Learning to drive using convex-convave programming

Alexandre Carlier
EPFL, Switzerland
alexandre.carlier@epfl.ch

Abstract

The nowadays standard deep learning approach for autonomous driving guarantees neither optimality of the outputs nor safety under all situations. Using an explicit optimization formulation for different driving scenarios can be an alternate method to handle this limitation. We show that convex-concave programming, and in particular the DCCP framework, can be used to solve those multi-time-step optimization problems, involving a single or several agents, when the feasible set is non-convex. All the code used in this work will be made available at https://github.com/alexandre01/ConvexConcaveDriving.

1. Introduction

Recent approaches for self-driving cars make use of deep learning (e.g. using gradient descent on a deep neural network), by learning from large datasets of real or simulated trajectories. This has the advantage to capture a human-like driving style. However, the outputs of deep neural networks are notoriously difficult to interpret and we don't have any guarantee that such a system will be efficient and safe in any situation. Using an explicit optimization formulation can be a solution to handle this problem. In this work, we will explore how convex-concave programming can be used to solve typical driving scenarios. While the objective of minimizing the distance to a target point is convex, the constraints to avoid obstacles or stay on-road are typically non-convex. We will see how one can formulate these problems as *Difference of convex* (DC) problems, and solve them automatically using the `DCCP` solver introduced in [3].

2. Related work

While convex optimization can model and solve a very broad range of engineering problems, we need other tools when non-convex problems occur. One example of these are Difference of convex (DC) problems, which can be formulated as follows:

$$\begin{aligned} &\text{minimize } f_0(x) - g_0(x) \\ &\text{subject to } f_i(x) - g_i(x) \leq 0, \quad i = 1, \quad \ldots, m \end{aligned} \tag{1}$$

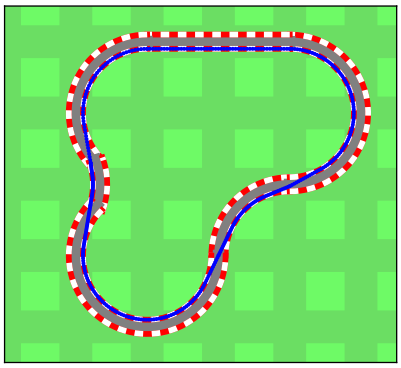


Figure 1: Local solution (in blue) to the optimization problem that minimizes the path length while staying on the road. Note how the optimal trajectory drives alongside the inner lane, except when "concave" bends occur.

where the functions $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ and $g_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ are convex. The convex-concave procedure (CCP) is one algorithm to find local optima of (1). The basic idea of this procedure is explained in algorithm 1 as introduced in [2], which convexifies the DC problem by linearizing the concave terms $-g_0(x)$ with their first order approximation. The transformed problem can then be solved optimally using the standard convex methods.

Algorithm 1: Basic CCP algorithm

Input: initial feasible point x_0
k := 0;
repeat
1. Convexify. Form $\hat{g}_i(x, x_k) = g_i(x_k) + \nabla g_i(x_k)^T(x - x_k)$;
2. Solve. Set the value of x_{k+1} to a solution of the convex problem

$$\begin{aligned} &\text{minimize } f_0(x) - g_0(x) \\ &\text{subject to } f_i(x) - g_i(x) \leq 0, \quad i = 1, \quad \ldots, m \end{aligned}$$

