A Multiobjective MPC Approach for Autonomously Driven Electric Vehicles ⋆

Sebastian Peitz * **Kai Schäfer** * **Sina Ober-Blöbaum** **
Julian Eckstein *** **Ulrich Köhler** *** **Michael Dellnitz** *

* *Department of Mathematics, Paderborn University, Warburger Str. 100, 33098 Paderborn, Germany (e-mail: speitz@math.upb.de)*
** *Department of Engineering Science, University of Oxford, Parks Road, Oxford OXI 3PJ, UK*
*** *Hella KGaA Hueck & Co., Beckumer Str. 130, 59552 Lippstadt, Germany*

Abstract: We present a new algorithm for model predictive control of non-linear systems with respect to multiple, conflicting objectives. The idea is to provide a possibility to change the objective in real-time, e.g. as a reaction to changes in the environment or the system state itself. The algorithm utilises elements from various well-established concepts, namely multiobjective optimal control, economic as well as explicit model predictive control and motion planning with motion primitives. In order to realise real-time applicability, we split the computation into an online and an offline phase and we utilise symmetries in the open-loop optimal control problem to reduce the number of multiobjective optimal control problems that need to be solved in the offline phase. The results are illustrated using the example of an electric vehicle where the longitudinal dynamics are controlled with respect to the concurrent objectives arrival time and energy consumption.

Keywords: Model predictive control; Explicit model predictive control; Multiobjective optimisation; Motion planning; Electric vehicles

1. INTRODUCTION

In many applications from industry and economy, the simultaneous optimisation of several criteria is of great interest. In transportation, for example, we want to reach a destination as fast as possible while minimising the energy consumption. In general, the different objectives are contradictory and the task of computing the set of optimal compromises, the so-called *Pareto set*, arises. There exist various algorithms for the solution of multiobjective optimal control problems (MOCPs) such as scalarisation techniques (cf. Ehrgott (2005) for an overview), evolutionary algorithms (Coello Coello et al. (2007)) or set oriented methods (Schütze et al. (2013)). All approaches have in common that a large number of function evaluations is typically needed, rendering a computation in real-time impossible. However, in particular the design of optimal drive strategies requires online adaption of the controller. This is even more the case now that autonomous driving and battery electric vehicles (EVs) with comparatively low ranges are gaining increased attention, requiring advanced control algorithms.

Control theory has been influenced significantly by the advances in computational power during the last decades. For a large variety of systems, it is nowadays possible to use model based optimal control algorithms to design sophisticated feedback laws. This concept is known as model predictive control (MPC) (see e.g. Maciejowski (2002); Grüne and Pannek (2011)). The general goal of MPC is to stabilise a system by using a combination of open and closed-loop control: using a model of the system dynamics, an open-loop optimal control problem is solved in real-time over a so-called *prediction horizon*. The first part of this solution is then applied to the real system while the optimisation is repeated to find a new control function, with the prediction horizon moving forward (for this reason, MPC is also referred to as moving horizon control or receding horizon control).

Due to the huge success of MPC, a large variety of algorithms has been established, where a first distinction can be made between linear (see e.g. Qin and Badgwell (1997) for an overview in applications and theory) and non-linear MPC (Grüne and Pannek (2011)). The advantage of non-linear MPC is that the typically non-linear system behaviour can be approximated in a more accurate way. Furthermore, special optimality criteria and non-linear constraints can be incorporated easily. However, the complexity and thus the time to solve the resulting optimisation problem increases such that it is often difficult to preserve real-time capability (see e.g. Eckstein et al. (2016)). Further extensions are, for example, *economic MPC* (see e.g. Rawlings and Amrit (2009); Diehl et al. (2011)) or *explicit MPC* (see e.g. Alessio and Bemporad (2009)). In the first approach alternative, *economic* objectives are pursued instead of stabilising the system. In

⋆ This research was funded by the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster *Intelligent Technical Systems OstWestfalenLippe (it's OWL).*

the second approach the problem of real-time applicability is addressed by introducing an offline phase during which the open-loop optimal control problem is solved for a large number of possible situations, using e.g. multi-parametric non-linear programming. The solutions are then stored in a library such that they are directly available in the online phase.

