Raceline Optimization

BA Thesis (*Afstudeerscriptie*)

written by

Bart van de Poel
(born April 3rd, 1988 in Amsterdam, Netherlands)

under the supervision of **Christos Dimitrakakis**, submitted in partial fulfillment of the requirements for the degree of

BA Kunstmatige Intelligentie

at the *Universiteit van Amsterdam*.

Abstract

In this thesis a solution approach is described for calculating the minimal time trajectory for a given race track and race car. The trajectory is discretized into a fixed number of points, transforming the problem into a optimization problem with a fixed number of dimensions. The position of these points is then optimized with respect to an estimation function of the actual time needed to traverse the trajectory. For the optimization, a Lipschitzian approach, combined with a local optimization procedure is used. The results show that with respect to the time estimation function, the trajectories generated by this method are an improvement over trajectories generated by taking only the race track into account.

Acknowledgments

I would like to thank my supervisor, Christos Dimitrakakis, for all his help during this project.

Contents

1 Introduction

In race car driving, the time needed to traverse the race track is heavenly influenced by the race line, or trajectory. It is therefore very useful for a driver to know and follow the optimal trajectory. Trajectories can, and have been optimized to minimize distance, lateral acceleration, or other functions for which only information about the race track is needed. However, depending on the race car, those trajectories might not be optimal with respect to time. The time needed to traverse a race track using some trajectory also depends on the characteristics of the race car, like its mass and power. So two different race cars might have two different minimal time trajectories. F. Braghin et al. optimize the trajectory by making a compromise between the shortest distance and least curvature trajectory, using a single weight parameter. This parameter is set, based on the characteristics of the race car [1]. But using a single weight parameter seems to be too simplistic too capture the optimal trajectory for every possible kind of race car.

In this thesis, a solution method is explored that works by directly optimizing the trajectory with respect to an estimation of the traversal time. The problem of optimizing some trajectory within the bounds of a race track is a constrained variational problem.

Section 2.1 shows how the problem can be transformed into a high dimensional optimization problem by discretizing the trajectory into a finite number of points.

Section 2.2 introduces a procedure to estimate the traversal time for a given trajectory and race car. This estimation acts as the cost function for the optimization algorithm.

in Section 2.3 an optimization algorithm is proposed, which is based on the concept of Lipschitzian optimization. Using this algorithm, the parameters by which a trajectory is defined, are optimized.

In the remaining sections the results are shown and discussed, and finally recommendations for future work are made.

2 Solution Method

2.1 Discretization of the trajectory

The problem of finding the optimal trajectory along the race track is a constrained variational problem. In order to solve this problem, it is transformed into an N-dimensional optimization problem by discretizing the trajectory into N points, which we will call control points. These control points can be placed anywhere between the left and right bound of the race track. The coordinates of the control points are controlled by a set of weight parameters $w(n)$, for $n \in 1, 2 \dots N$. The coordinates $p(n)$ of control point n are then defined by Equation 1.

$$p(n) = w\,(n)\,l(n) + (1 - w(n))\,r(n) \tag{1}$$

With $l(n)$ and $r(n)$ being the coordinates of the left and right bound of the track respectively. The relation between the set of weight parameters and the placement of the control points is illustrated in Figure 1.

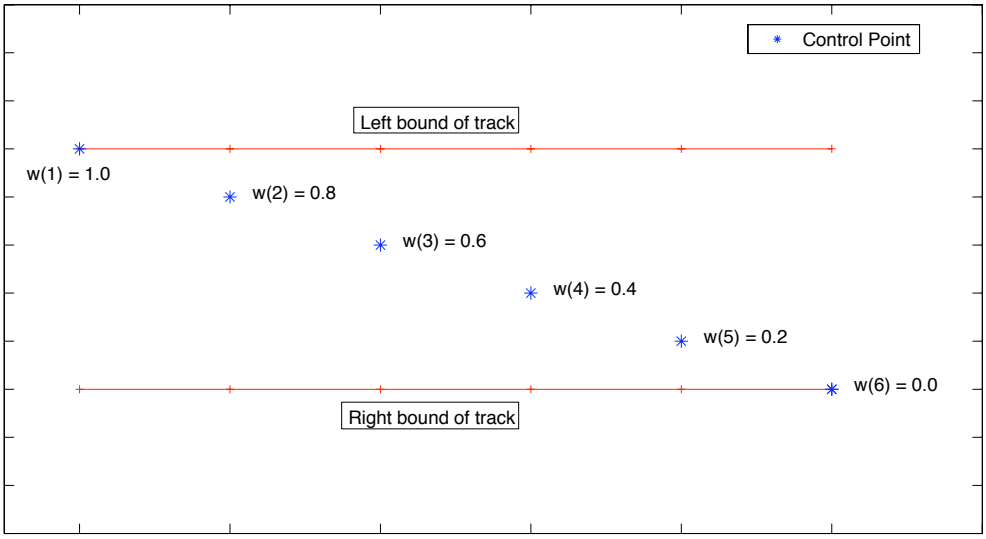


Figure 1: The relation between the weight parameters and the location of the control points

The actual trajectory is defined as a cubic spline fit through the control points. A cubic spline is a combination of polynomials of degree 3, each interpolating between between two data points. The cubic spline $S(\vec{x})$ that consists of the spline segments $S_i(\vec{x})$, has the following requirements:

- For all data points $\vec{x}$, $S(\vec{x}) = f(\vec{x})$
- If spline segment S_i interpolates between data points $\vec{x}_i$ and $\vec{x}_{i+1}$, then $S_{i+1}(\vec{x}_{i+1}) = S_i(\vec{x}_{i+1})$
- If spline segment S_i interpolates between data points $\vec{x}_i$ and $\vec{x}_{i+1}$, then $S'_{i+1}(\vec{x}_{i+1}) = S'_i(\vec{x}_{i+1})$
- If spline segment S_i interpolates between data points $\vec{x}_i$ and $\vec{x}_{i+1}$, then $S''_{i+1}(\vec{x}_{i+1}) = S''_i(\vec{x}_{i+1})$

These requirements ensure that the result is a smooth curve through all data points. This method is an easy and efficient way to fit a smooth curve through any number of control points.

