A Multiobjective Optimization Algorithm for Discovering Driving Strategies

Erik Dovgan
Department of Intelligent Systems
Jožef Stefan Institute
Ljubljana, Slovenia
erik.dovgan@ijs.si

Matjaž Gams
Department of Intelligent Systems
Jožef Stefan Institute
Ljubljana, Slovenia
matjaz.gams@ijs.si

Bogdan Filipič
Department of Intelligent Systems
Jožef Stefan Institute
Ljubljana, Slovenia
bogdan.filipic@ijs.si

ABSTRACT

This paper presents a deterministic multiobjective optimization algorithm for discovering driving strategies. The goal is to find a set of nondominated driving strategies with respect to two conflicting objectives: time and fuel consumption. The presented multiobjective algorithm is based on the breadth-first search algorithm and Nondominated Sorting Genetic Algorithm (NSGA-II). Experiments on a 10-km route show that the results significantly depend on the discretization of the search space.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*control theory, graph and tree search strategies, heuristic methods*

General Terms

Design, Algorithms, Performance

Keywords

driving strategies, multiobjective optimization, traveling time, fuel consumption

1. INTRODUCTION

The cost of vehicle driving mainly depends on human time spent and the fuel cost. Optimizing just one of these objectives yields undesirable and often unrealistic results. Therefore, both time and fuel consumption have to be taken into account simultaneously when constructing a driving strategy.

Several researchers have optimized both time and fuel consumption by including them into a cost function as a weighted sum, or by optimizing only fuel consumption where time was a constraint. Then single objective optimization algorithms were used. These algorithms find only one solution which, in addition, significantly depends on the weights in the cost function. Moreover, it is not clear how to determine the weights. In order to find various driving strategies with respect to time and fuel consumption, a multiobjective approach has to be used. This approach produces a set of nondominated strategies that are incomparable since no strategy is better in both objectives than any other strategy. Such set of strategies enables the user to select the preferred one without limiting the time consumption in advance and without defining the weights. Consequently, a user, e.g., a transportation company, that frequently operates on the same route can each time select a strategy with different trade-off between time and fuel consumption based on current requirements.

This paper presents a multiobjective optimization algorithm that searches for driving strategies and minimizes time and fuel consumption. It is a deterministic algorithm based on a breadth-first search algorithm that includes mechanisms from the Nondominated Sorting Genetic Algorithm (NSGA-II) [4]. A strategy is a set hypercubes which are subspaces in the space of vehicle and route states, e.g., vehicle velocity and route inclination. A hypercube stores the control actions, e.g., throttle percentage and gear, that are applied to the vehicle if its state and position on the route correspond to the hypercube. The initial results show that the quality of the strategies significantly depends on the hypercube discretization.

The paper is organized as follows. Section 2 describes the related work in this field. The implemented driving simulation is presented in Section 3. Section 4 describes the proposed multiobjective optimization algorithm that searches for a set of nondominated driving strategies. The experiments and results are described in Section 5. Section 6 concludes the paper with the ideas for future work.

2. RELATED WORK

Several researchers studied the minimization of vehicle time and fuel consumption. However, they included the objectives into a single cost function and solved single objective problems. Monastyrsky et al. [11] implemented an algorithm based on a dynamic programming approach that finds a global optimum but can be used only for limited route lengths due to the complexity of dynamic programming. Ivarsson et al. [8] used an analytical approach that is appropriate only for routes with small gradients. In addition, this approach requires a lot of knowledge about the vehicle engine, e.g., specific fuel consumption diagram, which is usually unknown. Hellstrom et al. [6] used dynamic pro-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'11, July 12–16, 2011, Dublin, Ireland.
Copyright 2011 ACM 978-1-4503-0690-4/11/07 ...$10.00.

gramming as a predictive algorithm that searches for optimal driving strategy by taking into account only a finite route length ahead of the vehicle. A similar approach was presented by Huang et al. [7], who used constrained nonlinear programming.

Another problem the researchers focused on is the search for an optimal traveling route. For example, Benjamin [2] optimized the traveling time and the vehicle safety. Both objectives were included into a single objective function. The approach is appropriate only for limited space of possible routes since the whole space has to be searched at each simulation step.

However, only few researchers, e.g., in the field of racing games, focused on multiobjective optimization, but did not include both time and fuel consumption. For example, Agapitos et al. [1] studied the driving strategy optimization of racing game competitors based on several objectives, e.g., avoiding collisions and minimizing steering changes. They used a simplified vehicle model that considers the vehicle as a point that moves with constant velocity in the selected direction.