Another way for optimal strategy planning is the concept *motion planning with motion primitives* going back to Frazzoli et al. (2005) (see also Kobilarov (2008); Flaßkamp et al. (2012)). Here, control and state trajectories are obtained by combining several short pieces of simply controlled trajectories that are stored in a motion planning library. These motion primitives can be sequenced to longer trajectories in various combinations. To reduce the computational effort, the motion primitive approach extensively relies on exploiting symmetries, e.g. invariance of the dynamics under translation or rotation.

In this article, we present a new algorithm for multiobjective MPC of non-linear systems. Problems with multiple criteria have been addressed by several authors using scalarisation techniques (see e.g. Bemporad and Muñoz de la Peña (2009) for a weighted sum or Zavala and Flores-Tlacuahuac (2012) for a reference point approach). For non-convex problems, scalarisation approaches often face difficulties such that we want to compute the entire Pareto set in advance. To this end, we combine elements from multiobjective optimal control, explicit MPC and motion planning with motion primitives. The difference to other approaches is the knowledge of the entire Pareto set and the possibility to interactively choose between different objectives such that the system behaviour can be modified online. This can be very useful for autonomous driving, where we are interested in reaching a destination as fast as possible while minimising the energy consumption.

The outline of the article is as follows. In Section 2, we introduce the multiobjective MPC problem and the concept of Pareto optimality before describing the algorithm in detail in Section 3 and comparing it to other MPC approaches. In Section 4, we describe the application of the algorithm to an electric vehicle. The aim is to realise autonomous driving where the passenger can decide between the objectives fast and energy efficient driving. We present the results in Section 5 before drawing a conclusion in Section 6.

2. PROBLEM FORMULATION AND METHODOLOGY

Before describing the algorithm, we will briefly introduce the two main concepts we will be making use of, namely multiobjective optimal control and model predictive control. For more detailed introductions, we refer to Ehrgott (2005) and Grüne and Pannek (2011), respectively.

A *multiobjective optimal control* problem (MOCP) can be formulated mathematically using differential(-algebraic) equations describing the physical behaviour of the system together with optimisation criteria and optimisation constraints in the following way

$$\min_{x,u,t_f} J(x,u,t_f) = \int_{t_0}^{t_f} C(x(t),u(t))\,dt + \Phi(x(t_f)) \quad (1)$$

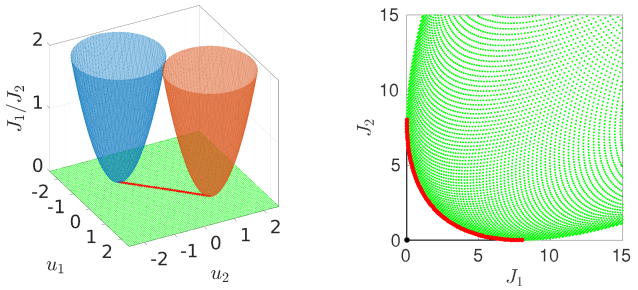
such that

$$\dot{x}(t) = f(x(t),u(t)) \quad \forall t \in [t_0,t_f], \quad x(t_0) = x_0 \quad (2)$$
$$h(x(t),u(t)) \leq 0 \quad \forall t \in [t_0,t_f], \quad (3)$$

where $x(t) \in \mathcal{X}$ is the system state (e.g. the position and velocity of a car) and $u(t) \in \mathcal{U}$ the control (e.g. the engine torque or the steering wheel position). $\mathcal{X}$ and $\mathcal{U}$ are the spaces of feasible states and controls, respectively. The constraints may depend on the state as well as the control, e.g. limiting the velocity or energy consumption. J describes criteria that have to be optimised. When there exists a unique solution $x(t) \in \mathcal{X}$ for every $u(t) \in \mathcal{U}$ and $x_0 \in \mathcal{X}$ and we fix the time frame, we can introduce a reduced objective $J : \mathcal{U} \times \mathcal{X} \to \mathbb{R}^k$, where k is the number of objectives, and the corresponding reduced problem:

$$\min_u J(u,x_0) = \int_{t_0}^{t_f} C(\varphi_u(x_0,t))\,dt + \Phi(\varphi_u(x_0,t_f)). \quad (4)$$

Here $\varphi_u(x_0,t)$ is the flow of the dynamical control system (2).