Now we have defined our trajectory as a function of a set of weight parameters, we need to optimize these weight parameters with respect to some cost function in order to find the optimal trajectory. The true cost function that we want to minimize, is the time needed to traverse the trajectory. The cost function that is used within the optimization algorithm is an estimate of this true cost function.

2.2 Estimating the traversal time

In order to optimize the trajectory of a race car along the track, we need a cost function that returns the value of a given trajectory. We want this cost function to be an estimate of the actual time needed to traverse the trajectory. Unfortunately, it is computationally too expensive to use a simulator for this purpose. Instead we can calculate the maximum possible speed and acceleration for each point on the trajectory, using the characteristics of the race car and the curvature. The power of the race car is used to put a bound on the maximum acceleration of the car, whereas the curvature is used to put a bound on both the maximum acceleration and the maximum speed for a given position on the trajectory. Using this information, an estimate of the traversal time is calculated using Algorithm 1 and Algorithm 2.

Algorithm 1 Estimate traversal time: forwards

Input: $Trajectory, \Delta t, g, m, P, v_{gear1}$

$t_0 \leftarrow 0$ {Elapsed Time}.
$v_0 \leftarrow 0$ {Speed}
$x_0 \leftarrow 0$ {Position on along the trajectory}
$i \leftarrow 1$
while End of trajectory is not reached **do**
  $c_i \leftarrow calculateCurvature(x_i)$
  $a_i \leftarrow \min\{a_P(v_{i-1}), a_C(x_i, v_{i-1})\}$ {Acceleration}
  $v_i \leftarrow \min\{v_C(x_i), v_{i-1} + a_i \Delta t\}$
  $v' \leftarrow \min\{v_{i-1}, v_C(x_i)\}$
  $x_{i+1} \leftarrow xi + 0.5(v_i + v')\Delta t$
  $t_i \leftarrow t_{i-1} + \Delta t$
  $i \leftarrow i + 1$
end while
return $\{t_i\}, \{c_i\}, \{v_i\}, \{x_i\}$

Where $v_C(x)$ is the maximum speed at position x, bounded by the curvature, and defined by Equation 2.

$$v_C(x) = \frac{g}{c(x)} \tag{2}$$

With $c(x)$ being the curvature at point x on the trajectory. $a_P(v)$ is the maximum acceleration when moving at speed v, bounded by the power of the race

car. Let $a_P(v)$ be defined by Equation 3.

$$a_P(v) = \min\left\{g, \frac{P}{m \max\{v_{gear1}, v\}}\right\} \tag{3}$$

Where v_{gear1} is the maximum speed in the first gear, and P is the power of the race car in watts. Finally, $a_C(x, v)$ is the maximum acceleration when at position x and moving at speed v, bounded by the curvature. $a_C(x, v)$ is defined by Equation 4.

$$a_C(x, v) = \left\{ \begin{array}{ll} \sqrt{g^2 - v^4 c^2} & g^2 \geq v^4 c^2 \\ 0 & g^2 < v^4 c^2 \end{array} \right\} \tag{4}$$

By using a fixed time step Δt, the trajectory is discretized with respect to time. The position x_t, speed v_t and acceleration a_t are then calculated for every time step until the end of the trajectory is reached. in line 10, the new position x_i+1 is calculated by assuming that the speed changes linearly from the previous to the new speed. The accumulated time is an estimate of the traversal time. This estimation does not take deceleration into account however. It was made under the assumption that that the car can always decelerate from speed v to v', for all $v' < v$. In Algorithm 2 deceleration is taken into account by moving backwards from the end of the trajectory to the beginning, lowering the speed estimations if necessary.

Algorithm 2 Estimate traversal time: backwards

Input: $\Delta t, \{c_i\}, \{v_i\}, \{x_i\}, g$

1: $t \leftarrow 0$
2: **for** $i = size(\{x_i\}) - 1$ to 1 **do**
3: $\quad \delta \leftarrow a_C(x_i, v_i)$
4: $\quad v' \leftarrow v_i + \delta \Delta t$
5: $\quad v_{i-1} \leftarrow \min\{v_{i-1}, v'\}$
6: $\quad t \leftarrow t + ((x_i - x_{i-1})/(0.5(v_{i-1} + v_i)))$
7: **end for**
8: **return** t

In line 4, the speed v' is calculated for the previous position on the trajectory, given that we move backwards with a maximum acceleration of δ. So then v' is the maximum speed for the previous position, given that the car needs to decelerate to the current speed. For all speeds greater than v', the car will not have enough time to decelerate to the current speed. In line 5 the speed for the previous position is lowered if it is greater than the maximum speed v'.

For an accurate estimation of the traversal time, Δt has to be set to a low enough value. This value has been set to 0.1 seconds to keep the computation time reasonable. In Figure 2 the convergence of the time estimation procedure is plotted, showing that setting Δt to 0.1 is low enough to get close to the converged time estimation.