Most researchers focus on single objective optimization. Moreover, they search for strategies that include some knowledge on the vehicle operation. However, from the user point of view, such knowledge is unavailable. More precisely, a user can only predict the vehicle response since the exact vehicle response to his/her actions is not known. Therefore, the black box approach is the only reasonable one. In addition, if multiobjective problems are addressed, the researchers use over-simplified vehicle models. Searching for driving strategies by modeling a real vehicle as a black box driving on a real route and using a multiobjective optimization algorithm has not been tested yet. An initial version of an algorithm of this type is presented in this paper.

3. DRIVING SIMULATION

3.1 Driving Simulator

The driving simulator is implemented as a black box. It receives the data about the current vehicle velocity v_V, the route that has been already traveled s_V, the route that has to be traveled Δs_V, and the control actions, i.e., throttle and braking percentage ε_V and gear g_V. The throttle percentage is ε_V if $\varepsilon_V > 0$, otherwise it is 0. The braking percentage is $-\varepsilon_V$ if $\varepsilon_V < 0$, otherwise it is 0. It outputs the updated v_V and s_V, and the time t and fuel c consumption on the traveled route [5].

The route is divided into segments. Each segment is defined with its length s_S, turning radius r_S, inclination α_S, and velocity limit $v_{S,lim}$. The current segment is given with s_V. The final velocity limit $v_{S,limit}$ is the minimum limit among $v_{S,lim}$ and maximum turning velocity $v_T = \sqrt{rsg\cos\alpha_S c_s}$ [3], where g is the gravitational acceleration and c_s is the static friction coefficient.

To simulate the vehicle traveling on the route Δs_V, the vehicle simulator described in Section 3.2 is applied in steps, each step simulating vehicle driving at a constant velocity for a small time Δt. Afterwards, the vehicle velocity and position on the route are updated, Δt is added to t, and Δc is added to c. Finally, the velocity feasibility is checked. The driving is infeasible if $v_V > v_{S,limit}$ or $v_V \leq 0$. In this case the simulation ends.

3.2 Vehicle Simulator

The input data for the vehicle simulator are v_V, α_S, ε_V, g_V and Δt. The forces acting on the vehicle are the moving force F_E, engine braking force F_{EB}, tire braking force F_{WB}, wheel friction force F_W, inertial force F_a, aerodynamic drag force F_A, and tangential component of the gravitational force F_t. They are combined together as follows [10]: $F_E - F_{EB} - F_{WB} = F_W + F_a + F_A + F_t$.

The moving force is 0 if $\varepsilon_V \leq 0$. Otherwise, it is [10]: $F_E = \frac{T_E i_G[g_V] i_D}{r_W}\eta$, where engine torque $T_E = \varepsilon_V T_{E,max}$, engine speed $n_E = n_W i_G[g_V] i_D$, maximum engine torque $T_{E,max} = f_{T_{E,max}}(n_E)$, wheel speed $n_W = \frac{v_V}{2\pi r_W}$, T_W is wheel torque, r_W is wheel radius, i_G are gear ratios, i_D is differential ratio, η is vehicle transmissions mechanical efficiency, and $f_{T_{E,max}}(n_E)$ is maximum torque function.

The engine braking force is 0 if $\varepsilon_V > 0$. Otherwise, it is [10]: $F_{EB} = \frac{T_{EB} i_G[g_V] i_D}{\eta r_W}$, where engine braking torque T_{EB} is a linear function between ($n_{E,min}$, $T_{E,Min}$) and ($n_{E,max}$, $\frac{T_{E,Max}}{3}$). Here, $n_{E,min}$ and $n_{E,max}$ are minimum and maximum engine speeds, and $T_{E,Min}$ and $T_{E,Max}$ are minimum and maximum engine torques at any n_E. The tire braking force is 0 if $\varepsilon_V \geq 0$. Otherwise, it is [9]: $F_{WB} = \mu m_V g \cos\alpha_S \varepsilon_V$, where μ is tire braking force percentage, and m_V is vehicle mass. The wheel friction force is: $F_W = c_r m_V g \cos\alpha_S$, where c_r is the rolling resistance coefficient. The aerodynamic drag force is: $F_A = 0.5\rho v_V^2 A_x c_x$, where ρ is air density, A_x is vehicle frontal area, and c_x is vehicle aerodynamic coefficient. The tangential component of gravitational force is: $F_t = m_V g \sin\alpha_S$. The inertial acceleration is: $a_V = \frac{F_a}{m_V}$ [10].