3. Update iteration. k := k+1
until *stopping criterion is satisfied*;

This procedure requires an initial feasible solution, which

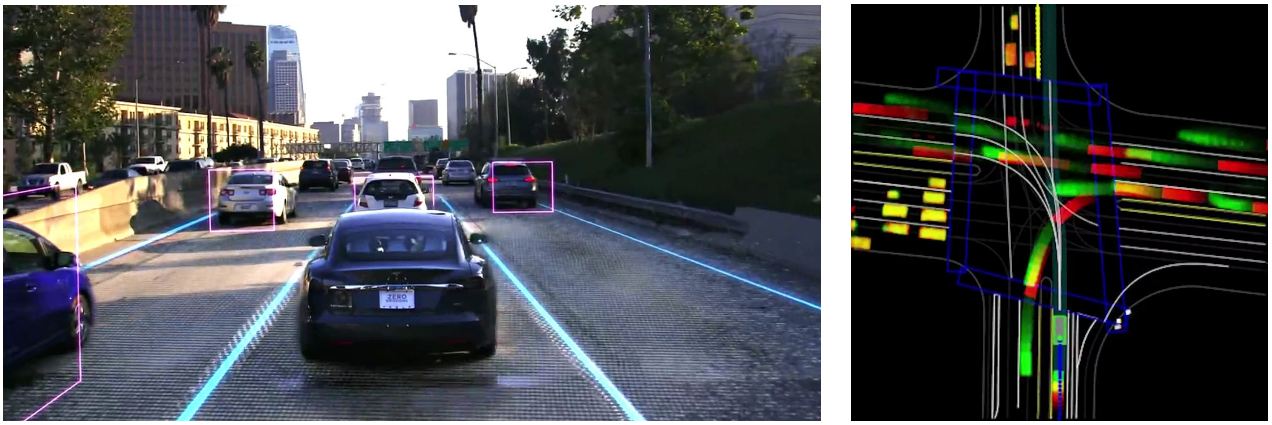


Figure 2: On the left: Tesla's autonomous driving system; in particular, it detects other vehicles, pedestrians and lanes. On the right: Waymo's ChauffeurNet [1] using a top-view abstract model of the road.

can be hard to find. Note however that a penalty CCP method was also introduced in [2], which removes the need for an initial feasible point, by relaxing the problem with slack variables and penalizing the sum of the violations.

Finally, the work by Shen et al. [3] introduces a new automated solver called `DCCP`, by combining the ideas of CCP and Disciplined convex programming (DCP). Disciplined convex programs have the form:

$$\begin{aligned} &\text{minimize/maximize } o(x) \\ &\quad \text{subject to } l_i(x) \sim r_i(x), \quad i = 1, \ldots, m \end{aligned} \tag{2}$$

where o is convex for a minimization problem and concave otherwise, and l_i, r_i are convex or concave depending on the relational operator $\sim$ which denotes one of $=$, $\leq$ or $\geq$.

This formulation leads to the *disciplined convex-concave program* (DCCP) which has the exact same form as the DCP problem (2), but without the convex/concave rules depending on the $\sim$ operator. This greater flexibility comes with the cost of having to use the heuristic described in algorithm 1 to find a local optimum. Note however that when the original problem is in the DCCP form, its convexification is in the DCP form, and each iteration of algorithm 1 can thus be solved using a standard DCP solver, like CVX of CVXPY. The DCCP implementation[1] introduced in [3] is written in Python and extends CVXPY.

In the following, we will analyze and solve different driving scenarios with a top-view description of the scene. Note that is not an unrealistic setting, since industry implementations [1] tend to use a similar abstract world representation to navigate the roads, as it leads to faster training of neural networks and better interpretability (see figure 2). Such a top-view segmentation can be obtained using the raw camera signal, detecting vehicles, lanes and other keypoints as a preprocessing step.

[1] https://github.com/cvxgrp/dccp

3. Experiments

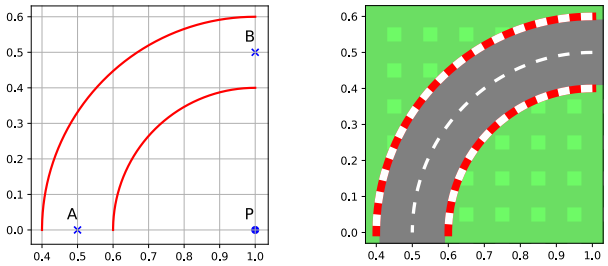
In this section, we will solve increasingly complex problems using the DCCP framework, as explained in section 2.

A. Simple road segment

We first solve the problem of shortest path from a point A to B, while staying on the road. The road we will use for this example is curvy, meaning that the set delimited by the left and right lanes is non-convex, as shown in figure 3. The path is parametrized by $n+1$ points $x_0, \ldots, x_n$, where $x_0 = A$ and $x_n = B$. We denote $v_i = x_i - x_{i-1}$ for all $1 \leq i \leq n$ the speed at every time-step, and impose the additional constraints of speed limit: $\|v_i\| \leq \Delta$ and staying on-road: $r_{\text{right}} \leq \|x_i - P\|_2 \leq r_{\text{left}}$. The problem can be summarized as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} \|v_i\|_2 \\ \text{subject to} \quad & x_0 = A, x_n = B \\ & \|v_i\|_2 \leq \Delta, \quad i = 1, \ldots, n \\ & \|x_i - P\|_2 \leq r_{\text{left}}, \quad i = 0, \ldots, n \\ & r_{\text{right}} \leq \|x_i - P\|_2, \quad i = 0, \ldots, n \end{aligned} \tag{3}$$

Since the constraint-set of problem (3) is non-convex, it cannot be written as a disciplined convex program (DCP) and hence can't be solved using a generic solver like `CVX`.



(a) (b)

Figure 3: (a): Plot of a simple curvy road. The goal is to go from A to B using the shortest path that stays on the road. Note that the set delimited by the 2 lanes is non convex. (b): Same plot, stylized using Python's Matplotlib library.