(a) (b)

Fig. 1. Pareto set (a) and front (b) of the multiobjective optimisation problem $\min_{u\in\mathbb{R}} J(u)$, $J : \mathbb{R}^2 \to \mathbb{R}^2$.

In many applications from industry and economy, we are interested in simultaneously optimising not only one but *several* criteria and hence, $k > 1$ and J is vector-valued. In this situation the solution does in general not consist of isolated optimal points but of the *set of optimal compromises*, the so-called *Pareto set* (cf. Ehrgott (2005) for a detailed introduction). The set consists of all functions $u(t)$ that are *non-dominated*, i.e. for which there does not exist a solution $u^*(t)$ that is superior in all objectives (cf. Figure 1).

For the solution of (4), we here use a scalarisation technique by which the Pareto set is approximated by a finite set of points that are computed consecutively by minimising the euclidean distance between a point $J(u,x_0)$ and a so-called *target point* outside the reachable set in image space, see Dellnitz et al. (2016) for details. Therein, this method is used to compute the Pareto set for the conflicting objectives driven distance and energy consumption for EVs.

The algorithm presented here extends these results in order to construct a feedback controller. This is realised by an *MPC* approach, where the problem (4) is solved repeatedly for varying time frames ($t_0 = t_s$, $t_f = t_{s+p}$, $s = 1,2,\ldots$) online. Then, the first interval of the predicted control, $u(t_s)$, is applied to the real system and the optimal control problem is solved again with a time frame shifted by one. The procedure is illustrated in Figure 2.

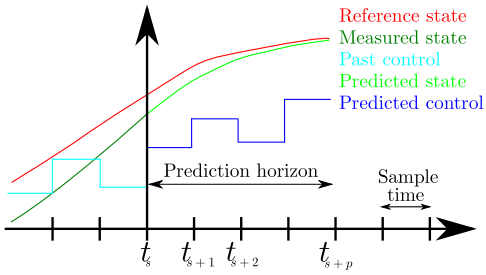


Fig. 2. Sketch of the MPC methodology. While the first part of the predicted control is applied to the system, the next control is predicted (via open-loop optimal control) on a shifted horizon.

MPC was initially developed to stabilise a system (Grüne and Pannek (2011)). However, stabilisation is not always the main concern (e.g. for inherently stable systems). Considering the EV, for example, we focus on the speed to remain within prescribed bounds. In addition to that we pursue further objectives such as minimising the energy consumption. This concept is known as *economic MPC* (see e.g. Rawlings and Amrit (2009); Diehl et al. (2011)).

3. THE OFFLINE-ONLINE MULTIOBJECTIVE MPC CONCEPT

A simple way to consider multiple objectives in MPC is to scalarise the objective function by introducing a weighting factor (i.e. $\widehat{J} = \sum_{i=1}^{k} \rho_i J_i, \rho_i \in [0,1]$). However, in this case an assumption has to be made in advance which can in practice lead to unfavourable results. A slight increase in one objective might allow for a strong reduction in another one, for example. Hence, we are interested in providing the entire Pareto set during the MPC routine. To avoid large computing times during execution, we therefore split the computation in an *offline* and an *online phase*, similar to explicit MPC approaches (cf. Alessio and Bemporad (2009)).

3.1 Offline Phase

The offline phase consists of several steps. First, various *scenarios* are identified for which MOCPs need to be solved. The scenarios are determined by the system states and the constraints. Secondly, in order to reduce the number of scenarios, the dynamical control system is analysed with respect to invariances, which are formally described by a finite-dimensional Lie group G and its group action $\psi : \mathcal{X} \times G \to \mathcal{X}$. A dynamical control system, described by (2), is invariant under the group action ψ, or equivalently, G is a symmetry group for the system (2), if for all $g \in G$, $x_0 \in \mathcal{X}$, $t \in [t_0, t_f]$ and all piecewise-continuous control functions $u : [t_0, t_f] \to \mathcal{U}$ it holds

$$\psi(g, \varphi_u(x_0, t)) = \varphi_u(\psi(g, x_0), t) \quad \forall g \in G. \tag{5}$$