The vehicle state and position are updated as follows [10]: $v_V = v_V + a_V\Delta t$, $\Delta c = f_c(T_E, n_E)P_E\Delta t$, $P_E = 2\pi T_E n_E$, $\Delta s = v_V\Delta t + \frac{a_V\Delta t^2}{2}$, where P_E is the engine power, Δc is the fuel consumption, $f_c(T_E, n_E)$ is the fuel consumption function, and Δs is the traveled route. When $v_V = 0$, the vehicle starts moving only if $g_V = g_{V,min}$ (and $a_V > 0$), since $T_{E,max} = 0$ if $n_E < n_{E,min}$ and $g_V > g_{V,min}$.

4. ALGORITHM FOR DISCOVERING DRIVING STRATEGIES

This section presents a deterministic multiobjective optimization algorithm for finding driving strategies based on the breadth-first search algorithm [13] and Nondominated Sorting Genetic Algorithm (NSGA-II) [4]. A strategy is a set of hypercubes which are subspaces in the space of vehicle and route states. This space has the following seven dimensions: vehicle gear g_V, vehicle engine speed n_E, current segment inclination α_{CS}, current segment velocity limit $v_{CS,limit}$, next segment inclination α_{NS}, next segment velocity limit $v_{NS,limit}$, and route to the next segment s_{NS}. Hypercubes store vehicle control actions ε_V and g_V.

The continuous space dimensions, i.e., n_E, α_S, $v_{S,limit}$, s_{NS} and ε_V, are discretized into intervals i_{n_E}, i_{α_S}, $i_{v_{S,limit}}$, $i_{s_{NS}}$ and i_{ε_V} by defining the interval bounds. Parameters α_S and $v_{S,limit}$ are discretized only once for both current and next segments. For example, α_S is discretized into n_{α_S} intervals by defining the vector of bounds $D_{\alpha_S} = [\alpha_{S,min}, \alpha_{S,1}, \alpha_{S,2}, \ldots, \alpha_{S,n_{\alpha_S}-1}, \alpha_{S,max}]$. A hypercube can be presented as a rule as follows:

IF $g_V = g_{V,RU}$, $n_E \in i_{n_{E,RU}}$, $\alpha_{CS} \in i_{\alpha_{CS,RU}}$, $v_{CS,limit} \in$

Algorithm 1 Multiobjective optimization algorithm for discovering driving strategies

$\boldsymbol{S_{pop}} = \{S_{init}\}$ {S_{init} is empty strategy}
$\boldsymbol{S_{final}} = \{\}$
repeat
  $\boldsymbol{S_{pop,nextStep}} = \{\}$
  for all $S \in \boldsymbol{S_{pop}}$ **do**
    $\boldsymbol{S_{pop,temp}} = \{\}$
    if {currently observed hypercube of strategy S stores control actions} **then**
      $\boldsymbol{S_{pop,temp}} = \{S\}$
    else
      for all $\boldsymbol{i_{\varepsilon_{V,control,RU}}}$ **do**
        for all $\boldsymbol{g_{V,control,RU}}$ **do**
          $S_{clone} = S.clone()$
          $S_{clone}.add(i_{\varepsilon_{V,control,RU}}, g_{V,control,RU})$
          $\boldsymbol{S_{pop,temp}}.add(S_{clone})$
        end for
      end for
    end if
    for all $S_{temp} \in \boldsymbol{S_{pop,temp}}$ **do**
      $S_{temp}.drivingSimulationForOneStep()$
      if S_{temp} simulated driving on whole route and feasible **then**
        $\boldsymbol{S_{final}}.add(S_{temp})$
      else if S_{temp} feasible **then**
        $\boldsymbol{S_{pop,nextStep}}.add(S_{temp})$
      end if
    end for
  end for
  $reduceNumberOfStrategies(\boldsymbol{S_{pop,nextStep}})$ {apply Fast Nondominated Sort and Crowding Distance [4]}
  $\boldsymbol{S_{pop}} = \boldsymbol{S_{pop,nextStep}}$
until $\boldsymbol{S_{pop}} = \{\}$
$\boldsymbol{S_{final}} = returnNondominatedStrategies(\boldsymbol{S_{final}})$ {apply Fast Nondominated Sort [4]}
return $\boldsymbol{S_{final}}$

$i_{v_{CS,limit,RU}}$, $\alpha_{NS} \in i_{\alpha_{NS,RU}}$, $v_{NS,limit} \in i_{v_{NS,limit,RU}}$, $s_{NS} \in i_{s_{NS}}$ **THEN** $i_{\varepsilon_{V,control,RU}}$, $g_{V,control,RU}$

The multiobjective optimization algorithm for discovering driving strategies is a deterministic algorithm that searches for driving strategies similarly to breadth-first search algorithm. It starts with a single strategy where none of the hypercubes stores the control actions. Afterwards, it simulates the driving of a set of strategies through several steps by using the algorithm described in Section 3 until the whole route has been traveled.

