Road Traffic Rules Synthesis using Grammatical Evolution

Eric Medvet, Alberto Bartoli, and Jacopo Talamini

Department of Engineering and Architecture, University of Trieste, Trieste, Italy

Abstract. We consider the problem of the automatic synthesis of road traffic rules, motivated by a future scenario in which human and machine-based drivers will coexist on the roads: in that scenario, current road rules may be either unsuitable or inefficient. We approach the problem using Grammatical Evolution (GE). To this end, we propose a road traffic model which includes concepts amenable to be regulated (e.g., lanes, intersections) and which allows drivers to temporarily evade traffic rules when there are no better alternatives. In our GE framework, each individual is a set of rules and its fitness is a weighted sum of traffic efficiency and safety, as resulting from a number of simulations where all drivers are subjected to the same rules. Experimental results show that our approach indeed generates rules leading to a safer and more efficient traffic than enforcing no rules or rules similar to those currently used.

Keywords: Simulation, Road traffic model, Stochastic evolution, Driverless cars

1 Introduction and related work

Car driving is one of the tasks that in a not far away future will be carried out by machines, rather than by humans. In a *driverless car* scenario a machine must be able to take a number of decisions in real time, with a limited and possibly noisy perception of the environment. Such decisions must take into account the need of abiding by the rules of the road and the presence of other moving, possibly hardly predictable, agents (pedestrian, bikers, other cars, either driverless or with a human driver).

Current traffic rules have been written for a scenario where humans drive cars and may hence be suboptimal in a driverless car scenario, or even in a scenario where both machines and humans drive cars. In this work, we take a fresh look at traffic rules and investigate the possibility of devising a novel set of rules that are amenable to automation and, at the same time, able to improve *global* indexes computed over the full population of vehicles. In particular, we focus on optimizing the global *traffic efficiency* and *safety*. We believe that, broadly speaking, an approach of this kind could lead to significant advantages to the society as a whole and that the driverless car revolution could offer a unique opportunity in this respect.

We propose a framework based on Grammatical Evolution (GE) and our contribution is as follows. First, we propose and experimentally evaluate a model for road traffic including the road graph, the cars, and the rules-aware drivers who try to abide by, but can possibly evade, the rules; the model is detailed enough to include concepts such as lanes, collisions, and safety distance, which are significant to our study. Second, we propose a language to define rules which can be enforced in our model: rules are predicates and the language is given in the form of a context-free grammar. Third and finally, we propose and experimentally evaluate a method for the automatic synthesis of rules based on GE: individuals represent sets of rules (i.e., regulations) and their fitness capture the degree to which, according to the results of several simulations, traffic regulated by the set of rules is efficient and safe.

Our experimental evaluation shows that, using GE, it is possible to obtain sets of rules which result in safer (less collisions) and more efficient traffic, w.r.t. both unregulated traffic and a set of hand-written rules designed to resemble a (simplified) real world set of rules.

To the best of our knowledge, no other studies concerning the automatic synthesis of road traffic rules have been proposed before. Recent research has focussed on how driverless cars should behave with respect to the existing rules: by proposing flexible control strategies which minimize the number of violated rules [14], by approaching the (highway) driverless algorithm design with rules complying as a first goal (legal safety) [15], or by formalizing rules for the sake of accountability after collisions involving driverless cars [10]. A much deeper problem in this area consists in determining which decision should be taken by a driverless car when facing a situation where only less-than-ideal outcomes are possible [5, 4]. In such a case, the decision may result in some sacrifice involving car users (passengers) or other people (e.g., pedestrians). This fundamental problem is orthogonal to our work.

Traffic is regulated not only by rules but also by the infrastructure, e.g., road markings and signs. In this area, several proposals have been made to modify the working principle of traffic lights in order to avoid congestion (e.g., [16]), also resorting to evolutionary computation [12]. More recently, motivated by the emergence of more automated vehicles, Tachet et al. even proposed the replacement of traffic lights with a novel (for road traffic) flow regulation solution build upon slot-based systems [13].