That means that the group action on the state commutes with the flow. Two trajectories are called *equivalent* if they can be exactly superimposed through time translation and the action of the symmetry group. In the classical concept of motion primitives (Frazzoli et al. (2005)), all equivalent trajectories are summed up in an equivalence class, i.e. only a single representative is stored that can be used at many different points when transformed by the symmetry action. In our approach, we extend this concept by identifying symmetries in the solution of the MOCP (4) under variations of the initial conditions x_0:

$$\arg\min_u J(u, x_0) = \arg\min_u J(u, \psi(g, x_0)) \quad \forall g \in G. \tag{6}$$

Thus, we require the Pareto set to be invariant under group actions on the initial conditions. If the objective function is also invariant under the same group action, then all trajectories contained in an equivalence class defined by (5) will also be contained in an equivalence class defined by (6). However, this class may contain more solutions since we do not explicitly pose restrictions on the state but only require the solution of (4) to be identical.

Identifying invariances according to (6), the number of MOCPs can be reduced. If the system is invariant under translation of the initial position $p(t_0)$, for example, we do not need to solve multiple MOCPs that only differ in the position. Once these equivalence classes have been identified, we can reduce the number of possible scenarios accordingly. We then solve the resulting MOCPs on the prediction horizon T_p, introduce a parametrisation ρ (which can then be chosen by the decision maker in the online phase) and store the Pareto sets and fronts in a library such that they can be used in the online phase. Since in general there is an infinite number of feasible initial conditions, there consequently exists an infinite number of scenarios that we have to consider. In practice, this obviously cannot be realised and we have to introduce a finite set of scenarios. In the online phase, we then pick the scenario that is closest to the true initial condition. If a violation of the state constraints has to be avoided (the EV, e.g., is not allowed to go faster than the maximum speed), then a selection towards the "safe" side can be made. In case of the EV, we would consequently pick a solution corresponding to a velocity slightly higher than the actual velocity. This way, the maximally allowed acceleration would be bounded such that exceeding the speed limit is not possible.

3.2 Online Phase

The online phase is now basically a standard MPC approach, the difference being that we obtain the solution of our control problem from a library instead of solving it in real-time, similar to explicit MPC approaches:

1. measure the current system states that are necessary for the identification of the current scenario,
2. choose the corresponding Pareto set from the library, i.e. the one with initial conditions closest to the current system state. (Due to the approximation, we cannot formally guarantee that the constraints are not violated. However, as a start we consider applications where this is acceptable.)
3. choose one optimal compromise u from the set, according to a decision maker's preference ρ,
4. apply the first step (i.e. the sample time) of the solution u to the real system and go back to 1.

The resulting algorithm thus provides a feedback law. In the offline phase, we define the scenarios in such a manner that the system cannot be steered out of the set of feasible states. This means that only controls u are valid that do not lead to a violation of the constraints. Additionally, we

include scenarios which steer the system into the set of feasible states from any initial condition. In the literature, this is known as *viability*, cf. Grüne and Pannek (2011). In case of the EV, for example, we have to include controls such that the velocity can be steered to values satisfying the constraints from any initial velocity.

The presented algorithm can be seen as an extension of (extended) MPC approaches to multiple objectives. We consider *economic* objectives (cf. Rawlings and Amrit (2009)) and do not focus on the stabilisation of the system. This allows us to pursue multiple objectives between which a decision maker can choose dynamically, e.g. in order to react on changes in the environment or the system state itself. In contrast to weighting methods, the entire Pareto set is known, providing increased system knowledge.

4. APPLICATION TO ELECTRIC VEHICLE

In this section the algorithm is utilised to control the longitudinal dynamics of an EV, thereby extending prior work, see Dellnitz et al. (2014) for a scalar optimal control problem, Dellnitz et al. (2016) for a multiobjective optimal control problem and Eckstein et al. (2016) for a comparison of two scalar MPC approaches.

4.1 Vehicle Model

The EV model is derived by coupling the equations for the electrical and the mechanical subsystem via efficiency maps. This yields a system of four coupled, non-linear ordinary differential equations for the system state $x(t) = (v(t), S(t), U_{d,L}(t), U_{d,S}(t))$. Here, v is the vehicle velocity, S is the battery state of charge and $U_{d,L}$ and $U_{d,S}$ are the long and short term voltage drops, respectively. The system is controlled by setting the torque $u(t)$ of the front wheels. Additionally, the battery current $I(t)$ is computed from the state $x(t)$ via an algebraic equation and the position by integrating the velocity: $p(t) = \int_{t_0}^{t} v(\tau)d\tau$. For the derivation and the exact formulation of the dynamical system, we refer the reader to Eckstein et al. (2016).

