Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Real-Time Multiobjective Optimization Algorithm for Discovering Driving Strategies

Erik Dovgan, Matjaž Gams, Bogdan Filipič

To cite this article:
Erik Dovgan, Matjaž Gams, Bogdan Filipič (2019) A Real-Time Multiobjective Optimization Algorithm for Discovering Driving Strategies. Transportation Science 53(3):695-707. https://doi.org/10.1287/trsc.2018.0872

http://pubsonline.informs.org/journal/trsc/

TRANSPORTATION SCIENCE
Vol. 53, No. 3, May–June 2019, pp. 695–707
ISSN 0041-1655 (print), ISSN 1526-5447 (online)

A Real-Time Multiobjective Optimization Algorithm for Discovering Driving Strategies

Erik Dovgan,[a,b] **Matjaž Gams,**[a,c] **Bogdan Filipič**[a,c]

[a] Department of Intelligent Systems, Jožef Stefan Institute, SI-1000 Ljubljana, Slovenia; [b] Faculty of Computer and Information Science, University of Ljubljana, SI-1000 Ljubljana, Slovenia; [c] Jožef Stefan International Postgraduate School, SI-1000 Ljubljana, Slovenia

Contact: erik.dovgan@ijs.si, http://orcid.org/0000-0003-4671-8627 (ED); matjaz.gams@ijs.si, http://orcid.org/0000-0002-5747-0711 (MG); bogdan.filipic@ijs.si, http://orcid.org/0000-0003-4428-4255 (BF)

Received: September 8, 2014
Revised: February 17, 2017; January 12, 2018
Accepted: May 22, 2018
Published Online in Articles in Advance: April 25, 2019

https://doi.org/10.1287/trsc.2018.0872

Abstract. Vehicle driving consists of selecting and applying the best control actions in real time to optimize several objectives such as the traveling time and the fuel consumption. Because more than one objective is optimized, this problem can be solved using multiobjective optimization techniques. However, the existing optimization algorithms mostly combine objectives into a weighted-sum cost function and solve the corresponding single-objective problem. To test the multiobjective approach, we developed the multiobjective optimization algorithm for discovering driving strategies (MODS) that searches for the best driving strategies by taking into account the entire route. Although this algorithm, on average, outperforms existing single-objective algorithms for discovering driving strategies, it has a drawback, namely, it cannot be used for real-time optimization because of its time complexity. To overcome this shortage, we redesigned the MODS algorithm, obtaining the real-time multiobjective optimization algorithm for discovering driving strategies (MODS-RT). The MODS-RT algorithm was tested on data from real-world routes and compared with MODS and traditional single-objective algorithms for discovering driving strategies. Although MODS-RT found worse driving strategies than MODS, it found better driving strategies than the single-objective algorithms, thus proving that the multiobjective approach can be effectively adapted for real-time discovery of driving strategies.

Funding: This work was funded by the Slovenian Research Agency [Grants Z2-7581, P2-0209]. This work was also funded in part by NERVteh, Raziskave in Razvoj, d.o.o., and is part of a project that has received funding from the European Union's Horizon 2020 Research and Innovation Program [Grant 692286].

Keywords: real-time optimization • driving strategy • traveling time • fuel consumption • multiobjective optimization • black-box approach

1. Introduction

Autonomous vehicle driving is currently being investigated by many automotive and other companies, for example, Ford (Fitchard 2012), Mercedes-Benz (Ingraham 2013), Toyota (Read 2013), BMW (Elmer 2013), Audi (Johnson 2013), and Google (Rosen 2012). Several driver assistance systems are already installed in modern vehicles, such as lane assistance (see, e.g., Audi 2014, Toyota 2014, Volkswagen 2014). In addition, fully autonomous vehicles have begun driving in urban environments. For example, 100 self-driving Volvo cars will drive on public roads around the city of Gothenburg by 2021 (Elsom 2017).

Autonomous driving can be achieved by applying real-time control algorithms that select the best control action at each step with respect to current vehicle and route state. A set of connections between the vehicle and route states and control actions is a driving strategy. The best control action is selected by optimizing various objectives, such as the traveling time and the fuel consumption. The objectives have to be taken into account simultaneously because, in most cases, the improvement of one objective deteriorates another objective; for example, by minimizing the traveling time, more fuel is consumed.

Several techniques can be used to discover driving strategies. Most of them use single-objective optimization methods in combination with predictive control (PC; Del Re et al. 2010) and can be divided into two groups: model-based techniques and black-box techniques. Model-based techniques are analytical and require knowledge about the applied vehicle model. They optimize either a weighted sum of the fuel consumption and the traveling time (see, e.g., Huang et al. 2008;

Ivarsson, Aslund, and Nielsen 2008), or the fuel consumption only while considering the traveling time as a constraint (see, e.g., Khmelnitsky 2000; Howlett, Pudney, and Vu 2009; Melnik 2009). Black-box techniques are usually numerical and use the vehicle models without any knowledge of vehicle operation. Such techniques are more suitable for driving optimization because such knowledge is usually unavailable. In addition, they mainly use dynamic programming (DP) methods to find the driving strategies. Similarly to model-based approaches, they optimize either a weighted sum of the fuel consumption and the traveling time (see, e.g, Hooker, Rose, and Roberts 1983; Monastyrsky and Golownykh 1993; Hellstrom et al. 2009), or the fuel consumption only while considering the traveling time as a constraint (see, e.g., Johannesson, Pettersson, and Egardt 2009; Hellstrom, Aslund, and Nielsen 2010).

The previously presented techniques use single-objective methods to find driving strategies, although they optimize two objectives. Multiobjective techniques should be, in principle, better suited for solving such problems, because they enable better exploration of the multiobjective search space (Deb 2001). To explore the multiobjective approach, we implemented and tested a two-level multiobjective optimization algorithm for discovering driving strategies (MODS) that minimizes the traveling time and the fuel consumption (Dovgan et al. 2012, 2013, 2014). The lower-level algorithm is a deterministic multiobjective optimization algorithm based on breadth-first search (Russell and Norvig 2010), which uses multiobjective mechanisms based on fast nondominated sort and crowding distance mechanisms from the nondominated sorting genetic algorithm (NSGA-II; Deb et al. 2002). It searches for driving strategies and minimizes the traveling time and the fuel consumption. The optimal values of the input parameters for the lower-level algorithm are searched for by the upper-level algorithm. This is a single-objective evolutionary algorithm (Eiben and Smith 2003) that maximizes the hypervolume (Zitzler and Thiele 1999) covered by the driving strategies found by the lover-level algorithm. The results show that MODS finds better driving strategies than traditional single-objective optimization algorithms, namely, predictive control (Del Re et al. 2010) and dynamic programming (Hellstrom, Aslund, and Nielsen 2010). Although capable of finding good driving strategies, MODS has a disadvantage: its time complexity does not enable to produce good driving strategies in real time. High time complexity is the consequence of the following MODS properties: (1) the two-level structure of the algorithm, where the upper-level algorithm is a metaoptimization algorithm that optimizes the parameter values of the lower-level algorithm; and (2) the usage of the fast nondominated sort and crowding distance, whose original implementation does not fulfill high time constraints.

In this paper, we present a real-time multiobjective optimization algorithm for discovering driving strategies (MODS-RT) that finds good driving strategies in real time and optimizes the traveling time and the fuel consumption. It is a deterministic algorithm based on breadth-first search (Russell and Norvig 2010) and uses multiobjective mechanisms based on those from NSGA-II (Deb et al. 2002). The MODS-RT algorithm was derived from the lower-level algorithm of the two-level MODS algorithm (Dovgan et al. 2014; the upper-level MODS algorithm is not included in MODS-RT) by improving the search capabilities of driving prediction and significantly reducing the time complexity of the multiobjective mechanisms used in the MODS algorithm.