From a more general point of view, automatic synthesis of rules is a task which fits particularly well GE, since the language of the rules can be described in terms of a grammar and candidate solutions may be evaluated by means of simulated application of the corresponding set of rules: for instance, in [2] GE has been used to generate trading rules for spot foreign-exchange markets. Moreover, recent works showed that GE is suitable for addressing real problems with complex grammars, such as learning of string similarity functions [1], or designing components of vehicle routing algorithms [3].

2 Road traffic model

We consider a scenario with continuous space and discrete time in which a number of cars move according to their driving algorithms.

2.1 Roads and cars

A *road graph* is a directed graph $\mathcal{G} = (S, I)$ in which edges represent road sections, vertices represent road intersections, and where each vertex is connected to at least two edges. A *road section* $p \in S$ is characterized by a length $l(p) \in \mathbb{R}, l(p) > 0$ and a width $w(p) \in \mathbb{N}, w(p) > 0$: the former represents the length of the section between two intersections and the latter represents the number of lanes in the section. A *road intersection* $p \in I$ is characterized by a size $w(p) \in \mathbb{N}, w(p) > 0$: without loss of generality, we assume that the size of an intersection is equal to the largest width among the sections connecting the intersection.

A *car* is an agent which moves on the road graph. At a given time step, the car is positioned somewhere on the road graph, i.e., its position can be determined in terms of the section/intersection, lane and distance from the section/intersection origin. The car movement is determined in terms of two speeds, i.e., along the section and along the lanes—see Figure 1.

In detail, a car is a tuple (p, x, y, v_x, v_y, s) where p, x, y constitute the *position*, $v_x \in \mathbb{R}$ is the *linear speed*, $v_y \in \{-1, 0, 1\}$ is the lane-changing speed, and $s \in \{\text{alive}, \text{dead}\}$ is the *status*. Within the position, $p \in S \cup I$ is the section or intersection where the car is. If $p \in S$, $x \in [0, l(p)]$ and $y \in \{1, \ldots, w(p)\}$ are the linear and lane coordinates of the car within the road section p—we assume that $x = 0$ refers to the starting side of p. If $p \in I$, $x \in [0, w(p)]$ is the coordinate of the car within the intersection and y is not relevant.

At each time step, if the status of a car is $s = \text{dead}$, the position is not updated. Otherwise, if the status is $s = \text{alive}$, the position (p, x, y) of a car is updated as follows. If $p \in S$ and $0 \leq x + v_x \leq l(p)$, then its position at the next step is $(p, x + v_x, \min(\max(y + v_y, 0), w(p)))$. Otherwise, if $p \in S$ and $x + v_x < 0$ or $x + v_x > l(p)$, then its position at the next step is $(p', 0, 0)$, where $p' \in I$ is the appropriate intersection between the two connected by p. Otherwise, if $p \in I$ and $x + |v_x| \leq w(p)$ then its position at the next step is $(p, x + |v_x|, 0)$. Otherwise and finally, if $p \in I$ and $x + |v_x| > w(p)$, then its position at the next step is (p', x_0, y_0), where $p' \in S$ is one of the sections connecting p and $x_0 = 0, y_0 = 0$ or $x_0 = l(p'), y_0 = w(p')$ depending on whether p' starts or ends in p, respectively—in the latter case, if $v_x > 0$, then at the next step it is set to $-v_x$. Concerning the choice of p', let $\{p'_1, \ldots, p'_n\} \subseteq S$ be the set of sections connecting p, then p' is set to p'_j, where j is chosen randomly in $\{1, \ldots, n\}$. In other words, a car moves on a section according to its speeds; whenever it reaches the end (or the beginning) of the section, it enters the connected intersection. While in an intersection, the car remains inside according to its linear speed and the intersection size. When exiting an intersection, it enters a connecting section.

