Unified Predictive Fuel Efficiency Optimization Using Traffic Light Sequence Information*

Tianyi Guan[1] and Christian Walter Frey[2]

***Abstract*— Energy efficiency has become a major issue in trade, transportation and environment protection. While the next generation of zero emission propulsion systems still have difficulties in reaching similar travel distances as power-trains with combustion engines, it is already possible to increase fuel efficiency in regular vehicles by applying a more fuel efficient driving behavior. The rise of V2X technologies have opened up new possibilities for safety and energy efficiency applications. This publication proposes a model predictive approach that makes use of a power-train model and a sequence of traffic lights over a finite optimization horizon. The optimization problem is solved in a unified manner, i.e. power-train properties and traffic light phases are not considered separately but evaluated in a single cost function. A stagewise forward-backward Dynamic Programming approach involving cost re-utilization is used for optimization. In order to further decrease the search space, certain continuous entities are not explicitly regarded as a state component, but rather calculated during optimization.**

***Key words*:** Dynamic Programming, reuse historic costs, search space reduction, fuel efficiency driving, model predictive optimization

I. INTRODUCTION

Several authors have published solutions in the area of fuel efficient driving. The approaches range from complete vehicle control to passive driver assistance systems. Autonomous approaches include the works of [2] [3] [4] [5] [6] [7]. Topography height maps and perception sensors are used to enable model predictive approaches over a receding horizon. In the case of [3] [4], gradient descent based approaches like sequential quadratic programming are used. Although these approaches can have a high computational efficiency, there are major problems which include possibly uncontrollable and excessive suboptimality, convergence difficulties due to numerous constraints and difficulty to maintain optimality in mixed integer problems. Another major class of optimization are planning algorithms [8] like Dynamic Programming [9]. Examples include [2] [7] [10] [11] [12]. Dynamic Programming is less affected by suboptimality as long as state and transition discretization are sufficiently subtle, but can lead to high computational complexity. In [11] the author use common stagewise Dynamic Programming with very subtle discretization. The claim is that the optimization can be executed in real-time on a modern PC. But to the best of our understanding, the author simply compares the aggregated execution time of the algorithm with the vehicle's travel duration in the simulation. But what is really needed is the average execution time or worst case execution time for one optimization update. Therefore, to the best of our understanding, it is uncertain if the system proposed in [11] is indeed real-time capable. In a subsequent publication [2], a search space reduction is proposed which makes use of the observation that optimal paths never intersect. But to the best of our understanding, this approach can make the optimization result implementation dependent, because with different ordering and selection of transitions there could be different outcomes. Alternatively, methods named Approximate Dynamic Programming [12] reduce the number of states in exchange for precision. The challenge is to use intelligent interpolation strategies and choose significant states to efficiently decrease the search space without losing too much optimality. Another technology that extends the possibilities of energy efficiency optimization is vehicle to infrastructure and vehicle to vehicle communication (V2X). Approaches that consider traffic lights include [13] [14] [15] [16]. By including traffic light phase sequences, more possibilities for optimization arise but there are also several new problems. If planning algorithms like Dynamic Programming are used, the search space must contain possible velocities, position dependent and time dependent properties for the entire computation horizon to attain a globally optimal solution. Thus, the search space and the computation time is drastically increased. If a gradient descent based approach is used, the constraints become more stringent due to red phases of the traffic lights. This makes the optimization algorithm harder to converge. In [7] the authors compute an optimal velocity profile for an intersection using traffic light phases of a single traffic light at an intersection and a fuel consumption model in a unified manner. In [13] the authors consider a series of traffic lights. They separate the predictive traffic light phase optimization from model predictive control. In order to save computation time and avoid non-convex optimization problems, they apply a greedy search on the available traffic light sequence. The result is a velocity sequence that can guide a vehicle through the green phases in a series of traffic lights using a subsequent model predictive optimization.

In this publication, the authors try to solve the optimization problem in a unified manner, i.e. power-train properties and traffic light phases are not considered separately but evaluated in a single cost function. Furthermore, a series of

*This work is part of the project REM 2030 [1] which is supported by the Fraunhofer society and the federal ministries of the state of Baden-Württemberg in Germany.