This paper is organized as follows. Section 2 describes the MODS-RT algorithm and the vehicle driving simulator used by MODS-RT to evaluate the driving strategies. The numerical experiments performed with MODS-RT, MODS, and two traditional algorithms for comparison, namely, predictive control and dynamic programming, are presented in Section 3. This section also describes how the vehicle behavior changes when the obtained driving strategies are applied to the vehicle. Finally, Section 4 concludes this paper with ideas for future work.

2. Real-Time Discovery of Driving Strategies with a Multiobjective Optimization Algorithm

This section presents a real-time multiobjective optimization algorithm for discovering driving strategies that uses a black-box driving simulator to search for driving strategies on a given route and minimizes the traveling time and the fuel consumption. First, the multiobjective optimization approach is described. Second, the black-box driving simulator is presented in terms of inputs, outputs, and internal behavior. Third, a detailed description of the MODS-RT algorithm is given.

2.1. Multiobjective Optimization Approach

Multiobjective optimization deals with optimization problems involving more than one objective function to be optimized simultaneously. In contrast to the single-objective optimization, it finds more than one solution, each with different trade-off between objectives. To find such a set of solutions, it is not enough to find an optimal solution corresponding to each objective function using single-objective approaches.

A multiobjective optimization problem consists of a number of objective functions (to be minimized or maximized) and constraints that any feasible solution must satisfy. Because each solution is evaluated with multiple objective functions, the comparison of solutions is performed using the dominance relation that is defined as follows (Deb 2001): a solution dominates

another solution if it is not worse in all objectives and strictly better in at least one objective. Consequently, if a solution is better in some and worse in other objectives when compared with another solution, these two solutions are incomparable. If a solution is not dominated by any other solution, it is called a nondominated solution. The dominance relation is used by most multiobjective optimization methods to search for nondominated solutions, that is, solutions that are not dominated by any other solution.

A multiobjective problem can be solved with (multiple simulation runs of) single-objective techniques by applying various approaches such as the weighted-sum approach. The weighted-sum approach weights and sums the objectives into a single function that is optimized afterward. However, such single-objective optimization depends on the selected weights. On the other hand, multiobjective optimization algorithms find a set of nondominated solutions without multiple simulation runs and without reduction of the optimization to a single objective function. To that end, they work with a population of solutions, which enables to capture a set of nondominated solutions in a single simulation run. An example is NSGA-II, which uses a crowded tournament selection operator to compare the solutions, and fast nondominated sort and crowding distance to select the best solutions for the new/improved population (Deb et al. 2002).

2.2. Vehicle Driving Simulator

The driving strategies found with MODS-RT and other algorithms were tested using a black-box driving simulator that we implemented based on the vehicle descriptions from Lechner and Naunheimer (1999), Blundell and Harty (2004), Randolph (2007), and Jazar (2008). The simulator receives the throttle and braking percentage ε_V and the gear g_V (i.e., the control action), simulates one step of Δs meters, and returns the spent time, the consumed fuel, the new vehicle state, and the driving feasibility. The driving is infeasible if the velocity limit is exceeded or if the vehicle stops.

To simulate one step, the following forces acting on the vehicle (Lechner and Naunheimer 1999) are taken into account:

- Engine moving force is the force produced by the engine when throttle percentage is greater than zero.
- Engine braking force is the force produced by the engine when throttle percentage is zero.
- Tire braking force is the force produced by brake pads when braking percentage is greater than zero.
- Wheel friction force is the force resisting the motion when the vehicle wheels roll on the road.
- Aerodynamic drag force is the force experienced by the vehicle moving through the air.
- The tangential component of the g-force is the force acting on the vehicle when the road is inclined.

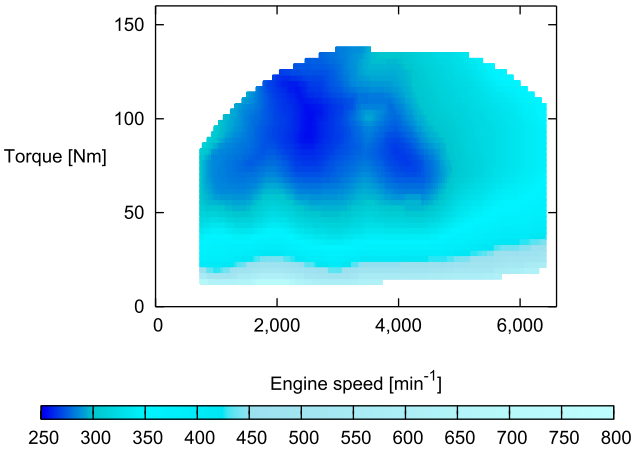
By combining these forces together, the inertial force is obtained, which causes the changes in vehicle velocity, that is, vehicle state. Next, the initial and final vehicle velocities are used to calculate the traveling time and assess the driving feasibility. Moreover, the fuel consumption is determined by taking into account the engine moving force and the specific fuel-consumption diagram (Randolph 2007) shown in Figure 1. The vehicle driving simulator is described in detail in Dovgan et al. (2014).

2.3. Real-Time Multiobjective Optimization Algorithm for Discovering Driving Strategies

MODS-RT is an optimization algorithm that searches for driving strategies in real time by minimizing the traveling time t and the fuel consumption c. The optimization at each step Δs searches for the best control action (i.e., discretized throttle and braking percentage $\varepsilon_V \in \boldsymbol{D}_\varepsilon$ and gear $g_V \in \boldsymbol{D}_g$) for the next step. The best control action is selected by predicting the vehicle driving for several steps ahead using multiobjective approaches and minimizing a weighted sum of traveling time t and fuel consumption c, that is, $f = \omega_c c + (1 - \omega_c)t$ (Lines 2–35 in Algorithm 1). After the best control action is discovered, it is applied to the vehicle for one-step simulation (Line 36). This procedure is repeated until the entire route has been simulated or the vehicle driving becomes infeasible (Lines 1–37).

The selection of the best control action for one step consists of predicting vehicle driving for N_P steps. This procedure starts by cloning the current driving strategy S and storing the clone in $\boldsymbol{S}_{\mathbf{pred,temp}}$ (Line 5). Afterward, each driving strategy in $\boldsymbol{S}_{\mathbf{pred,temp}}$ is cloned and $N_{P,p} \leq N_P$ steps are simulated for each possible control action $\{\varepsilon_V, g_V\}$ (Lines 9–16). If the obtained

Figure 1. (Color online) Specific Fuel-Consumption Diagram Shows the Fuel Consumption in Grams per Kilowatt Hour



Algorithm 1: MODS-RT

Data: ω_c, N_P

Result: driving strategy S

```
1  repeat
2      create empty set of predicted driving strategies S_pred
3      N_P,p = N_P
4      repeat
5          S_pred,temp = {S.clone()}, S*_pred,temp = {}
6          N_P,p,temp = 0
7          repeat
8              for S_temp ∈ S_pred,temp do
9                  for ε_V ∈ D_ε do
10                     for g_V ∈ D_g do
11                         S_clone = S_temp.clone()
12                         simulate driving of S_clone with control action {ε_V, g_V} for N_P,p steps
13                         if S_clone feasible then
14                             S*_pred,temp.add(S_clone)
15                         IF no time left THEN jump to Line 27
16                     IF no time left THEN jump to Line 27
17                 reduce the number of driving strategies in S*_pred,temp using elitism (see
                    Algorithm 2) and the enhanced Fast Nondominated Sort and Crowding
                    Distance mechanisms (see Algorithm 3)
18                 IF no time left THEN jump to Line 27
19             S_pred,temp = S*_pred,temp
20             N_P,p,temp = N_P,p,temp + N_P,p
21             IF no time left THEN jump to Line 27
22         until N_P,p,temp = N_P
23         S_pred = S_pred ∪ S_pred,temp
24         N_P,p = N_P,p / 2
25         IF no time left THEN jump to Line 27
26     until N_P,p < 1
27     {ε_V,best, g_V,best} = null
28     f_best = ∞
29     for S_pred ∈ S_pred do
30         {t, c} = S_pred.getObjectiveValues()
31         {ε_V, g_V} = S_pred.getControlActionOfTheFirstSimulationStep()
32         f_new = cω_c + t(1 − ω_c)
33         if f_new < f_best then
34             f_best = f_new
35             {ε_V,best, g_V,best} = {ε_V, g_V}
36     simulate driving of S with {ε_V,best, g_V,best} for one step
37 until S simulated the entire route or driving becomes infeasible
```

driving strategies are feasible, they are stored in $S^*_{\text{pred, temp}}$ for the next-step simulation. Such a procedure, similar to the breadth-first search, repeats until N_P steps have been simulated (Lines 7–22).