At each time step, a *collision* may occur between a pair of cars. A collision occurs if two cars meet on the same intersection or if two cars meet on the same

lane of the same road section. If a collision occurs, the status of the two cars is set to dead. In detail, let $(p_1^{(k)}, x_1^{(k)}, y_1^{(k)})$ and $(p_2^{(k)}, x_2^{(k)}, y_2^{(k)})$ be the positions of the two cars at time step k, a collision occurs at k if at least one of the following conditions is met:

- $p_1^{(k)} = p_2^{(k)} \in I$ and $\left|x_1^{(k)} - x_2^{(k)}\right| < d_{\text{collision}}$;
- $p_1^{(k)} = p_2^{(k)} \in S$ and $y_1^{(k)} = y_2^{(k)}$ and $\left|x_1^{(k)} - x_2^{(k)}\right| < d_{\text{collision}}$;
- $p_1^{(k)} = p_1^{(k-1)} = p_2^{(k)} = p_2^{(k-1)} \in I$ and $\text{sign}(x_1^{(k)} - x_2^{(k)}) \neq \text{sign}(x_1^{(k-1)} - x_2^{(k-1)})$;
- $p_1^{(k)} = p_1^{(k-1)} = p_2^{(k)} = p_2^{(k-1)} \in S$ and $y_1^{(k)} = y_2^{(k)}$ and $\text{sign}(x_1^{(k)} - x_2^{(k)}) \neq \text{sign}(x_1^{(k-1)} - x_2^{(k-1)})$;
- $p_1^{(k-1)} = p_2^{(k-1)} \in I$ and $p_2^{(k)} = p_2^{(k-1)}$ and $p_1^{(k)} \neq p_1^{(k-1)}$ and $x_1^{(k-1)} \leq x_2^{(k-1)}$;
- $p_1^{(k-1)} = p_2^{(k-1)} \in S$ and $p_2^{(k)} = p_2^{(k-1)}$ and $p_1^{(k)} \neq p_1^{(k-1)}$ and $y_1^{(k-1)} = y_2^{(k-1)}$ and $v_{x,1}^{(k)} > 0$ and $x_1^{(k-1)} \leq x_2^{(k-1)}$;
- $p_1^{(k-1)} = p_2^{(k-1)} \in S$ and $p_2^{(k)} = p_2^{(k-1)}$ and $p_1^{(k)} \neq p_1^{(k-1)}$ and $y_1^{(k-1)} = y_2^{(k-1)}$ and $v_{x,1}^{(k)} < 0$ and $x_1^{(k-1)} \geq x_2^{(k-1)}$;

where $d_{\text{collision}}$ is a parameter which represents the minimal distance between two cars—note that with any $d_{\text{collision}} > 0$ the capacity of the road graph is limited.

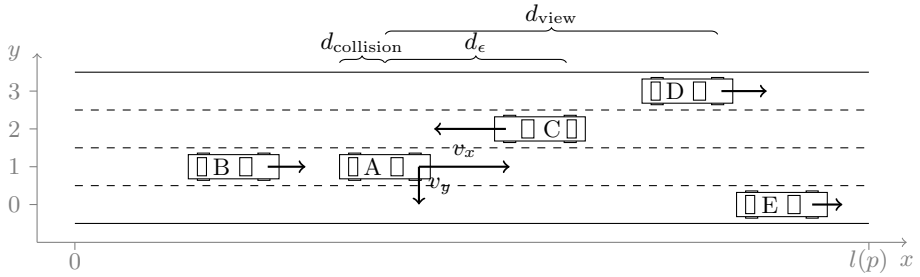


Fig. 1. A visualization of some of our model definitions. A car A is traveling with a positive linear speed v_x and a negative lane speed $v_y = -1$ on a road section on which there are other 4 cars. With respect to A, car C is the 1-lane closest car, because it is on $y_C = 2 = y_A + 1$ and its distance from A is $\Delta x \leq d_{\text{view}}$; moreover, $\delta v_1 =$ opposite, since C is traveling in the opposite direction w.r.t. A, and $\epsilon_1 =$ T, since C is closer than d_{epsilon} to A. There are no other j-lane closest cars for A: in facts B is behind (not ahead) A, D is two lanes away from A, and E distance to A is larger than d_{view}.