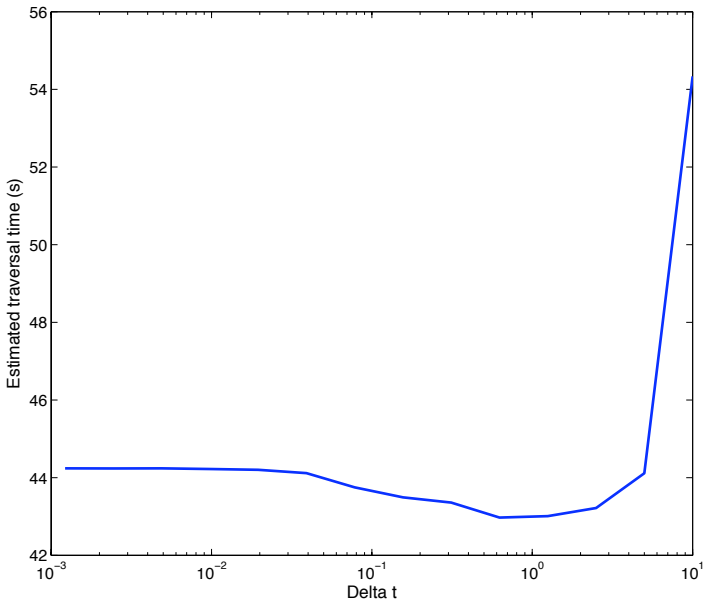


Figure 2: Convergence of the time estimation procedure

So now we have a cost function which we can use to optimize the position of the control points. However, the downside to this estimation procedure is that changing the weight of a control point can cause a non-local change in the time estimation, because the maximum speed at some point on the trajectory depends partly on the speed of the previous point. This means that it is not possible to differentiate this function to calculate the gradient. The gradient would help to guide the search through the high-dimensional search space, and not having it available complicates the problem significantly.

2.3 Lipschitzian optimization

The approach that was taken to optimize the weights of the control points with respect to the traversal time, is to divide the search space into subspaces and take a sample from each subspace. During each iteration a procedure is called to pick a subspace that will then be divided again. This approach is based on Lipschitzian optimization[2]. In Lipschitzian optimization, the Lipschitz property is assumed:

$$|f(x) - f(x')| < K\, d(x, x') \tag{5}$$

for some metric measure d and some constant $K > 0$. The Lipschitz constant K can be seen as a bound on the rate of change of the function f. When

this constant is known, we can calculate a lower bound on the cost function for each subspace of the search space. These lower bounds are used to pick the subspace that will be subdivided. Figure 3 illustrates this concept. In this one-dimensional problem, the search space is divided into two subspaces and two samples x_1 and x_2 are taken from the center of the subspaces. Using the Lipschitz constant K, a lower bound is calculated for each subspace, and the subspace with the lowest lower bound will be chosen to be divided again. In this example problem, subspace 1 would be chosen.

The lower bound of a subspace is proportional to the value of the sample that was taken, and inversely proportional to the size of the subspace. So on the one hand, local search is done by dividing subspaces with low sample values, and on the other hand global search is done by dividing large unexplored subspaces.

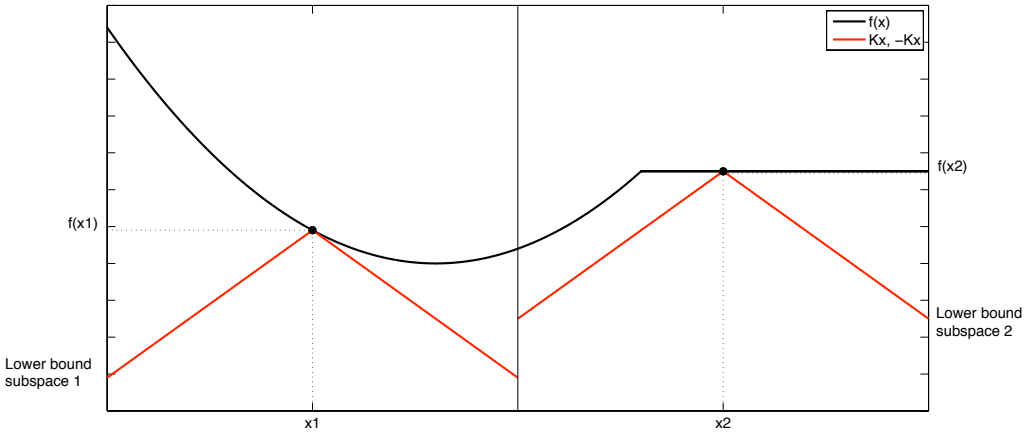


Figure 3: Illustration of using the lipschitz constant to determine a lower bound for a subspace

2.3.1 Setting the Lipschitz constant

Lipschitzian optimization methods rely on the knowledge of the Lipschitz constant of the target function. In the case of our cost function, which estimates the traversal time for a given trajectory, we don't know the value of the Lipschitz constant. In order to use Lipschitzian optimization we need to estimate the value of this important parameter. All approaches in this thesis make use of a fixed value for the Lipschitz constant, which is set manually. In Section 7 a possible improvement to this simplistic approach is described.

2.3.2 Dividing the search space

In the example shown in Figure 3, the search space was divided into two subspaces of equal size. There are however many more ways to handle this division. When dividing the search space, we must take into account that it is a high dimensional space. It will not be possible to divide along all dimensions in a single division, since the number of subspaces will grow exponentially with respect to the number of dimensions. If we were to optimize weight parameters for 40 control points by splitting the search space down the middle for every dimension, then we would end up with 2^{40} subspaces after only one division. Instead, two other approaches were evaluated.

The first approach is to choose one dimension, and to divide the search space in half along that dimension. The dimension could be chosen randomly, but we made the choice to divide along the dimension with the biggest size. This way, the size of the search space is decreased evenly along all dimensions. Using this division strategy results in two new subspaces after each division, regardless of the number of dimensions. In the context of optimizing the position of our control points, this approach can be viewed as constraining one control point to a smaller area of the race track, which is illustrated in Figure 4. In this illustration the initial search space is unconstrained. Then the fourth control point is chosen, and is constrained to the top half of the track in the first subspace, and constrained to the bottom half of the track in the second subspace.