A step is defined with the route length Δs_V where control actions do not change. Control actions currently applied to the vehicle are stored in the hypercube that covers the subspace which includes the current vehicle and route state. Since the control actions do not change within a hypercube, the step Δs_V is defined with the route length interval $i_{s_{NS}}$ of the hypercube.

If the control actions of the observed hypercube are not defined yet, e.g., the initial strategy has no control actions, they have to be defined before the simulation continues. The number of combinations of control actions is $|i_{\varepsilon_{V,control,RU}}| \times |g_{V,control,RU}|$. For each combination, the strategy is cloned and the combination is assigned to the new strategy. Since the number of strategies grows, the maximum number of strategies is limited with the population size S_{pop}. This size is maintained by applying the functions Fast Nondominated Sort and Crowding Distance from the Nondominated Sorting Genetic Algorithm (NSGA-II) [4] at each simulation step. As a result, only the best and diverse strategies with respect to the objectives are preserved. The algorithm is shown in Algorithm 1.

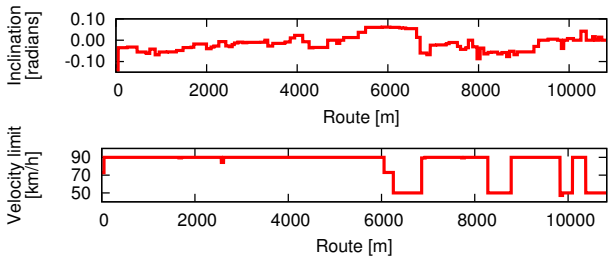

Figure 1: Inclinations and radii of the testing route.

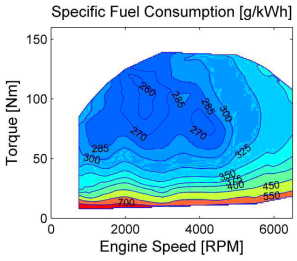

Figure 2: Specific fuel consumption diagram.

5. EXPERIMENTS

The presented algorithm has been tested on several routes. However, due to the space limit, we present the results of a test on a single route. Its length is 10829 m. It is presented in more detail in Figure 1. The vehicle parameter values used in the experiments are the following: $g_{V,min} = 1$, $g_{V,max} = 5$, $\eta = 0.8$, $\mu = 0.9$, $\rho = 1.225$ kg/m^3, $c_r = 0.04$, $c_s = 0.7$, $r_W = 0.33$ m, $m_V = 1700$ kg, $A_x = 2.16$ m^2, $c_x = 0.37$, $i_G = [3.45, 1.94, 1.28, 0.97, 0.80]$, $i_D = 3.67$, $n_{E,min} = 800$ min^{-1}, $n_{E,max} = 6400$ min^{-1}. Besides, the functions $f_{T_{E,max}}(n_E)$ and $f_c(T_E, n_E)$ defining the operation of the vehicle engine were derived from the specific fuel consumption diagram [12] shown in Figure 2.

The algorithm was run three times with $S_{pop} = 100$ and different hypercube discretizations. However, the hypercube gear dimension was always discretized into all five values. The other dimensions were discretized as shown in Table 1.

The results shown in Figure 3 indicate that the quality of the obtained strategies significantly depends on the hypercube discretization. For example, if only time is minimized, there are no significant differences among the discretizations. On the other hand, if only fuel consumption is minimized, the second and the third hypercube discretizations are better than the first one. Moreover, the strategies found using the second and the third discretization are incomparable, since better strategies are found with respect to both objectives with the second discretization, and better strategies are found with respect to fuel consumption only, using the third discretization.