2.2 Driver

A *driver* is an algorithm according to which the linear and lane-changing speeds of a car are updated. In particular, at each time step, the algorithm execution is

based on the processing of (a) a set of input variables, (b) a set of state variables, (c) the driver's car tuple, (d) a set of unmodifiable parameters and results in (a) the output of a non-empty sequence of actions and (b) the modification of the state variables.

The input variables are based on the concept of *j-lane closest car*, with $j \in \{-1, 0, 1\}$. The j-lane closest car is the closest car ahead of the driver's car on the $y + j$ lane such that its linear distance is $\Delta x < d_{\text{view}}$, where y is the lane of the driver's car and d_{view} is a driver's parameter. For the sake of brevity, we omit a more formal definition, which covers also the case in which the driver's car is in (or close to) an intersection. Note that the j-lane closest car could not exist for some j, if no cars are closer than d_{view} or there is no $y + j$-th lane.

The *input variables* are the following—see Figure 1 for a visual interpretation of the variables.

- Three *relative movement* variables δv_{-1}, δv_0, and δv_1. The value of δv_j is defined in $\{\varnothing, \text{opposite}, -1, 0, 1\}$ and is determined as follows. If there is no j-lane closest car, then $\delta v_j = \varnothing$. Otherwise, let $(p', x', y', v'_x, v'_y, s')$ the j-lane closest car: if $\operatorname{sign} v_x \neq \operatorname{sign} v'_x$, then $\delta v_j = \text{opposite}$, otherwise $\delta v_j = \operatorname{sign}(|v'_x| - |v_x|)$. In other words, δv_j says if there is a j-lane closest car and, if any, if it moves in the opposite direction or, otherwise, is becoming closer ($\delta v_j = -1$), farther ($\delta v_j = 1$), or has the same linear speed, w.r.t. the driver's car.
- Three *closeness* variables $\epsilon_{-1}, \epsilon_0, \epsilon_1$. The value of ϵ_j is a boolean which is true if and only if there is a j-lane closest car and its distance Δx from the driver's car is $\Delta x \leq d_\epsilon$, where $d_\epsilon < d_{\text{view}}$ is a driver's parameter. In other words, ϵ_j is set if the j-lane closest car, if any, is closer than a threshold.

The *state variables* include a single variable $d \in \mathbb{R}, d \geq 0$, which represents the distance the driver still wants to go and which is updated at each time step as $d^{(k+1)} = d^{(k)} - |v_x^{(k)}|$. The *parameters* include d_{view}, d_ϵ (whose meaning was described above), a value $v_{\max} \in \mathbb{R}, v_{\max} \geq 0$, and a value $v_\Delta \in \mathbb{R}, 0 < v_\Delta < v_{\max}$: $v_{\max}$ is the maximum linear speed the driver's car can reach and v_Δ represents the acceleration/deceleration of the driver's car.

The output of the driver's algorithm is a non-empty sequence A of *actions*, i.e., an ordered subset of the set $\mathcal{A}$ of possible driver's action, with $\mathcal{A} = \{\uparrow, \nearrow, \rightarrow \searrow, \downarrow, \swarrow, \leftarrow, \nwarrow, \varnothing\}$. An action determines how v_x and v_y are updated, as shown in Table 1.

In other words, an up arrow corresponds to accelerating and a down arrow corresponds to braking; a right arrow corresponds to moving on the right lane and a left arrow corresponds to moving on the left lane, and so on. The driver executes only one of the actions in A. The action which is actually performed is chosen after processing A according to a procedure that is detailed in Section 2.4 which takes traffic rules into account.

The driver's algorithm is presented in the form of a multiway branch control in Table 2. The rationale for the proposed algorithm is to resemble the behavior of a "reasonable" driver who aims at traveling a distance d while avoiding trivial

Table 1. Driver's actions.