The cloning of driving strategies produces an exponentially growing number of driving strategies. To resolve this issue, elitism and multiobjective approaches based on fast nondominated sort and crowding distance mechanisms from NSGA-II (Deb et al. 2002) are used (Line 17). This enables us to discover the best driving strategies at each step and maintain a constant number of driving strategies S_{pop}. More precisely, elitism maintains the best $S_{pop,E}$ driving strategies with respect to the weighted sum of the traveling time and the fuel consumption as shown in Algorithm 2. The other $S_{pop} - S_{pop,E}$ driving strategies are selected using the enhanced fast nondominated sort and crowding distance. These mechanisms perform several comparisons between driving strategies to sort them with respect to the traveling time and the fuel consumption. When applied in the MODS-RT algorithm, they are used several times, each time operating on a union of S_{pop} already compared driving strategies and up to $|\boldsymbol{D}_\varepsilon| \times |\boldsymbol{D}_g|$ new driving strategies (obtained as shown in Lines 9–17). Such multiple usage is due to the fact that their time complexity is not linear; therefore, it is better to use them several times on a lower number of driving strategies instead of once on a large number of

Algorithm 2: Elitism operator

Data: $S^*_{\text{pred, temp}}$, ω_c, $S_{\text{pop, E}}$

Result: updated $S^*_{\text{pred, temp}}$

```
for S ∈ S*_pred,temp do
    {t, c} = S.getObjectiveValues()
    f = cω_c + t(1 − ω_c)
    S.setWeightedSum(f)
sort S*_pred,temp ascending with respect to weighted sum f
i = 0
repeat
    i = i + 1
    S*_pred,temp[i].setElite(true)
until i = S_pop,E
```

driving strategies. However, these mechanisms, each time, perform all the needed comparisons. Consequently, the comparisons between S_{pop} already compared driving strategies are repeated, which is not efficient. Therefore, these mechanisms were enhanced by storing and reusing previously performed comparisons as shown in Algorithm 3, which significantly reduced the computational time.

The prediction of vehicle driving at one step is computed for various $N_{\text{P,p}}$ steps. More precisely, the prediction is done for $N_{\text{P,p}} = N_{\text{P}}, N_{\text{P/2}}, N_{\text{P/4}}, N_{\text{P/8}}, \ldots, 1$ (Lines 4–26). When driving prediction ends (Line 26), all the feasible driving strategies are combined together, and the driving strategy that minimizes the weighted sum of the traveling time and the fuel consumption is selected (Lines 27–35). The first control action that was applied to the selected driving strategy is then used for simulating one step (Line 36).

During the driving prediction, the computational time is checked several times (Lines 15, 16, 18, 21, and 25). When the computational time is going to exceed the available computational time, the driving prediction stops, that is, the algorithm jumps to Line 27 and continues with the selection of the best control action for the next step. The computational time available for the selection of the control action for the next step is equal to the traveling time of the current step. For example, if the best control action for step n spends t_n seconds of the traveling time, than the available computational time for step $n+1$ is equal to t_n seconds. Therefore, during the traveling along step n, the vehicle has time to find the best control action for step $n+1$. MODS-RT is shown in Algorithm 1 and Figure 2. The differences between MODS-RT and its predecessor, MODS, are shown in Figure 3 (for the description of the MODS algorithm, see also Dovgan et al. 2014).

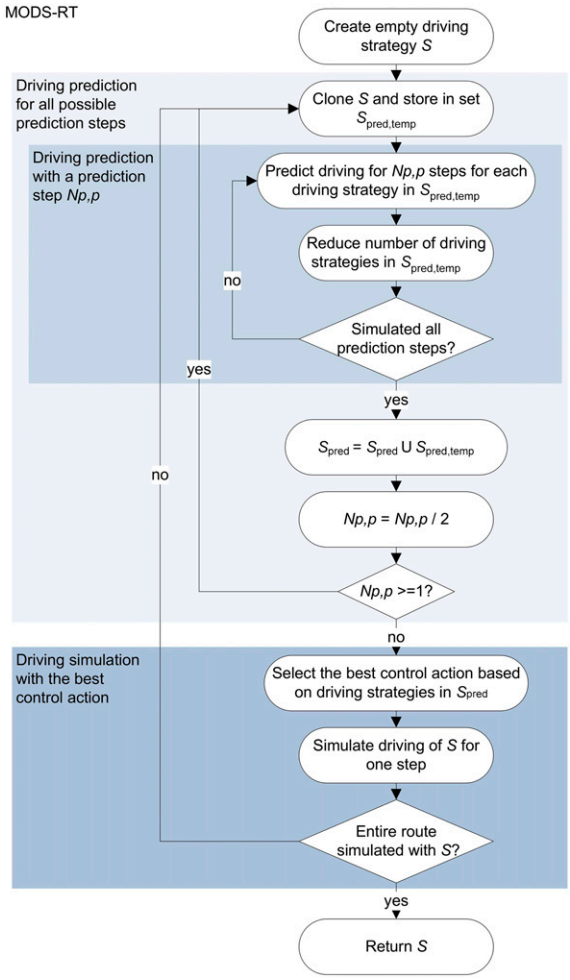
Algorithm 3: Enhanced Fast Nondominated Sort and Crowding Distance mechanisms

Data: $S^*_{\text{pred, temp}}$, S_{pop}, $S_{\text{pop, E}}$

Result: updated $S^*_{\text{pred, temp}}$

```
for S_1 ∈ S*_pred,temp and S_1 not already compared do
    for S_2 ∈ S*_pred,temp and S_2 already compared do
        compare S_1 and S_2
        if S_1 dominates S_2 then
            S_1.addDominatedDrivingStrategy(S_2)
            S_2.addDominatedByDrivingStrategy(S_1)
        if S_2 dominates S_1 then
            S_2.addDominatedDrivingStrategy(S_1)
            S_1.addDominatedByDrivingStrategy(S_2)
for S_1 ∈ S*_pred,temp and S_1 not already compared do
    for S_2 ∈ S*_pred,temp and S_2 not already compared do
        compare S_1 and S_2
        if S_1 dominates S_2 then
            S_1.addDominatedDrivingStrategy(S_2)
            S_2.addDominatedByDrivingStrategy(S_1)
        if S_2 dominates S_1 then
            S_2.addDominatedDrivingStrategy(S_1)
            S_1.addDominatedByDrivingStrategy(S_2)
find fronts and apply crowding distance to S*_pred,temp to discover the best non-elitist
S_pop − S_pop,E driving strategies by using the enhanced Fast Nondominated Sort and
Crowding Distance mechanisms
```