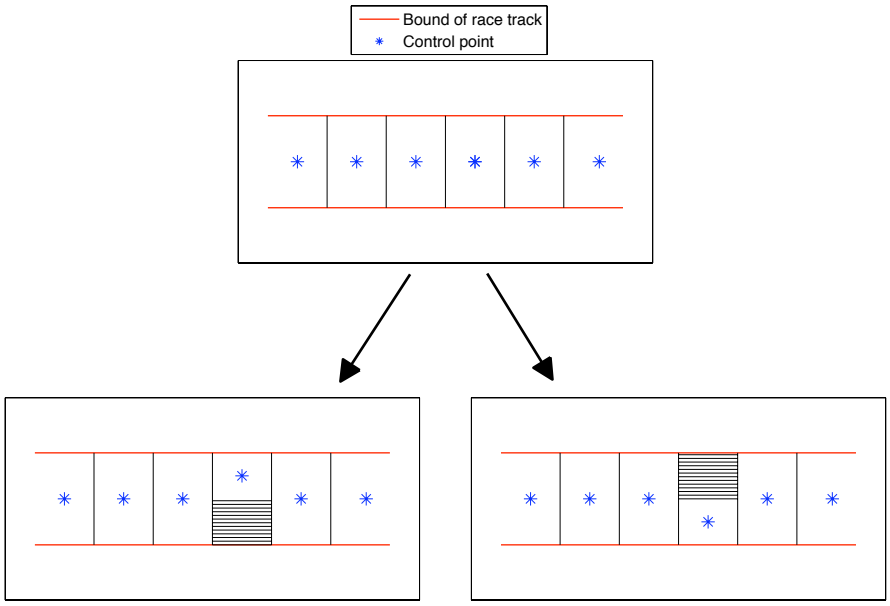


Figure 4: Constraining the search space for a control point

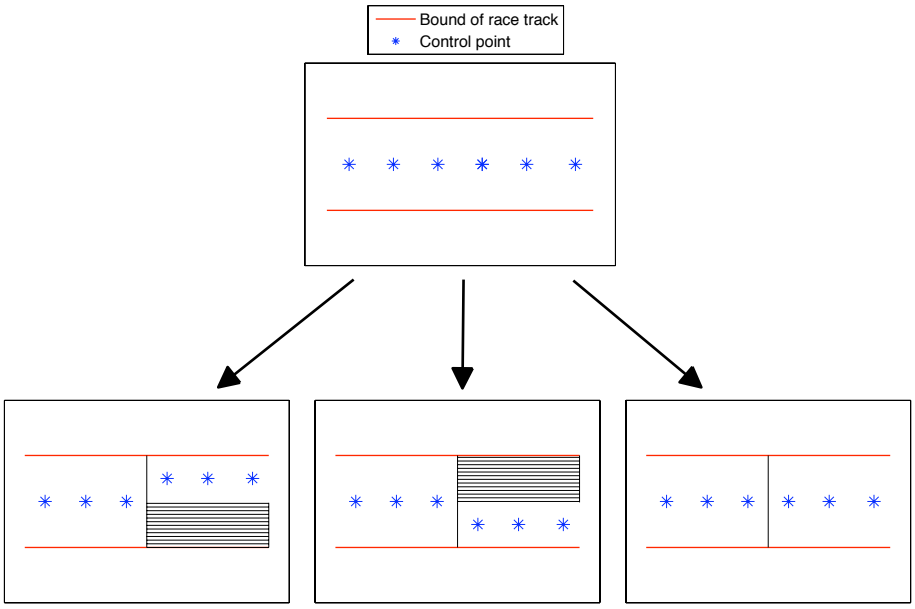


Figure 5: Constraining the search space for sets of control points

The other approach that was evaluated, is not to divide the search space along just one dimension, but along a set of dimensions instead. In our case, this set of dimensions is a set of consecutive control points. Initially there is one set, containing all control points. During the division of the search space, this set is divided into two smaller sets, and stricter constraints are given to one of them. Constraining a set of consecutive control points is intuitively pleasing since in an optimal trajectory the position of a control point is related to the position of other control points nearby. So using this division strategy could speed up the search and help reach better solutions. Figure 5 shows an example where a set of six control points is split into two sets containing three consecutive control points each. The constraints for one of the sets are then updated, resulting in two subspaces of the search space. The third subspace is the result of splitting the set of control points, but retaining the original constraints. This way, in a later step constraints can be set on a smaller set of control points.

2.3.3 Sampling from a subspace

Since we do not have a gradient for our approximation of the true cost function, we need to guide our search by taking samples. After dividing the search space, a sample is taken from each of the subspaces. These samples will represent the cost function for their respective subspace, and their value will decide on which of the subspaces gets picked to be divided again. So choosing a sampling

strategy is very important. As before, there are many strategies possible for sampling. For this thesis, three approaches were explored.

The first approach is also the simplest approach. All samples are taken from the center of the subspace. This an easy and efficient sampling method, although for larger subspaces, a single sample taken from the center of a space might not be a good representative.

Since we are looking for the subspace that holds the optimal parameters, we would like a sample to be as close as possible to the optimal point in its subspace. This will help the algorithm to focus on the right subspaces. For this reason, the second approach is to set a sample by performing a simple local search within the subspace. In Figure 6 the difference between using center sampling and local search is illustrated.

The local search can be done with a simple procedure that moves back and forth along the trajectory, changing the position of each control point, and determining if the value of the cost function decreases as a result. The amount of change in the position of the control point is initially set high, but is decreased when no more improvements can be made to the control points. This will allow the procedure to move from rough to fine optimization. The basic procedure is shown in Algorithm 3.