$a \in \mathcal{A}$	$v_x^{(k+1)}$	$v_y^{(k+1)}$
$\uparrow$	$\operatorname{sign}(v_x^{(k)} + v_\Delta) \min(\lvert v_{\max}\rvert, \lvert v_x^{(k)}\rvert + v_\Delta)$	0
$\nearrow$	$\operatorname{sign}(v_x^{(k)} + v_\Delta) \min(\lvert v_x^{(k)}\rvert, \lvert v_x^{(k)}\rvert + v_\Delta)$	$-\operatorname{sign} v_x^{(k)}$
$\rightarrow$	$v_x^{(k)}$	$-\operatorname{sign} v_x^{(k)}$
$\searrow$	$\operatorname{sign}(v_x^{(k)} + v_\Delta) \max(0, \lvert v_x^{(k)}\rvert - v_\Delta)$	$-\operatorname{sign} v_x^{(k)}$
$\downarrow$	$\operatorname{sign}(v_x^{(k)} + v_\Delta) \max(0, \lvert v_x^{(k)}\rvert - v_\Delta)$	0
$\swarrow$	$\operatorname{sign}(v_x^{(k)} + v_\Delta) \max(0, \lvert v_x^{(k)}\rvert - v_\Delta)$	$\operatorname{sign} v_x^{(k)}$
$\leftarrow$	$v_x^{(k)}$	$\operatorname{sign} v_x^{(k)}$
$\nwarrow$	$\operatorname{sign}(v_x^{(k)} + v_\Delta) \min(\lvert v_x^{(k)}\rvert, \lvert v_x^{(k)}\rvert + v_\Delta)$	$\operatorname{sign} v_x^{(k)}$
$\varnothing$	$v_x^{(k)}$	0

collisions: that goal is pursued by—in essence—trying to travel at the max linear speed while avoiding hitting other cars on the same lane. In detail, each row of Table 2 represents a proposition and the corresponding output A. The proposition is composed of the conjunction of membership checks on input variables and on the result of the comparison of the state variable d against x_{stop}, which is defined as $x_{\text{stop}} = k_{\text{stop}} v_x + \frac{1}{2}\left(k_{\text{stop}}^2 - k_{\text{stop}}\right) v_\Delta$ where $k_{\text{stop}} = \left\lceil \frac{v_x}{v_\Delta} \right\rceil$; x_{stop} represents the distance the driver's car would run if it constantly decrease its speed until stopping. For instance, row 4 proposition is $\delta v_0 \in \{-1, 0\} \wedge \epsilon_0 \in \{\text{F}\}$, row 5 proposition is $\delta v_1 \in \{\varnothing, 1\} \wedge \delta v_0 \in \{-1, 0\} \wedge \epsilon_1 \in \{\text{F}\} \wedge \epsilon_0 \in \{\text{T}\}$. The output is determined as follows: if the row 1 proposition is true, then the output is the row 1 A; otherwise, if the row 2 proposition is true, then the output is the row 2 A, and so on—note that the last row proposition is always true, hence it is guaranteed that a non empty sequence is always output.

2.3 Rules

A *traffic rule* is a predicate defined on a set of variables concerning a car and its driver, its j-lane closest cars, and the road graph. A car *breaks* a rule at a given time step if the corresponding predicate is false.

The variables on which a traffic rule can be defined include: (a) variables concerning the car and the corresponding driver: $\hat{v}_x$, $v_{\max}$, v_Δ, d_{view}, d_ϵ, p, $\hat{x}$, and $\hat{y}$, where $\hat{v}_x = |v_x|$, $\hat{x} = l(p) - x$ and $\hat{y} = y$, if $v_x \geq 0$, and $\hat{x} = x$ and $\hat{y} = w(p) - y$, otherwise; (b) variables concerning the car j-lane closest cars: δv_{-1}, δv_0, δv_1, Δx_{-1}, Δx_0, and Δx_1, where Δx_j is defined as in Section 2.2 and is set to $+\infty$ if the corresponding $\delta v_j = \varnothing$; (c) variables concerning the road graph section or intersection in which the car is: $l(p)$ and $w(p)$.

