uk.mathworks.com/products/matlab/.

[5] F. C. S. M. E. S. F. Braghin, „Race driver model," Computers & Structures, tom 86, nr 13-14, pp. 1503-1516, Lipiec 2008.

[6] D. L. P. L. L. A. P. B. Luigi Cardamone, „Searching for the Optimal Racing Line Using Genetic Algorithms," w Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, Dublin, Ireland, 2010.

[7] V. G. D. L. a. P. L. L. Matteo Botta, „Evolving the Optimal Racing Line in a High-End Racing Game," w IEEE Conference on Computational Intelligence and Games, Milano, Italy, 2012.

[8] R. S. &. P. S. D. Casanova, „Minimum Time Manoeuvring: The Significance of Yaw Inertia," Vehicle System Dynamics, tom II, nr 34, pp. 77-115, 2000.

[9] A. Rucco, „Computing minimum lap-time trajectories for a single-track car with load transfer," w 2012 IEEE 51st IEEE Conference on Decision and Control (CDC) , Maui, Hi, 2012.

[10] J. P. T. a. D. J. Cole, „Vehicle trajectory linearisation to enable efficient optimisation of the constant speed racing line," Department of Engineering - University of Cambridge, Cambridge, 2012.

[11] „Race Information - Britain 2013," [Online]. Available: https://sidepodcast.com/.

[12] „KML Documentation Introduction - Google," [Online]. Available: https://developers.google.com/kml/documentation/.

[13] „lla2flat - Matlab documentation," [Online]. Available: http://uk.mathworks.com/help/aerotbx/ug/lla2flat.html.