Based on the system dynamics, we formulate the MOCP for the EV with variable final time:

$$min_u \begin{pmatrix} S(t_0) - S(t_f) \\ t_f - t_0 \end{pmatrix}, \tag{7}$$

$$\dot{x}(t) = f(x(t), u(t)), \qquad t \in [0, t_f] \tag{8}$$

$$v_{min}(t) \leq v(t) \leq v_{max}(t), \qquad t \in [0, t_f] \tag{9}$$

$$I_{min}(t) \leq I(t) \leq I_{max}(t), \qquad t \in [0, t_f] \tag{10}$$

$$x(0) = x_0, \; p(t_f) = p_f. \tag{11}$$

We set the final position p_f to 100 m, which means that we here define the prediction horizon based on the position. Correspondingly, the sample time is also specified with respect to the position, $\delta = 20\,\text{m}$. The conflicting objectives are to reach p_f as fast as possible (J_2) while minimising the energy consumption (J_1). The battery current I is limited in order to avoid damaging the battery which results in implicit constraints on the control u. The velocity constraints are part of the scenarios which are defined in the offline phase.

4.2 Offline Phase: System Analysis and Solution of Multi-objective Optimal Control Problems

In this section we describe how the different steps of the offline phase are applied to the EV.

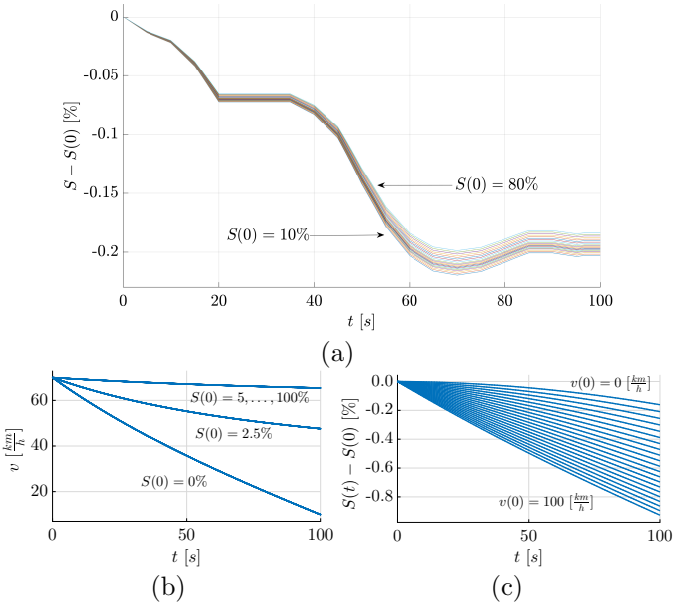
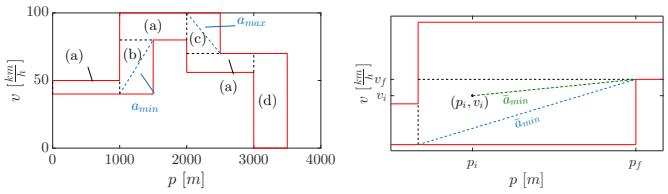


Fig. 3. (a) Almost invariance of $S(t)$ under variations of the initial value $S(0)$. (b) Invariance of the velocity $v(t)$ under variations of the initial value $S(0)$ for $S(0) \geq 5\%$. (c) No invariance of the state of charge $S(t)$ under variations of the initial velocity $v(0)$.

Symmetry Analysis The more invariances the MOCP possesses (in the sense of Equation (6)), the fewer problems need to be solved which significantly reduces the computational effort. Hence, we numerically analyse the system in this regard. Since the position p does not occur in the dynamical system (8), the dynamics are obviously invariant under translations in p. Moreover, when exemplary looking at the velocity v and the state of charge S (cf. Figure 3 (a) and (b)), we see that, on the one hand, the trajectories are almost invariant for a wide range of translated initial values of the state of charge $S(0)$. Note that this is not a strict invariance. However, as argued in Section 3, we do not require invariances according to Equation (5) but according to the weaker condition (6) which is satisfied much more accurately for the EV application. When looking at Figure 3 (c) on the other hand, we observe that the dynamics are clearly not invariant under translations in the initial velocity $v(0)$. After performing the same analysis with regards to the other state variables $U_{d,L}$ and $U_{d,S}$, we can conclude that we only need to define scenarios with respect to the initial velocity $v(0)$ and the active constraints $v_{min}(t)$ and $v_{max}(t)$.