Figure 2. (Color online) A Schematic View of MODS-RT



3. Experiments and Results

3.1. Experimental Setup

The MODS-RT algorithm was tested on data from three real-world routes, which are presented in terms of their inclinations and velocity limits in Figures 4–6. The data on these routes were obtained from a database describing the main Slovenian routes provided by the Slovenian Roads Agency (2014). The obtained results were compared with the results of traditional algorithms, that is, PC (Del Re et al. 2010) and DP (Hellstrom, Aslund, and Nielsen 2010). In addition, they were also compared with the state-of-the-art multiobjective optimization algorithm for discovering driving strategies MODS (Dovgan et al. 2014). PC, DP, and MODS-RT are algorithms that combine the objectives into a weighted-sum function; therefore, the weight of objectives, ω_c, has to be given in advance. All of them predict vehicle driving to find the best control action for the current step, where the number of predictive steps, N_P, has to be given in advance. Therefore, to obtain driving strategies with various trade-offs between the traveling time and the fuel consumption, the weights ω_c were discretized and stored in the vector Ω. Afterward, all combinations of input parameter values $\{\omega_c, N_P\}$ were tested with each algorithm, where $\omega_c \in \Omega$ and $1 \leq N_P \leq N_{P,max}$. On the other hand, MODS finds a set of driving strategies with various trade-offs between objectives in one run. However, MODS is a stochastic algorithm; therefore, it was run 10 times on each route. Next, the driving strategies obtained in all runs were combined in 50% attainment curves (Fonseca and Fleming 1996).

Figure 3. (Color online) A Schematic View of Differences Between MODS-RT and Its Predecessor, MODS

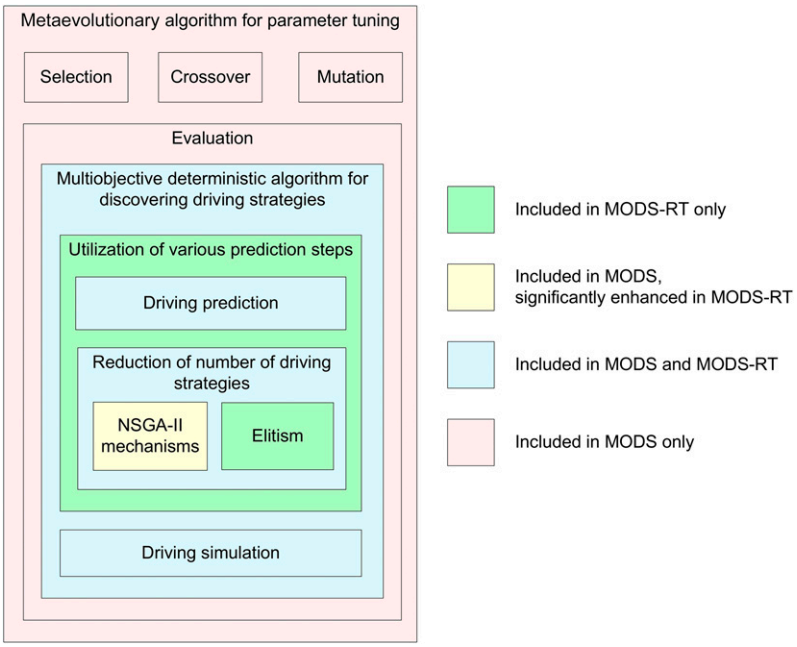
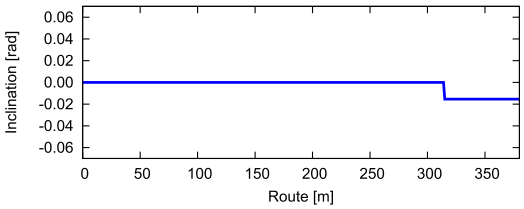
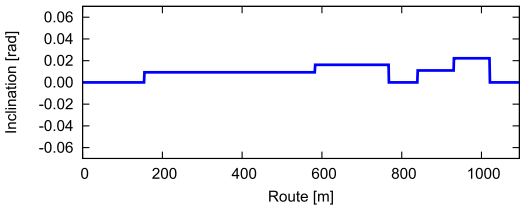


Figure 4. (Color online) Inclinations of the First Testing Route



Note. The velocity limit is 50 kilometers per hour along the entire route.

Figure 5. (Color online) Inclinations of the Second Testing Route



Note. The velocity limit is 50 kilometers per hour along the entire route.

A 50% attainment curve gives us an idea about the average performance of the algorithm because the region located to the top right of the curve is dominated by the driving strategies found in 50% of runs. MODS-RT, MODS, PC, and DP were tested with the input parameter values shown in Table 1. For other settings, such as parameter values of the vehicle simulator, see Dovgan et al. (2014).

3.2. Experimental Results

The nondominated driving strategies found with MODS-RT, PC, and DP and the 50% attainment curves obtained with MODS are shown in Figure 7. These results were obtained with a desktop computer with an Intel Core i7 3.5 GHz processor, with 16 GB of RAM and a 64-bit operating system. They show that MODS-RT found driving strategies that always dominate the driving strategies found with existing single-objective optimization algorithms, PC and DP. On the other hand, MODS found driving strategies that dominate those found with MODS-RT. This is also confirmed by the hypervolumes (Zitzler and Thiele 1999) covered by the obtained driving strategies, which are shown in Table 2. A higher hypervolume means that the algorithm found better driving strategies, that is, the driving strategies with lower traveling time and fuel consumption. MODS found better driving strategies than MODS-RT, and consequently obtained higher hypervolumes, because MODS searches for driving strategies by taking into account the entire route (global search), whereas MODS-RT searches for N_P steps ahead (local search) without considering the information about the rest of the route. Nevertheless, the MODS-RT driving strategies are similar to the MODS driving strategies in terms of the traveling time and the fuel consumption, or even better in some cases on the second and third routes.

The times needed by the algorithms to find good driving strategies are shown in Table 3 and Figure 8. In addition, Figure 8 contains a gray shaded area denoting the algorithms that meet real-time constraints, that is, the algorithms whose computational times for discovering driving strategies do not exceed the available traveling time. The table and figure show that PC and MODS-RT found driving strategies in real time, whereas DP and MODS spent 2.8 to 5.9 times more computational time than available. In summary, MODS-RT finds better driving strategies than traditional single-objective algorithms for discovering driving strategies, PC and DP. Although it does not find better driving strategies than MODS on average, it performs the search for driving strategies in real time, which is significantly faster than MODS. The actual time and space complexity of MODS-RT on the testing routes are shown in Table 4. The time complexity measured in floating point operations per second (FLOPS) FLOPS was determined based on the Lightspeed MATLAB toolbox.[1]

The results of MODS-RT could be, in theory, also compared with the globally optimal driving strategies, one driving strategy for each weight $\omega_c \in \Omega$. This could be done with exhaustive search by selecting a number of simulation steps N_P that cover the entire testing route and by keeping all the driving strategies at each step. This would result in exponential growth of the number of driving strategies, in contrast to a constant number of driving strategies maintained at each step by MODS-RT. Such a search space is equivalent to the

Figure 6. (Color online) Inclinations and Velocity Limits of the Third Testing Route