If, however, we removed the last constraint, namely $r_{\text{right}} \leq \|x_i - P\|_2$, the problem would become a canonical QCQP problem (quadratically constrained quadratic program). We can actually first try to solve optimally the convexification of this problem, by replacing the original feasible set with its convex-hull, as shown in figure 4. The optimal value d^* can be interpreted as the dual optimal value, and is obviously a lower-bound of the distance needed for the convex-concave constrained problem. However, the shape of such a road is not realistic anymore and the associated optimal trajectory would correspond to driving off-road in the original road.

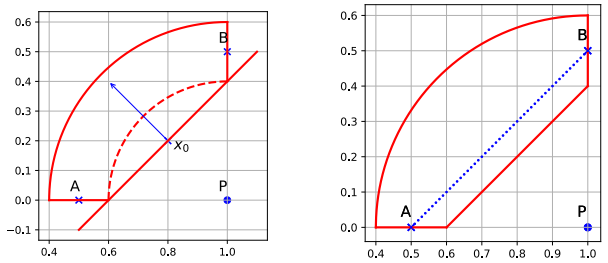
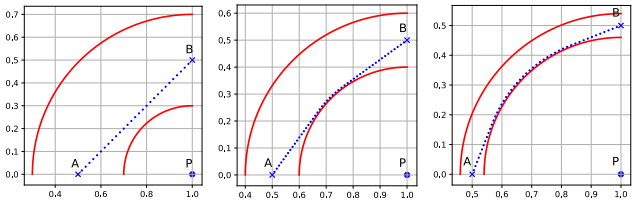


Figure 4: Convex hull of the feasible set delimiting the road segment described in figure 3. By replacing the constraint by this one, we can solve the optimization problem optimally. The solution (a straight line) is however less interesting than the curve obtained in the convex-concave program (see figure 5).

By using the DCCP framework, we find a local optimum to this problem, which we depict in figure 5.

B. Complex Formula 1 circuit

In order to solve a road trajectory optimization on a closed F1-like circuit, we cut the road in 6 chunks, each one being similar to the simple segment described in part A. We denote $x_i^{(j)}$ the position of the vehicle at time-step i and



(a) (b) (c)

Figure 5: Solution to the problem (3), with an increasingly tight road width.

on the j^{th} chunk of the circuit, $0 \leq i \leq n, 1 \leq j \leq 6$, and $v_i^{(j)}$ its associated velocity vector. The problem formulation becomes:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{6} \sum_{i=1}^{n} \left\| v_i^{(j)} \right\|_2 \\
\text{subject to} \quad & \left\| v_i^{(j)} \right\|_2 \leq \Delta, \quad 1 \leq i \leq n,\ j = 1, \ldots, 6 \\
& r_{\text{right}} \leq \left\| x_i^{(j)} - P_j \right\|_2 \leq r_{\text{left}}, \quad 0 \leq i \leq n,\ j = 1, 5 \\
& 1 - r_{\text{left}} \leq \left\| x_i^{(j)} - P_j \right\|_2 \leq 1 - r_{\text{right}},\ 0 \leq i \leq n,\ j = 2, 4, 6 \\
& 2 - r_{\text{left}} \leq {x_i^{(3)}}_y \leq 2 - r_{\text{right}}, \quad 0 \leq i \leq n \\
& x_0^{(1)} = x_n^{(6)} = A \\
& {x_n^{(1)}}_x = B_x, {x_0^{(2)}}_x = B_x, {x_n^{(2)}}_x = C_x \\
& {x_0^{(3)}}_x = C_x, {x_n^{(3)}}_x = D_x, {x_0^{(4)}}_x = D_x, {x_n^{(4)}}_x = E_x \\
& {x_0^{(5)}}_x = E_x, {x_n^{(5)}}_x = F_x, {x_0^{(6)}}_x = F_x
\end{aligned}
\tag{4}
$$

The optimal solution is shown in figure 1. Note that while we impose $x_0^{(1)}$ and $x_n^{(6)}$ to be exactly on the point A, the other checkpoints B, C, D, E and F only impose the trajectory to go through their corresponding x or y coordinate. Otherwise, we would force the trajectory to go through the