The set of possible traffic rules is defined by a context-free grammar which we here present with the Backus-Naur Form (r is the rule) in Figure 2.

Table 2. The driver's algorithm.

	δv_1	δv_0	δv_{-1}	ϵ_1	ϵ_0	ϵ_{-1}	$d \leq x_{\text{stop}}$	A
1	$\forall$	$\forall$	$\forall$	$\forall$	$\forall$	$\forall$	{T}	$\{\downarrow, \searrow, \swarrow\}$
2	$\{\varnothing, 1\}$	$\{\varnothing, 1\}$	$\{\varnothing, 1\}$	$\forall$	$\forall$	$\forall$	$\forall$	$\{\uparrow, \nearrow, \nwarrow, \varnothing, \downarrow\}$
3	$\forall$	$\{\varnothing, 1\}$	$\forall$	$\forall$	$\forall$	$\forall$	$\forall$	$\{\uparrow, \varnothing, \downarrow\}$
4	$\forall$	$\{-1, 0\}$	$\forall$	$\forall$	{F}	$\forall$	$\forall$	$\{\uparrow, \varnothing, \downarrow\}$
5	$\{\varnothing, 1\}$	$\{-1, 0\}$	$\forall$	{F}	{T}	$\forall$	$\forall$	$\{\nwarrow, \varnothing, \downarrow\}$
6	$\forall$	$\{-1, 0\}$	$\{\varnothing, 1\}$	$\forall$	{T}	{F}	$\forall$	$\{\nearrow, \varnothing, \downarrow\}$
7	$\forall$	{0}	$\forall$	$\forall$	{T}	$\forall$	$\forall$	$\{\varnothing, \downarrow\}$
8	$\forall$	$\{-1\}$	$\forall$	$\forall$	{T}	$\forall$	$\forall$	$\{\downarrow\}$
9	$\forall$	{opposite}	$\forall$	$\forall$	$\forall$	{F}	$\forall$	$\{\rightarrow, \searrow, \downarrow\}$
10	$\forall$	{opposite}	$\forall$	{F}	$\forall$	$\forall$	$\forall$	$\{\leftarrow, \swarrow, \downarrow\}$
11	$\forall$	{opposite}	$\forall$	$\forall$	{T}	$\forall$	$\forall$	$\{\searrow, \swarrow, \downarrow\}$
12	$\forall$	$\forall$	$\forall$	$\forall$	$\forall$	$\forall$	$\forall$	$\{\downarrow, \searrow, \swarrow\}$

For example, the rule stating that "the maximum speed of a car is 20" is written as $\hat{v}_x \leq 20$. The rule stating that "the car should stay on the rightmost free lane" is written as $\hat{y} \leq 0 \vee \Delta x_{-1} \leq 10$, where 10 represents a distance within which a lane is not considered free. The rule stating that "the car should proceed slowly when approaching an intersection" is written as $\neg\hat{x} \leq 20 \vee \hat{v}_x \leq 10$.

2.4 Rules-aware driver

A *rules-aware driver* is a driver that selects exactly one *winning action* out of a sequence $A = (a_1, a_2, \dots)$ of actions, given a set R of traffic rules. In brief, a rules-aware driver selects the action which, if repeated for the next steps, will cause the least number of broken rules.

More in detail, the selection is performed as follows. First, for each action a_i in A, the sequence A_i of future actions consisting of a_i repeated k_{advance} times is considered. Second, the number n_i of future broken rules caused by A_i is computed as the sum of the number of rules that would be broken at each future step $k + j$, with $0 \leq j \leq |A_i|$. Third and finally, the winning action is determined as the action $a_{i^\star}$ for which $\frac{n_{i^\star}}{|A_{i^\star}|}$ is the lowest—in case of tie, the action with the lowest index in A is selected. When computing n_i, the rules-aware driver predicts the future variable values assuming that: (i) the j-lane closest cars, if any, will maintain the same speeds of step k; (ii) no other j-lane closest cars will appear; (iii) the driver's car will update consistently with the sequence of actions A_i. If a sequence A_i is such that the future position p of the car changes, the sequence is truncated to the last element before that change.