[1]Tianyi Guan is with the Department of Measurement, Control and Diagnosis, Fraunhofer IOSB, 76131 Karlsruhe, Germany tianyi.guan@iosb.fraunhofer.de

[2]Christian W. Frey is with the Department of Measurement, Control and Diagnosis, Fraunhofer IOSB, 76131 Karlsruhe, Germany christian.frey@iosb.fraunhofer.de

traffic lights is regarded. The goal is to reduce overall trip time and overall fuel consumption. In general, a globally optimal solution would be ideal. But if the vehicle state is defined e.g. by the vehicle's position, velocity, gear level and the absolute time, searching for the optimal solution will be highly computationally expensive as absolute time is a continuous variable. Nevertheless, absolute time cannot be ignored as it is needed to identify traffic light phases. The proposed approach in this publication calculates the absolute travel duration along different path candidates, instead of expanding the search by the absolute continuous time as an explicit state component. Compared to a fully expanded search grid with absolute time as an explicit state component, we traverse through a much smaller search space, i.e. many state transitions are not regarded. Theoretically, this strategy can lead to a suboptimal solution compared to a fully expanded search grid, but greatly decreases the search space. Furthermore, the optimization reuses previously calculated costs to reduce computation time, which was presented in one of our previous works [17] and will be briefly explained. The optimization is used in the context of model predictive optimization, velocity profile optimization and fuel efficient driving. The final goal is to compute a fuel efficient driving profile consisting of velocity and gear. It can then be used to guide a controller that computes all control variables necessary to control e.g. an autonomous vehicle or serve as a passive driving assistance system. Some preliminary simulation results will be shown to give a first assessment of the fuel saving potentials of our system, while more detailed evaluations have to be reserved for future publications as certain evaluation tools are still under development.

A. Fuel consumption model

The formulation of the fuel consumption model in this work is inspired by [11] [18] [19] and defined in an inverse manner. This means the model output is computed from the vehicle's acceleration. Propagation through the model ultimately leads to a fuel consumption rate in the engine. The model incorporates the external resistance forces: acceleration resistance F_a, air friction resistance F_{air}, slope resistance F_{slope} and rolling resistance F_{roll}. With the wheel radius r_w, the resistance forces can be used to derive the resistance torque T_w applied to the wheel. The transmission model assumes that transient behaviour can be neglected. The wheel speed ω_w is directly translated to the engine speed ω_e via the transmission ratio $i_t(G)$. The ratio is dependent on the selected gear level G. By neglecting the engine inertia, the engine torque T_e can be directly computed from T_w and the engine drag torque $T_d(\omega_e)$. The engine drag torque $T_d(\omega_e)$ is approximated by a linear function similar to [11]. The maximum brake torque at the wheels $T_{b,max}$ is applied when the driver fully depresses the brake pedal. The fuel consumption can be computed from a brake specific fuel consumption map $f_{bsfc}(\omega_e, T_e)$, which is dependent on engine speed and engine torque. The ratio is basically the inverse of the engine's efficiency. It describes how much fuel is needed to generate power at different engine operation points. In this publication the brake specific fuel consumption characteristic map $\frac{dV_f}{P_e dt}$ of a Ford Zetec 2.0L engine is used.