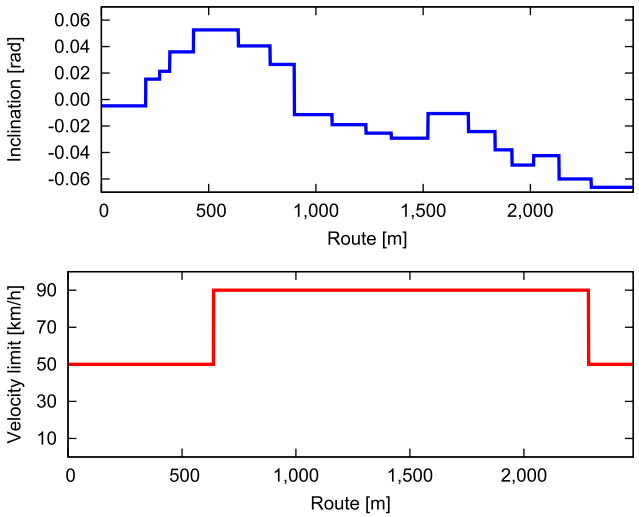


Table 1. The Parameter Values of MODS-RT, MODS, PC, and DP

Parameter	Value
Simulation step, Δs	5 m
Discretized throttle and braking percentages, $\boldsymbol{D}_\varepsilon$	[–1, –0.95, –0.9, –0.85, …, 0.9, 0.95, 1]
Gears, $\boldsymbol{D}_\mathbf{g}$	[1, 2, 3, 4, 5]
Number of driving strategies, S_{pop}	50
Number of elitist driving strategies, $S_{\text{pop,E}}$	5
Discretized weights of objectives, Ω	[0, 0.05, 0.1, …, 0.9, 0.95, 1]
Maximum number of predictive steps, $N_{\text{P,max}}$	50

full decision tree with branching factor $|\boldsymbol{D}_\varepsilon| \times |\boldsymbol{D}_\mathbf{g}|$. Because this search space is huge, it is not applicable to obtain globally optimal driving strategies as illustrated in Table 5. This table shows that the optimal driving strategies can be obtained only for very short routes, that is, routes of up to 15 meters, which is equivalent to three simulation steps. This table also shows that the maximum number of driving strategies grows exponentially with the number of simulation steps, although the actual number of driving strategies is lower because, when applying some control actions, the driving becomes infeasible and that subspace in not searched (this is equivalent to pruning the decision tree). Even though some subspaces are not searched, the required memory still grows exponentially with the number of simulation steps, which results in the premature end of the simulation because of the out-of-memory error already at four simulation steps. Thus, for four simulation steps, the table gives only estimates.

Examples of the vehicle's behavior when applying the driving strategies can be seen in Figures 9–11. Figure 9 shows the vehicle behavior on the first route for one nondominated driving strategy per algorithm—the strategy with a traveling time of around 45 seconds. Similarly, Figure 10 shows the driving strategies with a traveling time of around 1.5 minutes on the second route, and Figure 11 the driving strategies with a traveling time of around 4 minutes on the third route. These figures show that MODS-RT produces driving strategies with behavior similar to those of MODS, because both produce driving strategies with a more volatile vehicle velocity. On the other hand, PC and DP generate driving strategies with a more constant vehicle behavior. This is because PC and DP use the weighted-sum function for evaluating control actions with a constant weight. On the other hand, MODS-RT and MODS apply multiobjective search procedures to find the best control actions, which enables them to discover a larger set of driving strategies and therefore find better driving strategies without driving constantly in the same manner. Figures 7 and 9–11 also show that if the control actions rarely change, as in the cases when DP driving strategies are applied, worse driving strategies are

Figure 7. (Color online) Nondominated Driving Strategies in the Objective Space for the First Route (Top), Second Route (Middle), and Third Route (Bottom)

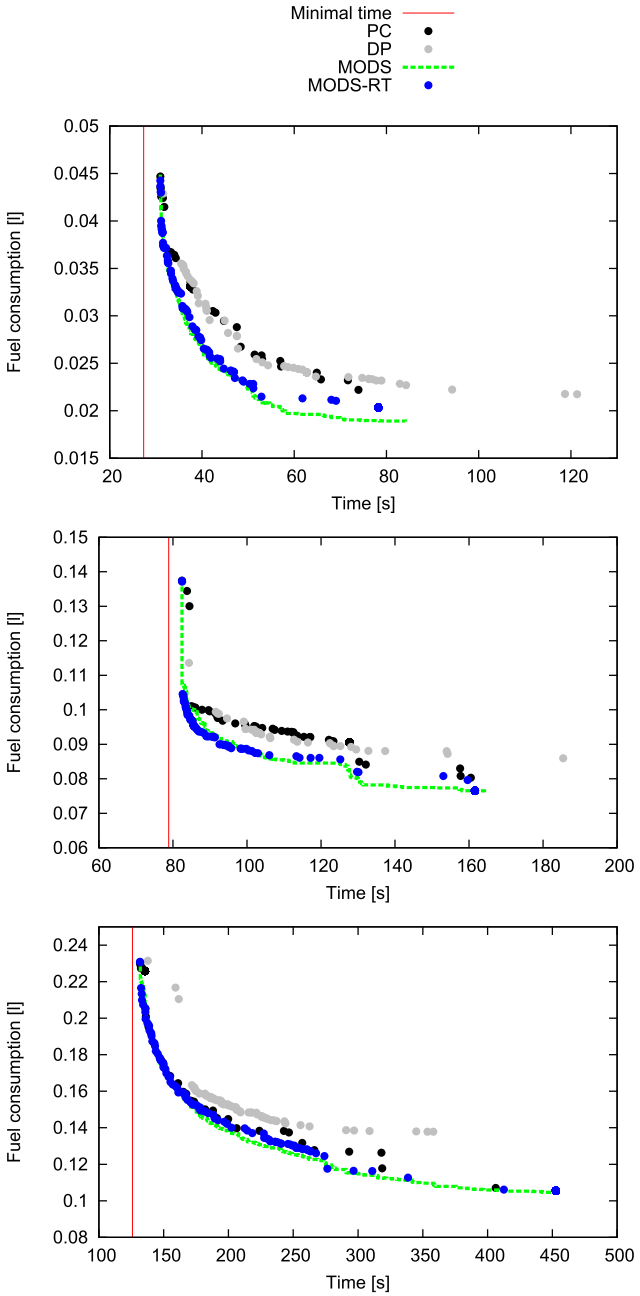
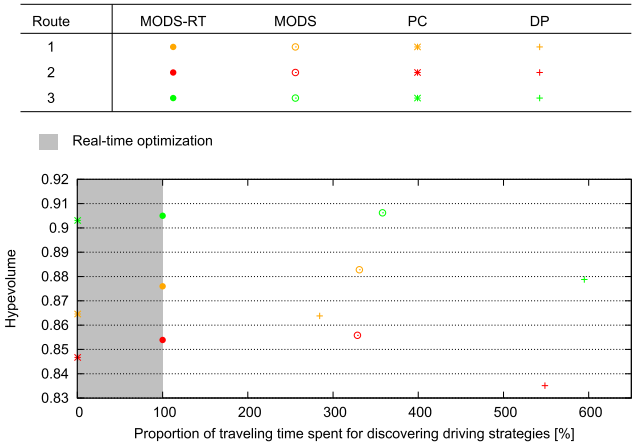


Table 2. Hypervolumes Obtained with the Applied Algorithms on the Testing Routes

Route	Algorithm	Hypervolume
1	PC	0.8646
	DP	0.8638
	MODS	0.8828 ± 0.0016
	MODS-RT	0.8760
2	PC	0.8467
	DP	0.8351
	MODS	0.8558 ± 0.0030
	MODS-RT	0.8539
3	PC	0.9031
	DP	0.8788
	MODS	0.9062 ± 0.0011
	MODS-RT	0.9050

Figure 8. (Color online) Hypervolumes Obtained with the Algorithms with Respect to Proportion of Spent Time



obtained. Consequently, when searching for good driving strategies, frequent changes of control actions are preferable.

Similar vehicle behavior based on driving strategies as shown in Figures 9–11 can be also seen in Figure 12. This figure shows the vehicle behavior based on driving strategies on the third, that is, the most complex, route for one nondominated driving strategy per algorithm—the strategy with a terminal velocity of around 50 kilometers per hour. More precisely, it shows that MODS and MODS-RT produce driving strategies with a more volatile driving behavior, whereas PC and DP generate driving strategies with a more constant vehicle behavior. The driving strategies shown in this figure are summarized in Table 6. This table shows that, among the presented driving strategies, MODS and MODS-RT driving strategies dominate PC and DP driving strategies.