The procedure given in Algorithm 3 "sweeps" through the trajectory, evaluating our time estimation function several times for each control point. Because many function evaluations have to be done, sweeping through the trajectory is computationally expensive. So in order to keep the global algorithm tractable, we have to constrain the number of sweeps during local search to a small value, which limits the performance of the global optimization algorithm. It is however not necessary to perform local search using the same cost function as the one we are optimizing. If we can get a sample that is closer to the minimum of the subspace using some other cost function, then this will also help to guide the global search.

Algorithm 3 Simple procedure for optimizing the cost function

Input: $\{cp\}, \Delta w, w_{min}, limit$

1: $count \leftarrow 0$
2: $Cost \leftarrow evaluateCost(cp)$
3: **repeat**
4: **for** $i = 0$ to $|cp|$ **do**
5: $cpPlus \leftarrow cp$
6: $cpMin \leftarrow cp$
7: $cpPlus_i \leftarrow cpPlus_i + \Delta w$
8: $cpMin_i \leftarrow cpMin_i - \Delta w$
9: $CostPlus \leftarrow evaluateCost(cpPlus)$
10: $CostMin \leftarrow evaluateCost(cpMin)$
11: **if** $CostPlus < Cost$ **then**
12: $Cost \leftarrow CostPlus$
13: $cp_i \leftarrow cpPlus_i$
14: **end if**
15: **if** $CostMin < Cost$ **then**
16: $Cost \leftarrow CostMin$
17: $cp_i \leftarrow cpMin_i$
18: **end if**
19: **end for**
20: **for** $i = |cp|$ to 0 **do**
21: Repeat lines 5 through 18
22: **end for**
23: **if** no change in $\{cp\}$ **then**
24: $\Delta w \leftarrow 0.5\Delta w$
25: **end if**
26: $count \leftarrow count + 1$
27: **until** $count > limit$, or $\Delta w < w_{min}$.
28: **return** $\{cp\}, Cost$

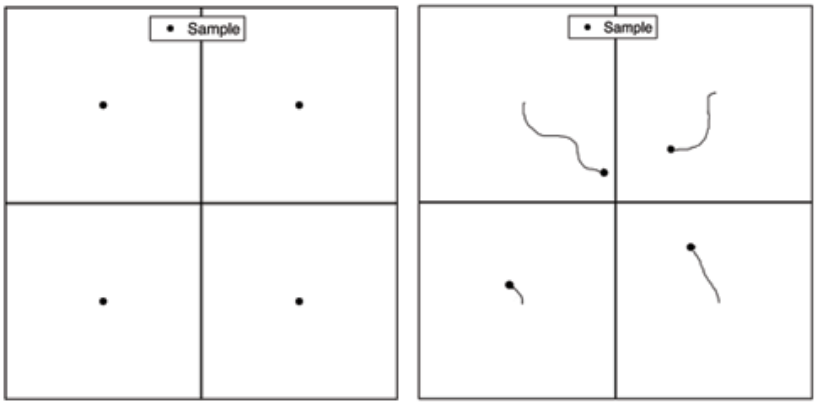
An alternative cost function for the local search, is minimizing the sum of the squared lateral accelerations for all the control points, which is calculated under the assumption that the car is moving with a constant speed. The lateral acceleration vector for a control point can be calculated based on its coordinates p, using Equations 6 and 7.

$$a(i) = \frac{u(i) - u(i-1)}{||p(i) - p(i-1)||} \tag{6}$$

with

$$u(i) = \frac{p(i+1) - p(i)}{||p(i+1) - p(i)||} \tag{7}$$

A more detailed explanation of the calculation of the squared lateral acceleration is given by Dimitrakakis[3]



(a) Sample from the center (b) Sample using local search

Figure 6: Illustration of two sampling methods

The alternative cost function is then defined by Equation 8 as the sum of lateral accelerations of all the control points, multiplied by the distance to the next control point.

$$Cost = \sum_{i=0}^{N} ||a(i)||^2 \, d\big(p(i), p(i+1)\big) \tag{8}$$

where $d(p, p')$ is the distance between point p and point p'

In most cases, minimizing this cost will not lead to optimal trajectories, but it will often improve the current solution and give us a better estimate of the optimal solution of a subspace than the center sampling approach would give. The advantage of using this cost function is that it is very simple and computationally inexpensive. It also has the property that changing the position of one control point only affects the lateral acceleration of the control points next to it, which makes recalculating the cost function even more efficient and gives us the opportunity to perform many sweeps along the trajectory during local search. Optimizing this alternative function can be done using Algorithm 3, but instead of evaluating the primary cost function, *evaluateCost* now evaluates the squared lateral acceleration.

3 Experimental Setup

The traversal time estimation function introduced in Section 2.2 was used to evaluate trajectories. In the experiments, the weight parameters were initialized to produce a trajectory along the center of the race track: $w(n) = 0.5$, for all $n \in \{1, \ldots, N\}$, with N being the number of control points. The results shown in Section 4 have been produced by averaging the results of experiments on a number of different race tracks and different race cars. Four types of cars were used for testing:

- **High power car:** power of 400.000 watts, mass of 800 kg, and first gear maximum speed of 100
- **Medium power car:** power of 200.000 watts, mass of 1000 kg, and first gear maximum speed of 70
- **Low power car:** power of 100.000 watts, mass of 1200 kg, and first gear maximum speed of 40
- **Very low power car:** power of 10.000 watts, mass of 1000 kg, and first gear maximum speed of 10

The Lipschitz constant was set to different fixed values for the different approaches and race tracks, based on which values showed the best performance. Δt in the traversal time estimation function was set to 0.1 seconds.

4 Results

4.1 Sampling

Three different sampling methods have been evaluated and compared to determine which method shows the best performance. For the evaluation the algorithm made use of the simpler search space division approach, which divides the space along a single dimension. Figure 7 shows the results of this evaluation.