$$v = \omega_w r_w \quad (1)$$

$$\omega_e = i_t(G)\omega_w \quad (2)$$

$$T_w = r_w(F_{air} + F_{roll} + F_{slope} + F_a) + T_b(b) \quad (3)$$

$$T_e = \frac{T_w}{i_t(G)} + T_d(\omega_e) \quad (4)$$

II. Model predictive optimization

The goal of the optimization is to find a velocity and gear change profile that maximizes fuel efficiency over a currently examined receding computation horizon S. The profile is a chain of decisions or more specifically target velocities and target gear levels as state components in the form of states $\{v, G\}$ or $\{v, G, t\}$ if the absolute time t is also considered as a state component. But for the sake of comprehension, we will leave out t for now. More details will be given in section II-B. The first optimal velocity and gear level combination within the optimal profile p^* is the next decision as well as the preferred next state. In order to find the optimal profile, a multi-stage graph is constructed over the computation horizon (Fig. 1). The stages s_i within S are erected with a constant frequency, e.g. every $100m$. Each stage has a specific position within the computation horizon as well as minimum and maximum velocity limits $v_{min}(s_i)$, $v_{max}(s_i)$ and other position related properties. For sake of simplicity, stages and stage positions share the same denotation s_i in the following sections. Furthermore, each stage s_i has a set $S_i = \{n_i^0, n_i^1, ...n_i^a, ..., n_i^M\}$ of graph nodes or rather state nodes n_i^a containing possible combinations of velocity and gear levels $n_i^a = \{v_a, G_a\}$. For sake of simplicity, we will from now on only address n_i^a as node which is equivalent to a state. The graph used for our work has an open end, i.e. the last stage has as many nodes as the intermediate stages. The single *origin node* n_0 at the first stage s_0 refers to the current position and current state $\{v_0, G_0\}$ of the vehicle when the optimization is started the very first time. It is also the very first *start node*. The range of possible velocities for a stage from s_1 to s_N is constrained by the corresponding minimum and maximum velocity limits as well as acceleration and braking constraints of the vehicle. The computation horizon moves with the vehicle. Whenever the vehicle has passed a stage, it triggers a new optimization. In the ideal case the next start node will be the first state $\{v^*, G^*\}$ within the optimal profile. During the *forward pass*, stagewise forward-backward Dynamic Programming solves the cost minimization problem by evaluating all transition costs from one stage to the next in a recursive manner. Beginning from the second stage onward, nodes start working with minimum accumulated costs of the nodes from the previous stage. The accumulated minimum cost $J^*(n_i^b)$ of a node n_i^b at stage i is the minimum sum of all possible predecessor nodes n_{i-1}^a to n_i^b and the accumulated minimum cost $J^*(n_{i-1}^a)$ of the respective node n_{i-1}^a. After optimization, every node with the exception of the original node has one unique optimal

predecessor node unless constraint violations forbid it. But a node may have several or no successors.

$$J^*(n_i^b) = \min_{n_{i-1}^a \in S_{i-1}} \Big(J(n_{i-1}^a, n_i^b) + J^*(n_{i-1}^a) \Big) \quad (5)$$

The optimal predecessor node $n_{i-1}^*(n_i^b)$ of n_i^b is identified the same way.

$$n_{i-1}^*(n_i^b) = arg \min_{n_{i-1}^a \in S_{i-1}} \Big(J(n_{i-1}^a, n_i^b) + J^*(n_{i-1}^a) \Big) \quad (6)$$

Finally, let n_N^* be the optimal end node with the smallest accumulated costs of all end nodes.

$$n_N^* = arg \min_{n_N^a \in S_N} J^*(n_N^a) \quad (7)$$

During the *backward pass*, the optimal profile $p^* = \{n_i^*\}$ with $i \in [0, N]$ is constructed by following the identified chain of optimal nodes from the last stage to the start node. It has one optimal decision $n_i^* = \{v_i^*, G_i^*\}$ for every stage s_i within the graph.

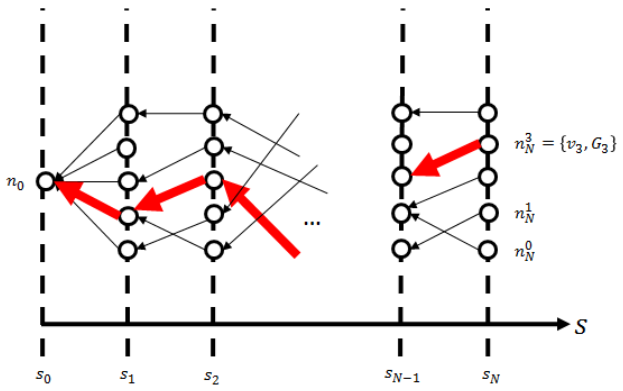

Fig. 1. Graph structure used for forward-backward Dynamic Programming (best viewed in color). The end stage is open, i.e. it has the same number of state nodes as intermediate stages. Red arrows denote the optimal path.

For further details and background information regarding Dynamic Programming, the reader may refer to [9].