Constraints A constraint on the velocity is given by the current speed limit $v_{max}(p)$ which depends on the current vehicle position. To avoid interference with other vehicles by driving too slow, we define a minimal velocity $v_{min}(p) = 0.8 \cdot v_{max}(p)$. (Here the velocities are functions of the position. In the MOCP, they have to be reformulated as functions of time.) Our *set of feasible states* is now determined by the velocity constraints, i.e. $v_{min}(t) \leq$



(a) (b)

Fig. 4. (a) Possible scenarios of boundary conditions. a: constant velocity. b: acceleration. c: deceleration. d: stop sign. (b) Computation of lower bound $\bar{a}_{min}$ for the velocity gradient dv/dp.

$v(t) \leq v_{max}(t)$, which determine the different scenarios. We distinguish between four cases (see Figure 4 (a)). While the cases constant velocity (box constraints) and stopping ($v = 0$ at the stop sign) are easily implemented, we introduce a linear constraint for the scenarios (b) and (c), respectively (see Figure 4 (b)) where, depending on the current velocity, a minimal increase $\bar{a}_{min} = (dv/dp)_{min}$ or decrease, respectively, must not be violated. An example is shown in Figure 5, where the Pareto set and the resulting velocity profiles are shown for the scenario $v(0) = 60\frac{km}{h}$ and $\bar{a}_{min} = 0.05\ \frac{km/h}{m}$. Note that here, we have chosen the control u to be constant over the prediction horizon in order to reduce the numerical effort. As mentioned in Section 3, we cannot solve an MOCP for every initial condition. Solving an MOCP for every step of 0.1 in the initial velocity leads to 1727 MOCPs in total. The resulting storage requirements are moderate and not restrictive for in-vehicle application.

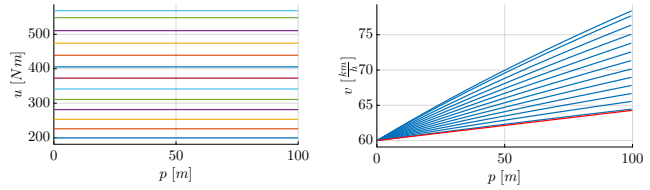


Fig. 5. (a) Pareto set for an accelerating scenario with $v(0) = 60\frac{km}{h}$ and $\bar{a}_{min} = 0.05\ \frac{km/h}{m}$. (b) The corresponding trajectories of $v(t)$.

4.3 Online Phase: Multiobjective MPC with Paretooptimal Control Primitives

The online phase is now exactly as described in Section 3. In each sample time, the current velocity and the active constraints (for the current position) are evaluated in order to determine the valid scenario. The corresponding Pareto set is then selected from the library and according to the weighting parameter $\rho \in [0, 1]$ determined by the decision maker, an optimal compromise is chosen which is then applied to the system. On a standard computer, this operation takes in the order of 10 milliseconds in Matlab.

5. RESULTS AND DISCUSSION

In Figure 6, several solutions with different weights ρ are shown for an example track including two stop signs. The *set of feasible states* is bounded by the red lines v_{min} and

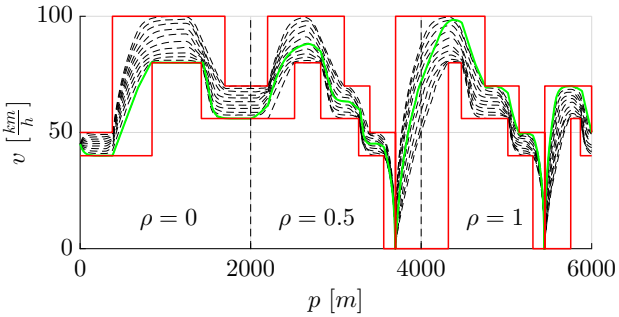


Fig. 6. Different trajectories computed by the MPC approach. The dashed lines use a constant weight ρ whereas the green line possesses dynamic weighting ($\rho = 0/0.5/1.0$, respectively)