The presented driving strategies minimize the traveling time and fuel consumption only, which might result in oscillatory behavior, either from the point of view of control actions or from the point of view of vehicle state, that is, vehicle velocity. Such a behavior might not be acceptable from a drivability point of view. To overcome this issue, an additional objective can be introduced, measuring the comfort as the derivative of acceleration. For more details, see Dovgan et al. (2012).

To sum up, PC and DP use one weight for each driving strategy, which results in a more constant driving behavior. On the other hand, MODS and MODS-RT apply the multiobjective search without predefining one weight per driving strategy. Consequently, the multiobjective approach enables us to discover a larger set of driving behaviors and find better driving strategies, which is confirmed by the results presented in Figures 7 and 9–12.

4. Conclusions

This paper presented a real-time multiobjective optimization algorithm for discovering driving strategies that minimizes the traveling time and the fuel

Table 3. Time Spent for Discovering Driving Strategies

Route	Algorithm	Average traveling time	Average computational time for discovering a driving strategy	Proportion of traveling time spent for discovering driving strategies (%)
1	PC	44 s	<1 s	0.3
	DP	55 s	2 min, 36 s	284.3
	MODS	49 s	2 min, 42 s	330.9
	MODS-RT	39 s	39 s	100.0
2	PC	1 min, 48 s	<1 s	0.3
	DP	1 min, 54 s	10 min, 25 s	548.6
	MODS	1 min, 45 s	5 min, 45 s	328.6
	MODS-RT	1 min, 35 s	1 min, 35 s	100.0
3	PC	3 min, 16 s	1 s	0.3
	DP	3 min, 51 s	22 min, 54 s	594.9
	MODS	3 min, 57 s	14 min, 10 s	358.1
	MODS-RT	3 min, 14 s	3 min, 14 s	100.0

Table 4. Complexity of the MODS-RT Algorithm

	Time complexity (MFLOPS)		Memory usage (MB)	
Route	Maximum	Average	Maximum	Average
1	2,260	670	318	106
2	2,170	736	530	124
3	1,900	627	365	193

consumption along a given route. This algorithm is based on breadth-first search (Russell and Norvig 2010) and multiobjective mechanisms from the non-dominated sorting genetic algorithm (Deb et al. 2002). Because it finds driving strategies in real time, it is suitable for deployment in adaptive cruise control of future intelligent vehicles, as suggested by Van Willigen, Haasdijk, and Kester (2013).

The MODS-RT algorithm was tested using a simulator and data from real-world routes, and the obtained driving strategies were compared with the driving strategies obtained with its predecessor, that is, MODS, and two traditional algorithms for comparison, predictive control and dynamic programming. The results show that MODS-RT performed better than PC and DP in terms of the quality of driving strategies measured by the traveling time and the fuel consumption. On the other hand, MODS-RT did not outperform MODS. Nevertheless, MODS-RT exhibited real-time performance, which was not the case with MODS. Therefore, the results show that MODS-RT found the best driving strategies when the real-time constraint was taken into account. Moreover, the multiobjective search procedures (utilized by MODS-RT and MODS) proved to be better suited for discovering driving strategies than single-objective ones (utilized by PC and DP). This is not surprising, because it is generally the case that multiobjective optimization algorithms exhibit advantages over single-objective ones. An additional analysis showed that the driving strategies obtained with multiobjective search procedures (MODS-RT and MODS driving strategies) produced more volatile vehicle velocity, in contrast to driving strategies obtained with single-objective search

Figure 9. (Color online) Examples of Vehicle Behavior on the First Route when Applying Driving Strategies with a Traveling Time of Around 45 Seconds

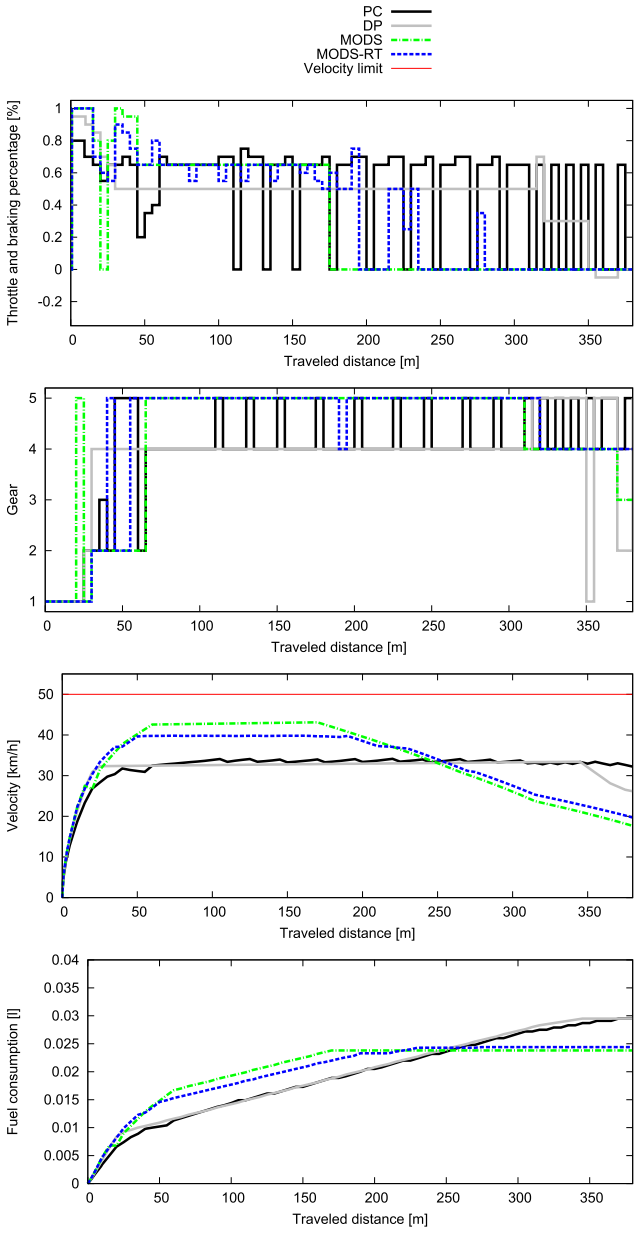


Table 5. Limitations of Discovering Globally Optimal Solutions with Exhaustive Search

		Number of driving strategies			
Route length (m)	Simulation steps	Maximum	Actual	Computational time	Max. memory usage (MB)
5	1	206	206	0.02 s	10
10	2	42,231	4,101	0.20 s	18
15	3	8,657,356	377,816	19.76 s	248
20	4	1,774,757,981	36,716,592*	27.21 min*	14,358*

Note. The values marked with an asterisk are estimated because the simulation ended prematurely because of the out-of-memory error.