A. Cost function

In this work, fuel efficiency is defined as a trade-off between low fuel consumption and short travel time. Thus, the two most important cost terms in the cost function are fuel consumption and travel time duration. Nevertheless, travel time duration is actually not directly taken into account. The reason is that judging from the engine's efficiency, i.e. fuel consumption compared to the power generated, the highest engine efficiency is often at mid-range engine speed and relatively high engine torque. But a constant cruise speed often only requires low engine torque. Therefore, the optimization result may frequently change between acceleration and coasting to attain high engine efficiency in certain scenarios. Velocity oscillation may be a theoretically optimal behavior in certain cases, but inconvenient in practice. Hence, instead of penalizing travel time, an optimal stationary state, i.e. constant cruise velocity and gear, is computed for each decision within the optimal decision chain. Velocity deviations from the stationary optimal cruise velocity is then punished. The cost terms are all weighted and normalized and can only be positive or equal zero. If a cost term exceeds 1, one can infer that the associated transition is unfavorable.

1) Optimal stationary state: In order to compute the optimal stationary state, it is demanded that both cruise speed and gear choice remain constant. The fuel consumption $dV_f(n_i^a, n_j^b)$ from a node n_i^a at stage s_i to a node n_j^b at the next stage s_j can be computed from velocity and gear transition and the fuel consumption model under the given stationary constraint. The weighting w_f is chosen as the inverse of the maximum fuel consumption rate $\frac{dV_{f,max}}{dt}$.

$$J_f(n_i^a, n_j^b) = w_f dV_f(n_i^a, n_j^b) = \frac{dV_f(n_i^a, n_j^b)}{dV_{f,max}} \quad (8)$$

The travel time duration is penalized using the distance ds between two stages and the average velocity of the node transition $\bar{v}(n_i^a, n_j^b)$. The weighting w_t is chosen as the inverse of maximum travel duration dt_{max} resulting from the minimum tolerable speed.

$$J_t(n_i^a, n_j^b) = w_t \frac{ds}{\bar{v}(n_i^a, n_j^b)} = \frac{ds}{\bar{v}(n_i^a, n_j^b) dt_{max}} \quad (9)$$

The optimal stationary cruise velocity is retrieved through enumeration.

$$\begin{aligned} \bar{v}_s^*(s_i, s_j) = & arg \min_{n_i^a \in S_i, n_j^n \in S_j} \Big(J_f(n_i^a, n_j^b) + J_t(n_i^a, n_j^b) \Big), \\ & v(n_i^a) = v(n_j^b) \wedge G(n_i^a) = G(n_j^b) \end{aligned} \quad (10)$$

2) Cost terms: Deviation from the optimal stationary speed $\bar{v}_s^*(s_i, s_j)$ is penalized. The weighting w_{dv} is a design parameter.

$$J_{dv} = w_{dv} |\bar{v}(n_i^a, n_j^b) - \bar{v}_s^*(s_i, s_j)| \quad (11)$$

The use of brakes is not directly penalized by the brake torque at the wheels. The optimization uses the model to investigate if active braking is needed. The necessity for braking is denoted by b, which leads to a strong penalty (e.g. 1) if the brake is used.

$$J_b(n_i^a, n_j^b) = b(n_i^a, n_j^b) = \begin{cases} 0, & T_b(n_i^a, n_j^b) = 0 \\ 1, & T_b(n_i^a, n_j^b) > 0 \end{cases} \quad (12)$$

Heavy acceleration and deceleration can be uncomfortable for passengers. Therefore, the absolute change in velocity from one stage to the next is penalized. The weighting w_a is a design parameter.

$$J_a(n_i^a, n_j^b) = w_a \left| dv(n_i^a, n_j^b) \right| \quad (13)$$

In order to prevent oscillations in the gear change solution and reduce attrition, gear changes are penalized. The weighting w_G is a design parameter.