v_{max}. The dashed lines correspond to constant weights, varying from $\rho = 0$ (energy efficiency) to $\rho = 1$ (high velocity) and the solid green line is a solution where the weighting is changed from 0 over 0.5 to 1 during driving. We clearly see that the vehicle is driving according to the decision maker's preference. This means that we have realised a closed-loop control for which the objectives can be adjusted dynamically. This can either be done manually or by an additional algorithm, which for example takes into account the track, the battery state of charge and the current traffic. The objective function values for the entire track and different values of ρ are depicted in Figure 7 (a).

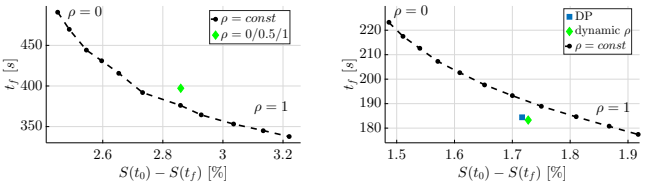


Fig. 7. Function values for the scenarios depicted in Figure 6 and in Figure 8 for different weights ρ and in comparison to the Dynamic Programming solution.

In order to evaluate the quality of our solution, we compare it to a control computed via dynamic programming (DP, see Bellmann and Dreyfus (2015) for an introduction and Sundström and Guzzella (2009) for the algorithm that is used): For computational reasons, the comparison is performed on a shorter track without stop signs and a relatively coarse discretisation leading to a 100-dimensional problem. In the DP problem, we use a simplified linear model (cf. Eckstein et al. (2016)) and the objective is a weighted sum of the MOCP (7), $J = t_f + \beta E(t_f)$, where E is the consumed energy computed by integrating over the wheel torque and $\beta = 6 \cdot 10^{-5}$. In Figure 7 (b), we see that the solution obtained via DP is superior to our MPC approach. This is not surprising since in MPC, we only consider a finite horizon such that the results are at best suboptimal (Grüne and Pannek (2011)), whereas the entire track is considered at once in DP. Consequently, the DP algorithm is not real-time applicable and does not possess feedback behaviour. Additionally, we have until now only considered constant torques over the prediction horizon

in our approach. We intend to refine the discretisation in future work and expect an improved performance.

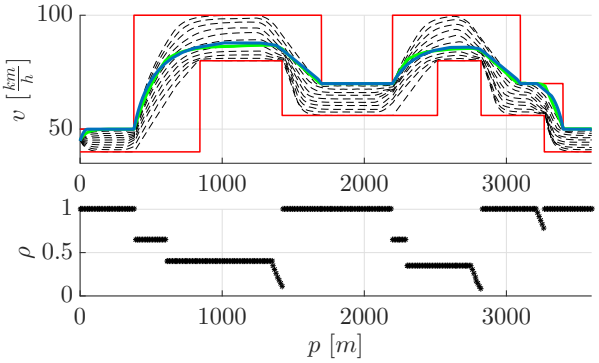


Fig. 8. Validation of the approach versus a Dynamic Programming solution (blue). Green line: dynamic weighting according to the lower plot.

When using a simple, manually tuned heuristic for the preference ρ based on the velocity instead of fixed values (larger values for ρ at low velocities, lower values at high velocities and linear changes in ρ when approaching braking manoeuvres, see Figure 8, bottom), we see that we can improve the quality of our solution significantly which is now comparable to the global optimum obtained by DP. We see in Figures 8 (top) and 7 (b), respectively, that the resulting trajectories as well as the function values J_1 and J_2 almost coincide. By this, we obtain two different ways to utilise the results. On the one hand, a decision maker can select the preference according to his wishes and on the other hand, ρ can be determined by a heuristic, leading to solutions of a quality comparable to the global optimum.

6. CONCLUSION

We present an algorithm for MPC of non-linear dynamical systems with respect to multiple criteria. The algorithm utilises elements from economic and explicit MPC, multi-objective optimal control and motion planning. According to a decision maker's preference, the system is controlled in real-time with respect to an optimal compromise between conflicting objectives. Using a simple heuristic for the weighting factor ρ, we obtain solutions of equivalent quality compared to a global optimum computed by open loop DP. In the future, we intend to analyse the proposed method from a more theoretical point of view, addressing questions concerning feasibility and stability for systems where these aspects are critical. Furthermore, we want to improve our control strategies by developing intelligent heuristics for the preference weighting function ρ.