Sampling from subspaces by locally minimizing the squared lateral acceleration clearly shows the best performance, whereas using local search to minimize the primary cost function shows the worst performance. It eventually reaches the same solution as the center sampling approach does, but only after many cost function evaluations. The large difference in performance between the two local search approaches can be explained by the fact that the local search procedure that minimizes the primary cost function is computationally very expensive. It visits the control points one by one, taking two samples for each control point. So for each traversal along the trajectory, the primary cost function has to be evaluated $2|cp|$ times, with $|cp|$ being the number of control points. In order to keep the algorithm computationally tractable, only a couple traversals can be made along the trajectory, which greatly limits the advantage of performing local search. The alternative cost function on the other hand, is very efficient and

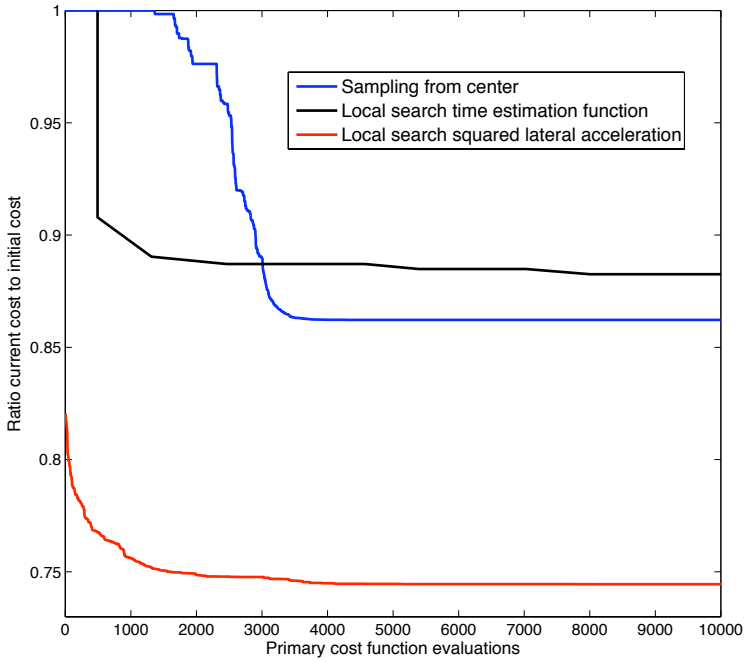


Figure 7: Evaluation of different sampling methods

can be evaluated many times during local search without having a big impact on the computation time.

4.2 Division of the search space

In Section 2.3.2 two methods for dividing the search space were discussed. Figure 8 shows the results of their evaluation. Against expectation, dividing the search space by constraining sets of control points shows a slightly worse performance compared to constraining single control points. The difference is very small however, and due to the limited number of race tracks tested on, it could be caused by randomness. A possible explanation for the disappointing performance of the division method based on constraining sets of control points, is the fact that in the algorithm a random set of control points is chosen to be constrained. Ideally we would like to choose a set of control points based on some selection criterion. By choosing a set randomly, this approach is not used to its full potential.

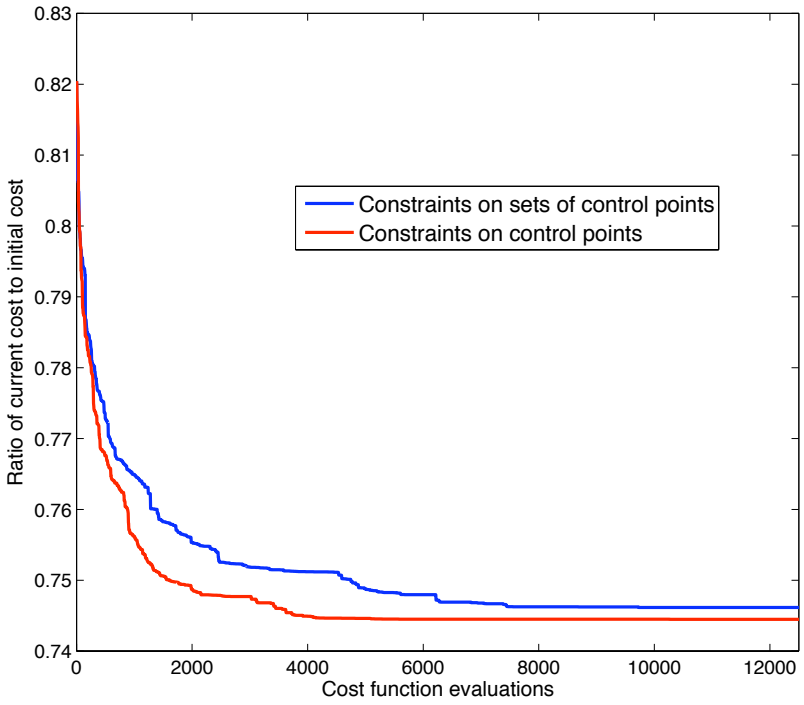


Figure 8: Evaluation of two approaches to dividing the search space

4.3 Improving the resulting trajectory using local search

The results in section 4.1 show that due to its computational inefficiency, the local search procedure that minimizes the primary cost function, is not suitable to be used within the algorithm to sample from the search space. Besides using the procedure as a sampling method, we can also use it as a standalone method for optimization. Figure 9 shows the performance of this method compared to the performance of the main algorithm.

This comparison shows that the local search procedure also does not perform well as a standalone optimization algorithm, because it gets easily stuck in suboptimal trajectories. But when we initialize the local search procedure with the solution produced by the main algorithm, then it actually makes a significant improvement. This is illustrated in Figure 10.