Figure 10. (Color online) Examples of Vehicle Behavior on the Second Route when Applying Driving Strategies with a Traveling Time of Around 1.5 Minutes

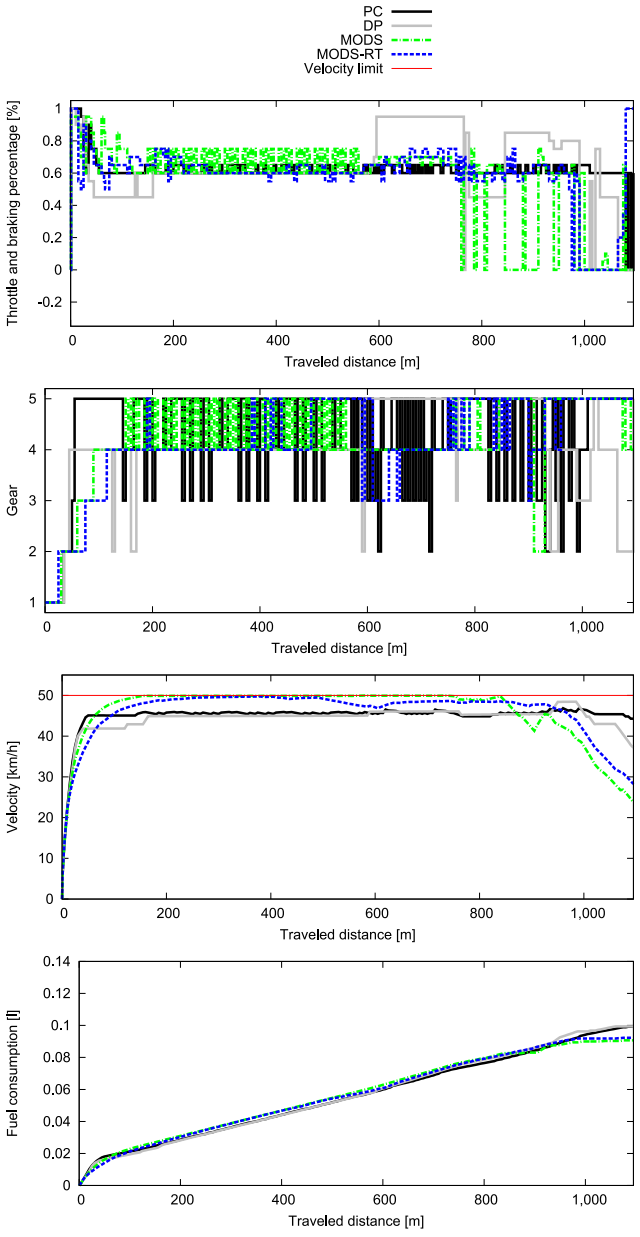


Figure 11. (Color online) Examples of Vehicle Behavior on the Third Route when Applying Driving Strategies with a Traveling Time of Around Four Minutes

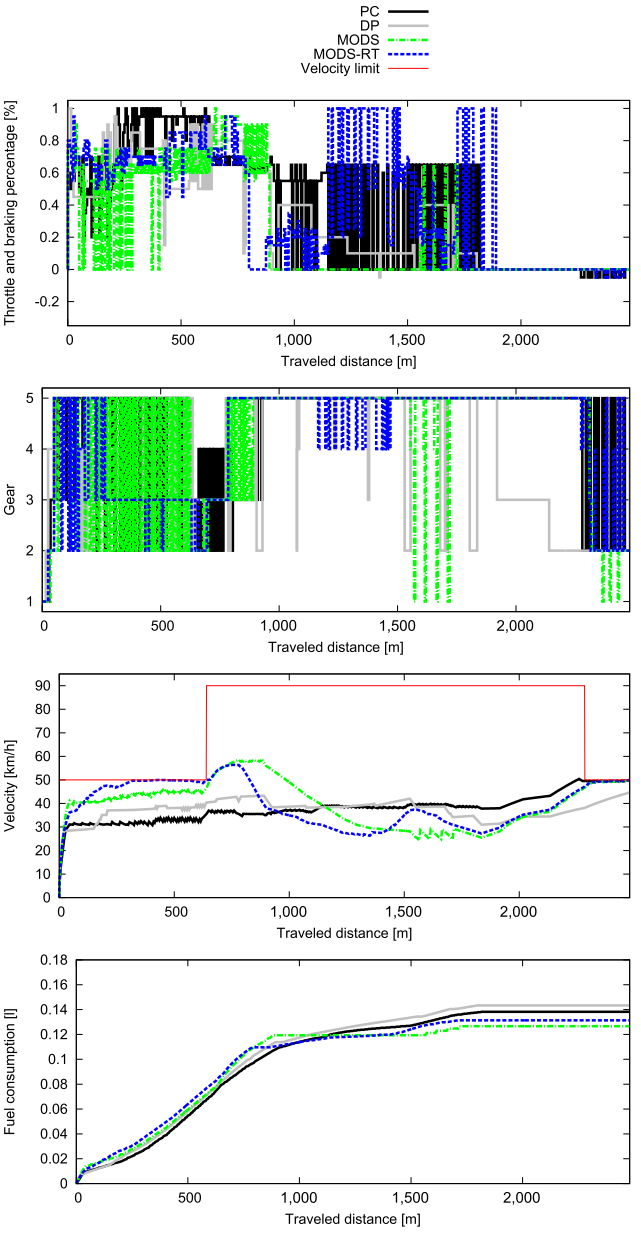
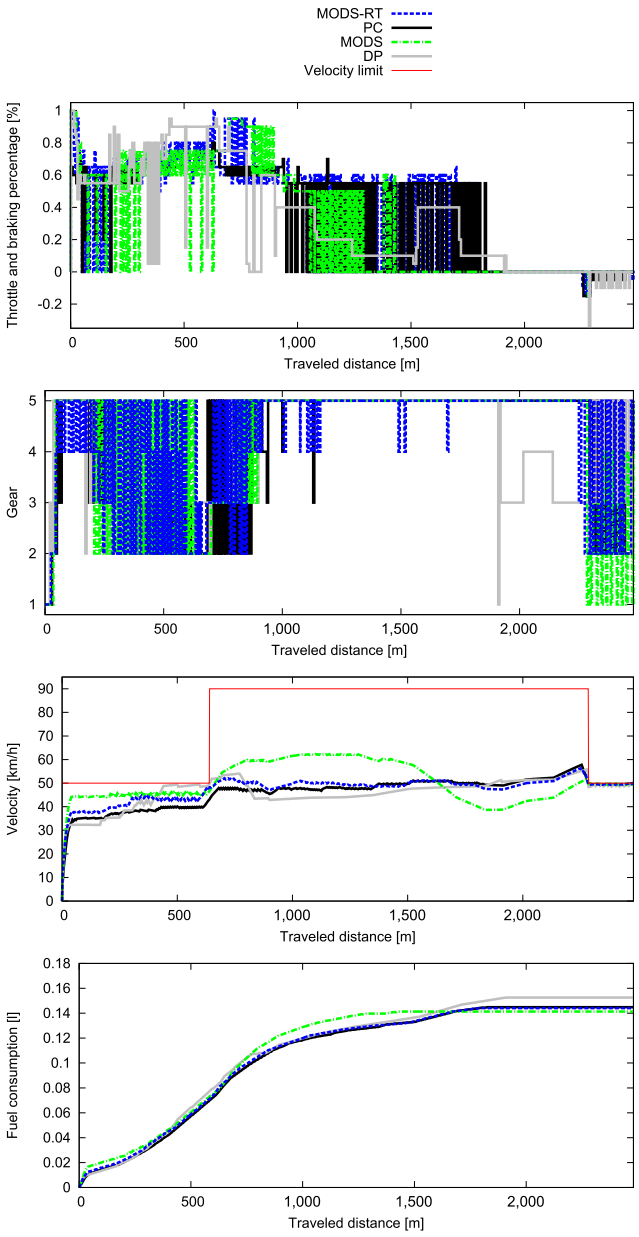


Figure 12. (Color online) Examples of Vehicle Behavior on the Third Route when Applying Driving Strategies with a Terminal Velocity of Around 50 Kilometers per Hour



procedures (PC and DP driving strategies), which resulted in more constant vehicle velocity.

In future work, we will test MODS-RT on additional routes. It would be interesting to include other vehicles and/or unexpected events in the simulation. Consequently, the MODS-RT algorithm should be enhanced to take this additional information into account appropriately and to include additional control actions into driving strategies, such as changing lanes and overtaking. A particular challenge would be the deployment of the algorithm in a real-life vehicle and its evaluation on a real route, where real-life neighboring vehicles and unexpected events have to be considered.