$$J_G(n_i^a, n_j^b) = w_G \Delta G(n_i^a, n_j^b) \quad (14)$$

B. Absolute time computation and search space reduction

For sake of simplicity, only the red and green phases of traffic lights are distinguished. The yellow phase is incorporated into the closest green phase. In this publication we confine our investigation to traffic lights with fixed phases. The most important variables when dealing with traffic lights with fixed phases are the absolute time and absolute position of the vehicle. When the vehicle drives past a traffic light during a green phase, the traffic light has no influence on fuel efficiency. But when it changes to red, it becomes a barrier. Thus, absolute time is actually a dynamic constraint during red phases and does not influence fuel efficiency otherwise. The time transitions or increments from one node to the next can be computed independently from absolute time. Different to velocity and gear, time transition dt is unambiguously determined by velocity v and the fixed distance ds between two stages through $v = \frac{ds}{dt}$. As the optimization evaluates different state transitions to find the optimal solution, variation with respect to absolute time will undoubtedly often lead to numerous invalid time transitions $dt(n_i, n_j)$ between two nodes n_i and n_j, because the corresponding velocity transitions and fixed stage positions often lead to contradictions in the sense of $v(n_i, n_j) \neq \frac{s_j - s_i}{dt(n_i, n_j)}$. Thus, variation over absolute time leads to a great amount of unusable transitions. Nevertheless for a globally optimal solution, absolute time actually needs to be incorporated as an additional state component to ascertain that all possible path are evaluated. But considering that time is a continuous entity, subtle discretization can make the optimization too slow for real time applications. Combined with the previously stated observations, there is a strong urge for simplification. In this work, we propose to simply aggregate the absolute time during the forward pass of Dynamic Programming instead of expanding the graph by an additional continuous state component. Whenever a node $n_i^a = \{v_i^a, G_i^a\}$ has been completely evaluated, i.e. all transitions to the previous stage have been evaluated, n_i^a will memorize the absolute travel duration $t(n_i^a)$ from the start node via its optimal predecessor $n_{i-1}^*(n_i^a)$. An example is given in Fig. 2 for node n_2^4, whose optimal predecessor is n_1^4.

$$t(n_i^a) = dt(n_{i-1}^*(n_i^a), n_i^a) + t(n_{i-1}^*(n_i^a)) \tag{15}$$

We assume that the passengers do not wish to stop at a red light. To penalize red phases, a high cost is chosen. Nevertheless, the vehicle is allowed to stop at a red light if no other possibility is viable. As explained in section II, due to normalization a cost value of 1 denotes high penalty. Therefore the penalty for encountering a red phase must be at least 1. In order to express the severity of red phases, costs significantly above 1 are recommended, e.g. 10. Let $T_k = [t_c, t_d]$ be a single red phase interval of some traffic light with t_c and t_d denoting the start and end time of the red phase. Let $T(s_i) = \{T_1, T_2, ..., T_L\}$ be the set of all relevant red phase intervals of a traffic light at a stage s_i. If there is no traffic light at s_i then $T(s_i) = \{\}$. Let $t(n_i^a)$ be the calculated absolute time at a node n_i^a. This leads to the formulation of an additional cost term $J_{tl}(n_i^a, n_j^b)$ that only applies to traffic lights' red phases.

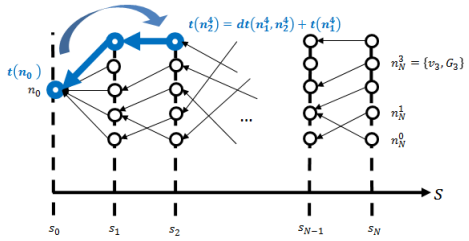

Fig. 2. Aggregation of absolute time (best viewed in color). Absolute time is not regarded as a state component. Instead, the absolute time is calculated by aggregating transition time duration from one stage to the next. The time duration is retrieved from stage positions and the velocity transition to the optimal predecessor node. In the example the origin node n_0 has the absolute time $t(n_0)$. The accumulated absolute time of node n_2^4 is composed of the transition time duration $dt(n_1^4, n_2^4)$ and the accumulated absolute time $t(n_1^4)$ of the optimal predecessor n_1^4. The involved nodes and transitions are marked in blue.

$$J_{tl}(n_i^a, n_j^b) = \begin{cases} 0, & t(n_i^a) \notin T(s_i) \wedge t(n_j^b) \notin T(s_j) \\ 10, & t(n_i^a) \in T(s_i) \vee t(n_j^b) \in T(s_j) \end{cases} \tag{16}$$

The sum of all cost terms constitute the complete cost function.