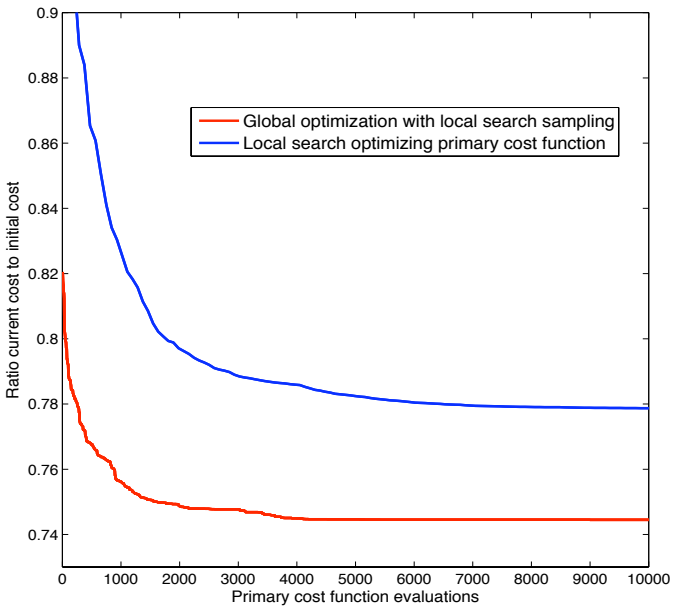


Figure 9: Standalone local search procedure compared to the Global optimization algorithm that used local search for sampling

So based on these results, we can say that the best performing approach is to:

- Divide the search space by setting constraints on single control points
- Sample from the search space by locally minimizing the squared lateral acceleration
- Improve the best solution by locally minimizing the traversal time estimation function

We can now compare the performance of trajectories produced by the best performing approach to the performance of those produced by minimizing the squared lateral acceleration, as shown in Figure 10. From these results we can conclude that a significant improvement has been made by optimizing the trajectory with respect to a traversal time estimation function that takes the characteristics of the race car into account. And because the traversal time estimation function is an approximation of the true cost function, we can also conclude that it is very likely that the trajectories that are better with respect to this estimation, are also better with respect to the true cost function.

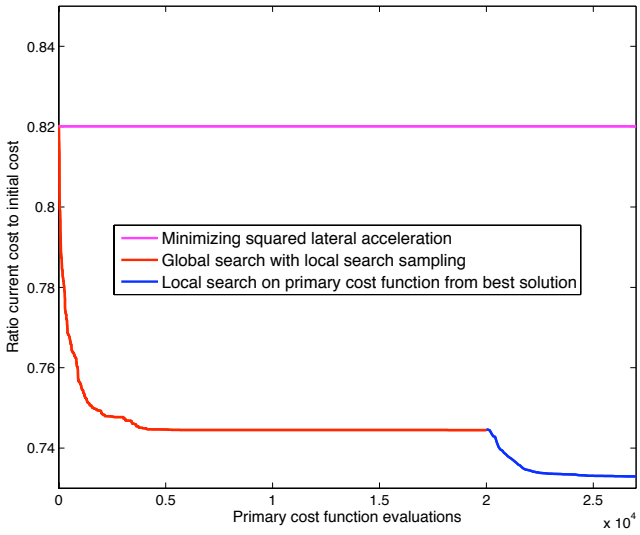


Figure 10: Improving the best solution with one last local search using the primary cost function

4.4 Example trajectories

Figure 11 shows an example race track with two optimized trajectories. The first trajectory is optimized by just minimizing the squared lateral acceleration. The second trajectory on the other hand, is optimized with respect to the time estimation function for a high power race car. The main difference that can be observed is that the second trajectory is much smoother around the corners of the race track. This way, the race car does not have to decelerate as much, resulting in a lower traversal time according to the time estimation function.

Figure 12 shows the difference between the calculated trajectories for high and low power cars on another example track. The difference between the two trajectories at the first straight part of the race track can be explained by the fact that the low power car needs more time to reach high speeds. So when going into the first curve of the race track, it is still moving at a relatively low speed, allowing him to take the shortest distance path. Coming out of the second curve of the racetrack, another difference can be observed. The low power car does not traverse the curve as tightly as the high power car does. This makes sense because the low power car has less power to accelerate, so it benefits more from a trajectory that allows it to retain its speed than it would from a trajectory where it needs to slow down and start accelerating again. The high power car on the other hand, does have a lot of power to accelerate, and can take the curve less widely.

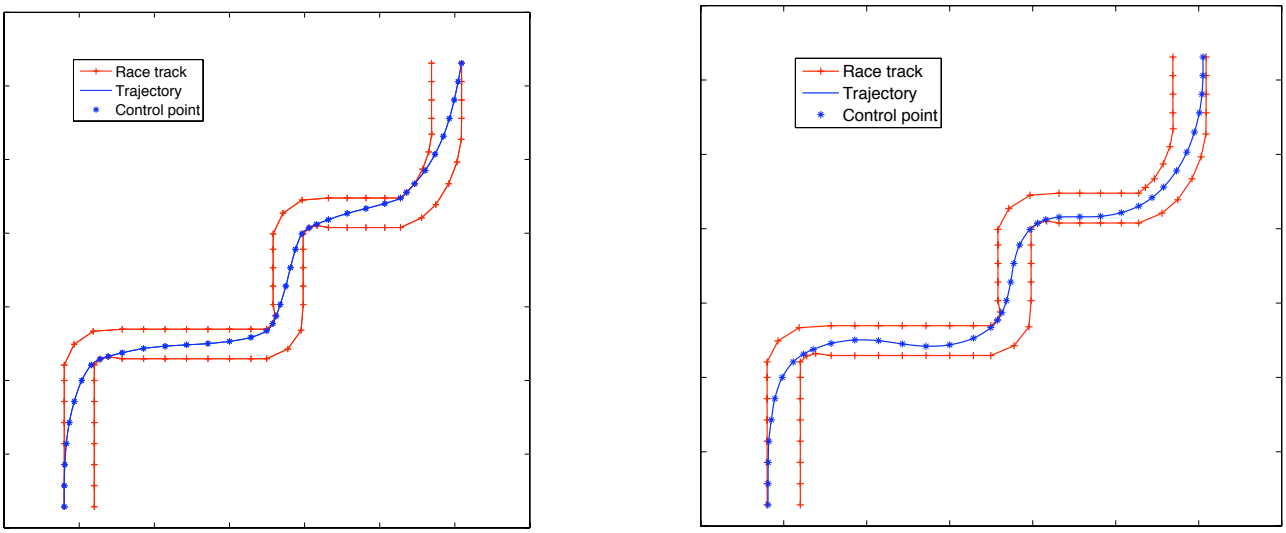


Figure 11: Comparison of a minimal squared lateral acceleration trajectory (left) and a trajectory optimized for a high power car with respect to the time traversal estimation function (right)