Table 6. Summary of Driving Strategies Shown in Figure 12

Algorithm	Time (s)	Fuel consumption (L)	Terminal velocity (km/h)
PC	199.8	0.145	50.0
DP	199.3	0.153	49.7
MODS	183.7	0.141	49.8
MODS-RT	191.9	0.144	49.8

Endnote

[1] See https://github.com/tminka/lightspeed (acessed February 6, 2017).

References

Audi (2014) Stay in lane. Accessed August 26, 2014, http://www.audi.co.uk/new-_cars/a3/a3-_sportback/driver-_assistants/audi-_active-_lane-_assist.html.

Blundell M, Harty D (2004) *The Multibody Systems Approach to Vehicle Dynamics* (Elsevier, London).

Deb K (2001) *Multi-Objective Optimization Using Evolutionary Algorithms* (John Wiley & Sons, Chichester, UK).

Deb K, Pratap A, Agrawal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Comput.* 6(2):182–197.

Del Re L, Allgower F, Glielmo L, Guardiola C, Kolmanovsky I (2010) *Automotive Model Predictive Control: Models, Methods and Applications*, Lecture Notes in Control and Information Sciences, vol. 402 (Springer, Berlin).

Dovgan E, Tušar T, Javorski M, Filipič B (2012) Discovering comfortable driving strategies using simulation-based multiobjective optimization. *Informatica* 36(3):319–326.

Dovgan E, Javorski M, Tušar T, Gams M, Filipič B (2013) Comparing a multiobjective optimization algorithm for discovering driving strategies with humans. *Expert Systems Appl.* 40(7):2687–2695.

Dovgan E, Javorski M, Tušar T, Gams M, Filipič B (2014) Discovering driving strategies with a multiobjective optimization algorithm. *Appl. Soft Comput.* 16(March):50–62.

Eiben AE, Smith JE (2003) *Introduction to Evolutionary Computing* (Springer, Berlin).

Elmer S (2013) BMW targets 2020 for self-driving cars. Accessed August 26, 2014, http://www.autoguide.com/auto-_news/2013/02/bmw-_targets-_2020-_for-_self-_driving-_cars.html.

Elsom D (2017) Volvo delays Drive Me driverless trials until 2021 over safety fears. *The Sun* (December 18), Accessed November 28, 2018, https://www.thesun.co.uk/motors/5163726/volvo-delays-drive-me-driverless-trials-until-2021-over-safety-fears/.

Fitchard K (2012) Ford is ready for the autonomous car. Are drivers? Accessed August 26, 2014, http://gigaom.com/2012/04/09/ford-_is-_ready-_for-_the-_autonomous-_car-_are-_drivers/.

Fonseca CM, Fleming PJ (1996) On the performance assessment and comparison of stochastic multiobjective optimizers. Ebeling W, Rechenberg I, Schwefel H-P, Voigt H-M, eds. *Proc. Parallel Problem Solving Nature IV* (Springer-Verlag, Berlin Heidelberg), 584–593.

Hellstrom E, Aslund J, Nielsen L (2010) Design of an efficient algorithm for fuel-optimal look-ahead control. *Control Engrg. Practice* 18(11): 1318–1327.

Hellstrom E, Ivarsson M, Aslund J, Nielsen L (2009) Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Engrg. Practice* 17(2):245–254.

Hooker JN, Rose AB, Roberts GF (1983) Optimal control of automobiles for fuel economy. *Transportation Sci.* 17(2):146–167.

Howlett PG, Pudney PJ, Vu X (2009) Local energy minimization in optimal train control. *Automatica* 45(11):2692–2698.

Huang W, Bevly DM, Schnick S, Li X (2008) Using 3D road geometry to optimize heavy truck fuel efficiency. Daily D, ed. *Internat. IEEE Conf. Intelligent Transportation* (Institute of Electrical and Electronics Engineers, Beijing, China), 334–339.

Ingraham N (2013) Mercedes-Benz shows off self-driving car technology in its new $100,000 S-Class. Accessed August 26, 2014, http://www.theverge.com/2013/5/18/4341656/mercedes-_benz-_shows-_off-_self-_driving-_car-_technology.

Ivarsson M, Aslund J, Nielsen L (2008) Optimal speed on small gradients – consequences of a non-linear fuel map. *Proc. 17th World Congress Internat. Federation Automatic Control*, 3368–3373.

Jazar RN (2008) *Vehicle Dynamics: Theory and Application* (Springer, New York).

Johannesson L, Pettersson S, Egardt B (2009) Predictive energy management of a 4QT series-parallel hybrid electric bus. *Control Engrg. Practice* 17(12):1440–1453.

Johnson D (2013) Audi predicts self-driving cars by 2020. Accessed August 26, 2014, http://www.leftlanenews.com/audi-_predicts-_self-_driving-_cars-_by-_2020.html.

Khmelnitsky E (2000) On an optimal control problem of train operation. *IEEE Trans. Automatic Control* 45(7):1257–1266.

Lechner G, Naunheimer H (1999) *Automotive Transmissions: Fundamentals, Selection, Design and Application* (Springer, Berlin).

Melnik RVN (2009) Coupling control and human factors in mathematical models of complex systems. *Engrg. Appl. Artificial Intelligence* 22(3):351–362.

Monastyrsky VV, Golownykh IM (1993) Rapid computation of optimal control for vehicles. *Transportation Res. Part B: Methodological* 27(3): 219–227.

Randolph T (2007) Waste heat regeneration systems for internal combustion engines. Accessed August 20, 2012, http://www.heat2power.net/downloads/GPC2007/20070618_heat2power_GPC_WHR_presentation.pdf.

Read R (2013) Toyota will roll out autonomous cars by the "mid-2010s." Accessed August 26, 2014, http://www.thecarconnection.com/news/1087636_toyota-_will-_roll-_out-_autonomous-_cars-_by-_the-_mid-_2010s.

Rosen RJ (2012) Google's self-driving cars: 300,000 miles logged, not a single accident under computer control. *The Atlantic* (August 9), Accessed August 26, 2014, http://www.theatlantic.com/technology/archive/2012/08/googles-_self-_driving-_cars-_300-_000-_miles-_logged-_not-_a-_single-_accident-_under-_computer-_control/260926/.

Russell SJ, Norvig P (2010) *Artificial Intelligence: A Modern Approach* (Prentice Hall, Upper Saddle River, NJ).

Slovenian Roads Agency (2014) Accessed August 26, 2014, http://www.dc.gov.si/.

Toyota (2014) Lane keeping assist. Accessed August 26, 2014, http://www.toyota-_global.com/innovation/safety_technology/safety_technology/technology_file/active/lka.html.

Van Willigen W, Haasdijk E, Kester L (2013) A multi-objective approach to evolving platooning strategies in intelligent transportation systems. Blum C, Alba E, Auger A, Barcardit J, Bongard J, Branke J, Bredeche N, et al., eds. *Proc. Genetic Evolutionary Comput. Conf.* (ACM, New York), 1397–1404.

Volkswagen (2014) Lane assist. Accessed August 2014, http://www.volkswagen.co.uk/technology/proximity-_sensing/lane-_assist.

Volvo (2013) Volvo car group initiates world unique Swedish pilot project with self-driving cars on public roads. Accessed August 2014, https://www.media.volvocars.com/global/en-_gb/media/pressreleases/136182/volvo-_car-_group-_initiates-_world-_unique-_swedish-_pilot-_project-_with-_self-_driving-_cars-_on-_public-_roads.

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evolutionary Comput.* 3(4):257–271.