REFERENCES

Alessio, A. and Bemporad, A. (2009). A Survey on Explicit Model Predictive Control. In L. Magni, D.M. Raimondo, and F. Allgöwer (eds.), *Nonlinear Model Predictive Control: Towards New Challenging Applications*, 345–369. Springer Berlin Heidelberg.

Bellmann, R.E. and Dreyfus, S.E. (2015). *Applied dynamic programming*. Princeton University Press.

Bemporad, A. and Muñoz de la Peña, D. (2009). Multi-objective model predictive control. *Automatica*, 45(12), 2823–2830.

Coello Coello, C.A., Lamont, G.B., and van Veldhuizen, D.A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*, volume 2. Springer New York.

Dellnitz, M., Eckstein, J., Flaßkamp, K., Friedel, P., Horenkamp, C., Köhler, U., Ober-Blöbaum, S., Peitz, S., and Tiemeyer, S. (2014). Development of an Intelligent Cruise Control Using Optimal Control Methods. In *Procedia Technology*, volume 15, 285–294. Elsevier.

Dellnitz, M., Eckstein, J., Flaßkamp, K., Friedel, P., Horenkamp, C., Köhler, U., Ober-Blöbaum, S., Peitz, S., and Tiemeyer, S. (2016). Multiobjective Optimal Control Methods for the Development of an Intelligent Cruise Control. In G. Russo et al. (ed.), *Progress in Industrial Mathematics at ECMI 2014 (to appear)*.

Diehl, M., Amrit, R., and Rawlings, J.B. (2011). A lyapunov function for economic optimizing model predictive control. *IEEE Transactions on Automatic Control*, 56(3), 703–707.

Eckstein, J., Schäfer, K., Peitz, S., Friedel, P., Ober-Blöbaum, S., and Dellnitz, M. (2016). A Comparison of two Predictive Approaches to Control the Longitudinal Dynamics of Electric Vehicles. *Procedia Technology*, 26, 465–472.

Ehrgott, M. (2005). *Multicriteria optimization*. Springer Berlin Heidelberg New York.

Flaßkamp, K., Ober-Blöbaum, S., and Kobilarov, M. (2012). Solving Optimal Control Problems by Exploiting Inherent Dynamical Systems Structures. *Journal of Nonlinear Science*, 22(4), 599–629.

Frazzoli, E., Dahleh, M.A., and Feron, E. (2005). Maneuver-Based Motion Planning for Nonlinear Systems with Symmetries. *IEEE Transactions on Robotics*, 21(6), 1077–1091.

Grüne, L. and Pannek, J. (2011). *Nonlinear model predictive control*. Springer.

Kobilarov, M. (2008). *Discrete geometric motion control of autonomous vehicles*. Ph.D. thesis, University of Southern California.

Maciejowski, J.M. (2002). *Predictive Control: With Constraints*. Prentice Hall, Harlow, England.

Qin, S.J. and Badgwell, T.A. (1997). An overview of industrial model predictive control technology. In *AIChE Symposium Series*, volume 93, 232–256. American Institute of Chemical Engineers.

Rawlings, J.B. and Amrit, R. (2009). Optimizing process economic performance using model predictive control. In *Nonlinear model predictive control*, 119–138. Springer.

Schütze, O., Witting, K., Ober-Blöbaum, S., and Dellnitz, M. (2013). Set Oriented Methods for the Numerical Treatment of Multiobjective Optimization Problems. In E. Tantar et al. (ed.), *EVOLVE - A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation*, volume 447 of *Studies in Computational Intelligence*, 187–219. Springer Berlin Heidelberg.

Sundström, O. and Guzzella, L. (2009). A generic dynamic programming Matlab function. In *2009 IEEE Control Applications, (CCA) & Intelligent Control,(ISIC)*, 1625–1630.

Zavala, V.M. and Flores-Tlacuahuac, A. (2012). Stability of multiobjective predictive control: A utopia-tracking approach. *Automatica*, 48(10), 2627–2632.