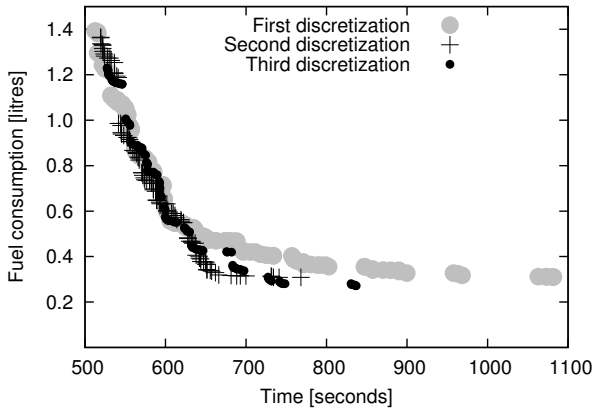

Figure 3: Nondominated strategies in the objective space.

These results show that a predefined hypercube discretization is not appropriate. Therefore, we are currently studying a two-level approach to discovering driving strategies. The lower level is based on the algorithm described in this paper. The upper level consists of an evolutionary algorithm that evolves the hypercube discretizations and forms a final set of nondominated strategies from all strategies found with the applied discretizations.

6. CONCLUSIONS

This paper presented a deterministic multiobjective optimization algorithm for discovering driving strategies. It searches for a set of nondominated strategies as sets of hypercubes and aims to minimize the time and fuel consumption. A hypercube is a subspace in the space of vehicle and route states that stores the vehicle control actions. The algorithm was tested using three different hypercube discretizations and the results show that the quality of strategies significantly depends on discretization and that no discretization is better than the other discretizations. Therefore, the future work will focus on the automatic search for appropriate discretizations. Moreover, the results will be compared to results of other single objective algorithms such as dynamic programming.

Table 1: Hypercube discretizations

Discretization	Variable	Unit	Min	Max	Step
1	D_{n_E}	min^{-1}	0	6500	250
	D_{α_S}	radians	−0.16	0.16	0.01
	$D_{v_{S,limit}}$	km/h	0	130	10
	$D_{s_{NS}}$	m	0	200	10
			200	400	50
			400	1000	100
	D_{ε_V}		−1	1	0.05
2	D_{n_E}	min^{-1}	0	6500	500
	D_{α_S}	radians	−0.16	0.16	0.02
	$D_{v_{S,limit}}$	km/h	0	130	20
	$D_{s_{NS}}$	m	0	200	20
			200	400	100
			400	1000	200
	D_{ε_V}		−1	1	0.1
3	D_{n_E}	min^{-1}	0	6500	1000
	D_{α_S}	radians	−0.16	0.16	0.04
	$D_{v_{S,limit}}$	km/h	0	130	40
	$D_{s_{NS}}$	m	0	200	50
			200	500	300
			500	1000	500
	D_{ε_V}		−1	1	0.2

7. REFERENCES

[1] A. Agapitos, J. Togelius, S. M. Lucas, J. Schmidhuber, and A. Konstantinidis. Generating diverse opponents with multiobjective evolution. In *IEEE Symposium on Computational Intelligence and Games*, pages 135–142, December 2008.

[2] M. R. Benjamin. Multi-objective autonomous vehicle navigation in the presence of cooperative and adversarial moving contacts. In *Proceedings of OCEANS*, volume 3, pages 1878–1885, October 2002.

[3] M. Blundell and D. Harty. *The Multibody Systems Approach to Vehicle Dynamics*. Elsevier, London, 2004.

[4] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pages 849–858, September 2000.

[5] E. Dovgan, M. Javorski, and B. Filipič. A multiobjective genetic algorithm for discovering car driving strategies. Technical Report IJS-DP 10402, Jožef Stefan Institute, Ljubljana, 2010.

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

[7] W. Huang, D. M. Bevly, S. Schnick, and X. Li. Using 3D road geometry to optimize heavy truck fuel efficiency. In *International IEEE Conference on Intelligent Transportation Systems*, pages 334–339, October 2008.

[8] M. Ivarsson, J. Aslund, and L. Nielsen. Optimal speed on small gradients – Consequences of a non-linear fuel map. In *The International Federation of Automatic Control*, pages 6–11, July 2008.

[9] R. N. Jazar. *Vehicle Dynamics: Theory and Application*. Springer, New York, 2008.

[10] G. Lechner and H. Naunheimer. *Automotive Transmissions: Fundamentals, Selection, Design and Application*. Springer, Berlin, 1999.

[11] V. V. Monastyrsky and I. M. Golownykh. Rapid computation of optimal control for vehicles. *Transportation Research Part B: Methodological*, 27(3):219–227, June 1993.

[12] T. Randolph. Waste heat regeneration systems for internal combustion engines. `http://www.heat2power.net/downloads/GPC2007/20070618_heat2power_GPC_WHR_presentation.pdf`, June 2007.

[13] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 2010.