$$\begin{aligned} J_s(n_i^a, n_j^b) = & J_f(n_i^a, n_j^b) + J_{dv}(n_i^a, n_j^b) \\ & + J_b(n_i^a, n_j^b) + J_a(n_i^a, n_j^b) \\ & + J_G(n_i^a, n_j^b) + J_{tl}(n_i^a, n_j^b) \end{aligned} \tag{17}$$

The calculation of the absolute travel time along path candidates during optimization, greatly decreases the number of evaluated state transitions compared to a graph with fully expanded states. Theoretically, this can lead to suboptimal solutions compared to search grids with higher dimensions. In our case each completely evaluated node only saves the accumulated time with respect to its optimal predecessor instead of all predecessors. Although there are many node candidates at each stage to choose from, many valid (albeit seemingly unfavourable) node transitions are discarded and cannot be retrieved unless the optimization is repeated.

C. Reusing previous costs

In order to compute the optimal solution, an adapted forward-backward Dynamic Programming method (from one of our previous publications) is used, which incorporates memorized accumulated minimum costs of previous optimization steps [17]. Whenever the horizon moves forward, it adds a new stage to its end and removes its first stage at the beginning. From now on, only the transitions to the new end stage are evaluated. All other minimum costs on intermediate stages are retrieved from memory. This method does not increase memory usage compared to ordinary Dynamic Programming. Another difference is that the optimization now refers to the origin state, i.e. the start state when the system computes the very first optimization. As the computation horizon moves on, we actually need an optimal path that refers to the new start state of the vehicle. But as long as the new optimal path traverses through the new start state, the

new optimal path is also optimal with respect to the new start start due to the principle of optimality [9]. If the optimization problem excessively changes during the next optimization, the optimal path may no longer traverse through the new start node. In this case a possibly suboptimal correction of the optimal path is necessary. For further details, the reader may refer to [17].

III. Results

In this section, preliminary results are shown regarding fuel and time saving potentials. The preliminary evaluation is completely conducted in a simulated environment. The simulations are implemented in C++ and conducted on a modern PC (Intel i7 870 at 2.93GHz, 8GB RAM). The simulation scenario uses the same vehicle model that has been described in section I-A. The road has no inclination and no other traffic participants are regarded. Numerous scenarios with random simulation parameter combinations are investigated.

A. Example scenario

The following random scenario is presented as an example to give a graphic understanding of the process: The route has three traffic lights, their time sequences and positions are randomly chosen integer numbers. The speed limit is randomly set to $v_{max} = 150$kph. The velocity discretization is set to 5kph and gear discretization is set to 1. The task is to compute optimized profiles for the next 2000m of road. The length of the horizon is 1000m with a discretization of 100m. It is assumed that the vehicle is able to precisely execute the computed velocity and gear-shift profiles. A simplified simulated driver is defined. It is assumed that the driver's favourable cruise speed is the current speed limit and prefers a high gear level as long as the engine speed is between 1000rpm and 2000rpm. If he sees a red traffic light within 100m in front of him, he will start linearly decelerating to a full stop at the traffic light unless the light switches back to green. The start velocity and gear level are randomly set to $v_{start} = 50$kph and $G_{start} = 3$. The optimization yields a stationary optimal target velocity of 125kph at the highest gear level $G_{max} = 5$. As can be seen in Fig. 3, the optimization (blue line) does not immediately shift to the highest gear during the initial acceleration from 50kph to 100kph. The reason behind this decision is that combustion engines usually have their peak efficiency at relatively high loads and mid-range engine speed. Although the fuel consumption rate is higher, the acceleration duration is shorter and more fuel energy can be actually converted into kinetic energy. In contrast, the driver registers that the engine speed is significantly above 1000rpm and shifts to the highest gear right at the beginning. Additionally, he accelerates to the speed limit which is an unfavourable cruise velocity for the given vehicle. When facing traffic lights, the optimization makes the vehicle coast to a lower speed. The velocity profile is comparatively smooth while the driver decelerates and accelerates more than necessary because he does not know the traffic light phases. In this specific example, the optimization profile consumed approximately 30% less fuel compared to the simulated driver and had an approximately 1.4% shorter travel duration. For this scenario, regular Dynamic Programming updates approximately need 1.3s in average, while Dynamic Programming with cost re-utilization reduces the computation time to approximately 0.14s. No corrections were necessary during cost re-utilization.