3 Grammatical Evolution

Grammatical Evolution (GE) [11] is a form of grammar-based Genetic Programming (GP) [6] which can evolve strings belonging to a language $\mathcal{L}(\mathcal{G})$ defined

$$
\begin{aligned}
r &::= \langle \text{conditions} \rangle \\
\langle \text{conditions} \rangle &::= \langle \text{condition} \rangle \mid \langle \text{conditions} \rangle \vee \langle \text{condition} \rangle \\
\langle \text{condition} \rangle &::= \langle \text{baseCondition} \rangle \mid \neg \langle \text{baseCondition} \rangle \\
\langle \text{baseCondition} \rangle &::= \langle \text{numericCondition} \rangle \mid \langle \text{deltaCondition} \rangle \mid \langle \text{graphCondition} \rangle \\
\langle \text{numericCondition} \rangle &::= \langle \text{numericVariable} \rangle \leq \langle \text{numericValue} \rangle \\
\langle \text{numericVariable} \rangle &::= \hat{v}_x \mid v_{\max} \mid v_\Delta \mid d_{\text{view}} \mid d_\epsilon \mid \hat{x} \mid \hat{y} \mid \Delta x_{-1} \mid \Delta x_0 \mid \Delta x_1 \mid l(p) \mid w(p) \\
\langle \text{numericValue} \rangle &::= \langle \text{digit} \rangle . \langle \text{digit} \rangle \text{E} \langle \text{exp} \rangle \\
\langle \text{digit} \rangle &::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\
\langle \text{exp} \rangle &::= -1 \mid 0 \mid 1 \\
\langle \text{deltaCondition} \rangle &::= \delta v_{-1} = \langle \text{deltaValue} \rangle \mid \delta v_0 = \langle \text{deltaValue} \rangle \mid \delta v_1 = \langle \text{deltaValue} \rangle \\
\langle \text{deltaValue} \rangle &::= \varnothing \mid \text{opposite} \mid -1 \mid 0 \mid 1 \\
\langle \text{graphCondition} \rangle &::= p \in S
\end{aligned}
$$

Fig. 2. Backus-Naur Form of the context-free grammar for the traffic rules.

by a context-free grammar $\mathcal{G}$. In brief, GE operates on *genotypes*, which are variable-length bit strings, maps them to *phenotypes*, which are strings of $\mathcal{L}(\mathcal{G})$, and finally associates them with fitness values in $\mathbb{R}$.

The genotype-phenotype mapping procedure is the distinguishing trait of GE. In this procedure, the genotype is viewed as a variable-length integer string where each i-th integer, called *codon* and denoted by g_i, is obtained by decoding bits from the $(8(i-1))$-th to the $(8i-1)$-th, included, in the genotype g. The procedure is iterative and starts by setting the phenotype to $p = s_0$, s_0 being the grammar starting symbol, a counter i to 0, and a counter w to 0. Then, the following steps are iterated.

1. The leftmost non-terminal s in p is expanded using the j-th option (zero-based indexing) in the production rule r_s for s in $\mathcal{G}$, with j being the remainder of the division between the value g_i of the i-th codon (zero-based indexing) and the number $|r_s|$ of options in r_s, i.e., $j = g_i \mod |r_s|$.
2. The counter i is incremented by 1; if i exceeds the number of codons, i.e., if $i > \frac{|g|}{8}$, then i is set to 0 and w is incremented by 1—the latter operation is called wrapping and w represents the number of wraps performed during the mapping.
3. If w exceeds a predefined threshold n_w, the mapping is aborted, i.e., a null phenotype is returned which will be associated to the worst possible fitness.
4. If p contains at least one non-terminal to be expanded, return to step 1, otherwise end.