5 Discussion

Though the results in Section 4 show that the algorithm proposed in this thesis finds good trajectories that are an improvement over trajectories obtained by more simplistic approaches, it also seems that in many cases, the algorithm does not quite reach the optimal solution within the number of iterations tested on. With some further improvements, the algorithm should be able to produce even better trajectories.

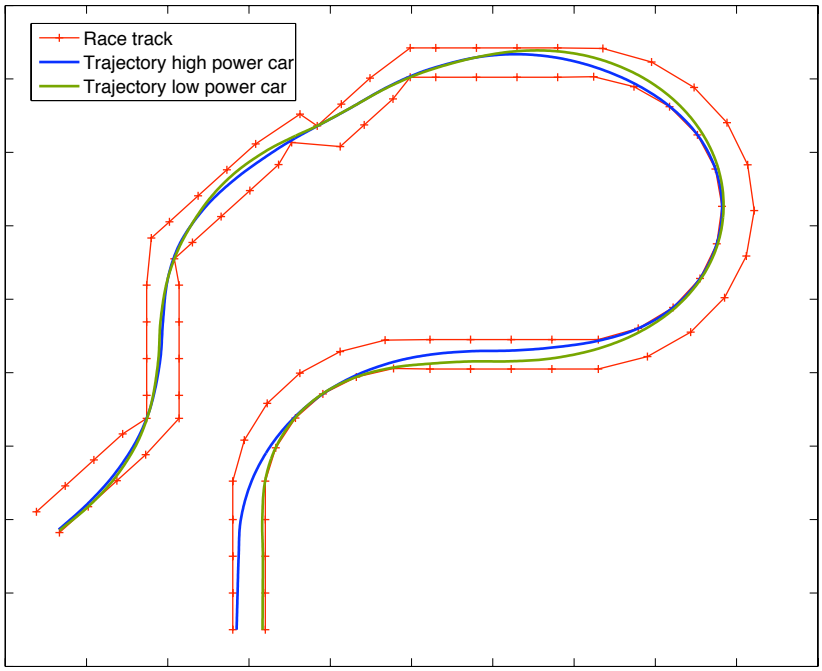


Figure 12: Optimized trajectories for high and low power cars

6 Conclusion

In this thesis, the problem of optimizing trajectories given information about the characteristics of the race car was explored. A solution method based on Lipschitzian optimization was proposed and different variations of this approach were evaluated. The experiments show that the best results can be obtained by:

- Dividing the search space by setting constraints on single control points
- Sampling from the search space by locally minimizing the squared lateral acceleration
- Improving the best solution by locally minimizing the traversal time estimation function

The final results were compared to the more simplistic approach of minimizing the squared lateral acceleration of a trajectory. This comparison shows that

a significant increase in performance can be achieved by taking the characteristics of the race car into account when optimizing trajectories. Even though the resulting trajectories are a significant improvement, in many cases they do not seem to be the actual optimal trajectory. With some more work on the algorithm, even better results could be obtained.

7 Future Work

The current algorithm makes use of a fixed Lipschitz constant that is set manually. This constant is a difficult parameter to set, because it can be different for every race track and race car. In many cases this constant will be set to a suboptimal value, limiting performance of the algorithm. Future work on this algorithm could address this problem by somehow automatically estimating the Lipschitz constant. Since the Lipschitz constant is defined as a bound on the rate of change of a function, a possible solution approach for this problem would be to base the estimate on the value of samples taken in the course of the algorithm, and the distance between them in the search space. With these two values, the rate of change can be calculated and used to make an estimation of the maximum rate of change.

Dividing the search space by constraining sets of consecutive control points instead of single control points showed disappointing results. As explained in Section 4.2, this can be explained by the fact that sets are randomly chosen. The algorithm could be improved by researching ways to choose sets of control points based on some value estimate. This would help the algorithm to ignore sets of control points that show little promise to improve the solution, and thus increase its performance.

In the current algorithm, when a search space is divided into subspaces, the original space is never looked at again. The lower bound that is calculated for a subspace is based on just one sample. A possible variation worth investigating, is to use the lower bounds calculated for subspaces to adjust the lower bound of their parent space. This could be done by propagating the lower bound of a subspace upwards to its ancestors. This way, every time a sample is taken for a subspace, it will improve the estimate of the lower bound of ancestor spaces.

In Section 2.2 the problem of not being able to calculate the gradient for the primary cost function was discussed. Having the gradient available would be helpful in our search through the search space. A possible adjustment to our solution method that will allow us to calculate a gradient, would be to parameterize trajectories not only by a set of weights, but also by a set of speeds at the control points. Making the speed at a control point one of the parameters of the trajectory would solve the problem of not being able to calculate the gradient. Because changing the value of a parameter would no longer cause a non-local change in the time estimation.

References

[1] E. Sabbioni F. Braghin F. Cheli and S. Melzi. Race driver model. *Computers and Structures*, 86, 2007.

[2] C. D. Perttunen D. R. Jones and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Application*, 79, 1993.

[3] Christos Dimitrakakis. Online statistical estimation for vehicle control: A tutorial. 2001.