B. Other scenarios

The reduction in fuel consumption and travel duration heavily depends on the specific scenario and the definition of the driver. The example scenario refers to cases, in which the speed limit is very high or no speed limit is defined. These types of scenarios yield the highest savings of more than 30%. The optimization knows the optimal cruise velocity while the driver may choose a much higher velocity that leads to high consumption through heavy air drag. Savings in travel time duration can be relatively low (single digit percentage) if the optimization strictly abides the speed limits, while the driver prefers to cruise at the highest possible speed. Certain traffic light sequence configurations accompanied by moderate speed limits (e.g. 50kph) can sometimes lead to higher travel time savings (approximately 10%) if the red phases are very long (e.g. more than one minute). In these cases, both optimization and simulated driver prefer the highest allowed velocity and the optimization is sometimes able to catch a short green phase, while the driver misses it and has to wait. In other scenarios, the optimization was able to guide the vehicle through a green phase (preceded by a long red phase) at high velocity, while in the same scenario the driver had previously already decelerated and therefore needed a long time to accelerate back to the preferred cruise speed. There were also a few cases with multiple traffic lights, in which the driver chose to drive through an early green phase, but had to stop completely at the next traffic light with extremely long red phases, while the optimization chose a later green phase at the first traffic light and was able to drive past the second traffic light with little deceleration. Scenarios with extremely low speed limits, e.g. 30kph usually lead to the smallest gains (down to 5% or less), largely because the optimization will choose the same cruise speed reference as the driver, namely the speed limit and sometimes also the same gear level depending on the definition of the scenario. More subtle discretization only leads to slightly higher fuel and time savings (less than 5%). This observation may change in future work when road elevation is regarded, for which predictive strategies have shown excellent results [2] [3]. In the future, the most pressing question is how much search space reduction leads to deviations from the true optimal result. In order to construct the globally optimal result, a fully expanded Dynamic Programming optimization will be performed that explicitly incorporates absolute time as a state variable and uses very subtle discretization. The resulting problem cannot be solved in real time, but can nevertheless serve as a reference.

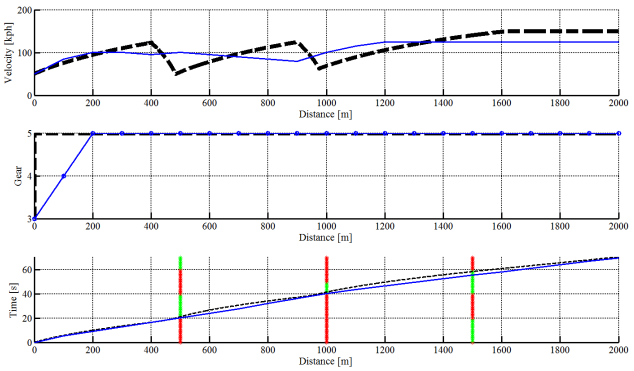

Fig. 3. Optimized profile (blue) in comparison to simulated driver (black dotted) in a randomly chosen scenario (best viewed in color). The road does not have inclination. The speed limit is randomly set to 150kph.

IV. CONCLUSION

The authors have presented a method to compute a predictive velocity and gear shift profile through a sequence of traffic lights. Because traffic light timings are hard constraints, absolute time is of great importance. In our search grid, absolute time was not directly used as a state component. Instead, absolute time was accumulated during the search and merely served as a constraint. This complexity reduction greatly reduces the search space, but can be theoretically suboptimal compared to a full expansion of all state components. Furthermore, a forward-backward Dynamic Programming adaptation using historic costs was briefly presented that further reduced computation time compared to regular Dynamic Programming. The combination of both strategies enabled real-time or close to real-time capabilities on a modern PC. For further details, the reader may also refer to one of our previous publications [17]. Preliminary evaluations based on simulation show great potentials in increasing fuel efficiency although more detailed investigation is needed.