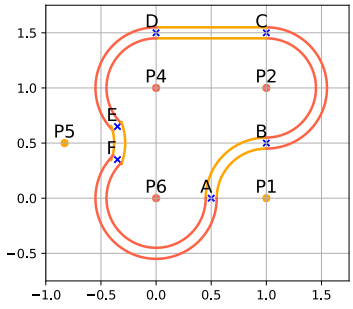
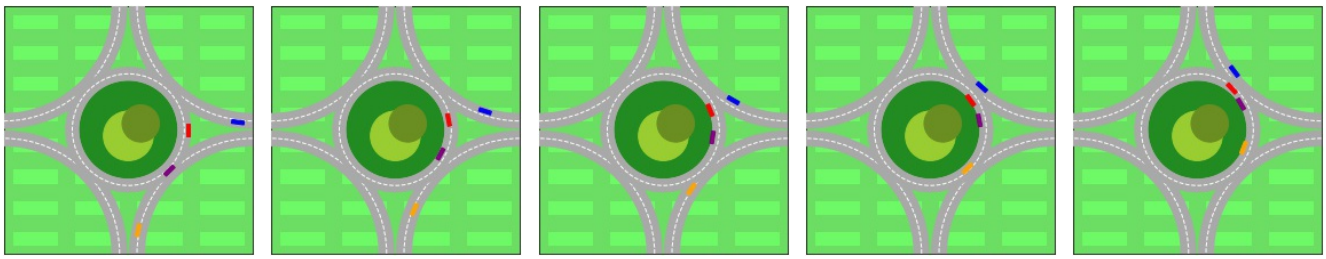
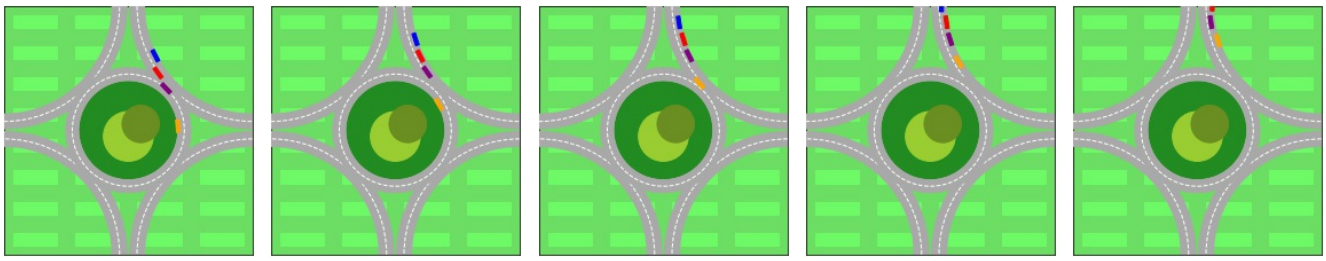


Figure 6: Plot of the complex F1 circuit, as described in section B.



(a) $t = 0$ (b) $t = 5$ (c) $t = 10$ (d) $t = 15$ (e) $t = 20$



(f) $t = 25$ (g) $t = 30$ (h) $t = 35$ (i) $t = 40$ (j) $t = 45$

Figure 7: Local solution to the traffic flow problem of a roundabout crossing with 4 agents (colored in blue, red, purple and orange). Notice how in this run, the red and purple vehicles slow down to let the blue vehicle go first: the convex concave program didn't learn the traffic regulations!

middle of the road, which may not be the shortest path (see figure 1 where the trajectory drives alongside the inner lane of the circuit.)

C. Multi-agent roundabout crossing

Until now, we have studied problems involving a single agent in a specific environment. Let us consider a more complex scenario with M agents interacting with each other. The number one priority being safety, we impose as a constraint a minimum distance between the vehicles, and optimize the positions for $t = 0, \dots, T$ time-steps, such that the total fuel (or distance travelled) is minimized. However, on straight road segments, this formulation is not very interesting since vehicles do not interact much, and would simply stand in line. Therefore, we build a simple roundabout scene, and consider M agents driving to the same target. By denoting $x_t^{(m)}$ the position of the m^{th} vehicle at time-step t, the DCCP program can be formulated similarly to (4), by adding the following constraints:

$$\left\| x_t^{(m)} - x_t^{(m')} \right\|_2 \geq \delta, \quad \begin{array}{l} 1 \leq m, m' \leq M, m \neq m' \\ t = 0, \dots, T \end{array} \tag{5}$$

where δ is the minimum distance that we wish between the vehicles.

4. Conclusion

In this work, we have shown how we can extend the limitations of convex optimization using the DCCP framework, and apply it to different driving scenarios, closed circuits and roundabout crossings, involving one or several vehicles.

References

[1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *arXiv preprint arXiv:1812.03079*, 2018.

[2] Thomas Lipp and Stephen Boyd. Variations and extension of the convex–concave procedure. *Optimization and Engineering*, 17(2):263–287, 2016.

[3] Xinyue Shen, Steven Diamond, Yuantao Gu, and Stephen Boyd. Disciplined convex-concave programming. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 1009–1014. IEEE, 2016.