In future work, the most pressing question is how much the search space reduction leads to deviations from the true optimal result. In order to construct the true optimal result as comparison, a fully expanded Dynamic Programming optimization will be performed that explicitly incorporates absolute time as a state variable. The resulting problem cannot be solved in real time, but can nevertheless serve as a reference. As our search space reduction implicitly uses continuous time, the reference optimization needs sufficiently subtle time discretization to recreate the same optimization scenario. This demand has emerged as a practical problem because subtle discretization can lead to a state grid that needs numerous hours or even days to evaluate. Therefore, efficient strategies need to be developed to conduct the experiments in reasonable time.

Furthermore, the optimization will shift to a newly developed electric vehicle created by our project partners [1]. The randomly chosen traffic light phases will be replaced by measurements from traffic lights that exist in the real world. A real route will be defined that incorporates elevation data from commercial height maps. A scientific, commercial driving simulator will be used to compare the energy consumption of regular drivers with the energy consumption of drivers or controllers guided by the model predictive optimization. The driving simulator will also introduce other traffic participants that will be incorporated into the optimization. The energy savings will eventually be verified in the real world power-train of the electric vehicle. Finally, it is desirable to test the real time capability of the final system in an automotive embedded system.

REFERENCES

[1] "Rem2030." `http://www.rem2030.de/`. Last access April 2015.

[2] E. Hellström, *Look-ahead Control of Heavy Trucks utilizing Road Topography*. PhD thesis, Linköpings universitet, 2007.

[3] S. Terwen, *Vorausschauende Längsregelung schwerer Lastkraftwagen*. PhD thesis, Karlsruhe Institut für Technologie, 2009.

[4] W. Huang, D. Bevly, S. Schnick, and X. Li, "Using 3d road geometry to optimize heavy truck fuel efficiency," *Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems*, 2008.

[5] B. Passenber, P. Kock, and O. Stursberg, "Combined time and fuel optimal driving driving of trucks based on a hybrid model," *European Control Conference*, pp. 4955–4960, 2009.

[6] T. can Keulen, G. Naus, B. de Jager, R. van de Molengraft, M. Steinbuch, and E. Aneke, "Predictive cruise control in hybrid electric vehicles," *World Electric Vehicle Journal Vol. 3*, 2009.

[7] R. Kamalanathsharma and H. Rakha, "Multi-stage dynamic programming algorithm for eco-speed control at traffic signalized intersections," *16th International IEEE Conference on Intelligent Transportation Systems*, 2013.

[8] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[9] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd ed., 2000.

[10] O. Chevrant-Breton, T. Guan, and C. W. Frey, "Search space reduction in dynamic programming using monotonic heuristic in the context of model predictive optimization," *IEEE Intelligent Transportation Systems*, 2014.

[11] E. Hellström, "Explicit use of road topography for model predictive cruise control in heavy trucks," Master's thesis, Linköpings universitet, 2005.

[12] H. G. Wahl, M. Holzäpfel, and F. Gauterin, "Approximate dynamic programming methods applied to far trajectory planning in optimal control," *IEEE Intelligent Vehicles Symposium*, 2014.

[13] B. Asadi and A. Vahidi, "Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time," *IEEE Transactions on Control Systems Technology*, 2011.

[14] R. Kamalanathsharma and H. Rakha, "Eco-driving at signalized intersections using v2i communication," *14th International IEEE Conference on Intelligent Transportation Systems*, 2011.

[15] Y. Chen, D. Zhang, and K. Li, "Enhanced eco-driving system based on v2x communication," *15th International IEEE Conference on Intelligent Transportation Systems*, 2012.

[16] H. Xia, K. Boriboonsomsin, F. Schweizer, A. Winckler, K. Zhou, W. Zhang, and M. Barth, "Field operational testing of eco-approach technology at a fixed-time signalized intersection," *15th International IEEE Conference on Intelligent Transportation Systems*, 2012.

[17] T. Guan and C. W. F, "Reuse historic costs in dynamic programming to reduce computational complexity in the context of model predictive optimization," *IEEE International Conference on Vehicular Electronics and Safety 2015*, 2015.

[18] L. Guzzella and A. Sciarretta, *Vehicle Propulsion Systems*. Springer, 2nd ed., 2005.

[19] L. N. U. Kiencke, *Automotive Control Systems*. Springer, 2010.