The search engine of GE, i.e., the way in which the population of individuals is updated across subsequent generations, is conventionally based on Genetic Algorithms (GA). In Section 4.2 we provide the evolutionary parameters values which we used in our experimentation.

In order to adapt the general-purpose GE framework to a specific case, one has to provide a grammar, which implicitly define the phenotype space, and a *fitness function* f, which maps a phenotype to a number in $\mathbb{R}$. In our case, phenotypes are sets of rules and hence we modified the grammar of Figure 2, which describes the language for defining a single rule r, by replacing the first rule in order to make it defining a rule set R, as follows:

$$R ::= \langle \text{conditions} \rangle \mid \langle \text{conditions} \rangle \wedge \langle \text{conditions} \rangle$$

Rules within a set or rules are separated by a conjunction $\wedge$.

Concerning the fitness function, we aimed at defining a function which captures two desired high-level objectives of a set R of road traffic rules: (i) traffic flow regulated by R should allow car drivers to reach their destination without taking too long time, i.e., with a large average speed, and (ii) traffic flow regulated by R should result in no or few collisions. It can be noted that, in principle, the two objectives are conflicting: for instance (and simplifying), a set R imposing a very low speed limit will likely prevent many collisions, but will cause long traveling times for all drivers; on the other hand, a set R not imposing any speed limit will allow drivers to quickly reach their destination, but will likely result in many collisions.

We implemented these high-level objectives by means of two indexes which can be measured for each single driver. The *average speed ratio* (ASR) is the ratio between the actual average speed $\frac{d_{\text{tot}}}{k_{\text{tot}}}$ a driver traveled at and the maximum theoretical average speed $v_{\max}$. The *collision-per-time* (CpT) is the ratio between the number $n_{\text{collision}}$ of collisions a car had during its travel and k_{tot}, where $n_{\text{collision}} \in \{0, 1\}$. For the former index, the greater, the better; the opposite for the latter. Hence, instead of ASR, we considered $1 - \text{ASR}$, i.e., $\left(1 - \frac{d_{\text{tot}}}{k_{\text{tot}}} \frac{1}{v_{\max}}\right)$. We associate a rule set R with a fitness value which is a linear combination of the two indexes averaged across all cars n_{car} during a number n_{sim} of simulations:

$$f(R) = \alpha_{\text{time}} \frac{1}{n_{\text{sim}}} \frac{1}{n_{\text{car}}} \sum_{\text{cars}} \left(1 - \frac{d_{\text{tot}}}{k_{\text{tot}}} \frac{1}{v_{\max}}\right) + \alpha_{\text{collision}} \frac{1}{n_{\text{sim}}} \frac{1}{n_{\text{car}}} \sum_{\text{cars}} \frac{n_{\text{collision}}}{k_{\text{tot}}} \tag{1}$$

where α_{time} and $\alpha_{\text{collision}}$ are the weights.

4 Experiments

We performed two experimental campaigns. The first one aimed at validating our road traffic model. The second one aimed at verifying the effectiveness of our GE-based approach for the synthesis of road traffic rules.

4.1 Validation of the road traffic model

We performed a number of simulations in order to validate our proposed model for the road traffic scenario. In particular, we were interested in: (i) finding

appropriate values for the model ($d_{\text{collision}}$, d_{view}, d_ϵ, $v_{\max}$, v_Δ, k_{advance}, and all the values concerning the road graph) and simulation ($d^{(0)}$, n_{car}, k_{dead}, and $k_{\max}$) parameters; (ii) verifying if the model behaves consistently when varying the number of cars in the road graph; (iii) verifying if a set of manually crafted rules causes a sound modification of the model behavior w.r.t. the absence of rules. To this end, after an exploratory experimentation, we set the values of the parameters as shown in Table 3. In order to simulate different drivers, we set different values, chosen randomly according to a uniform distribution, for the driver-related parameter $v_{\max}$.

Table 3. Model and simulation parameters.

Param.	Meaning	Value
$d_{\text{collision}}$	Minimum distance between cars without collision	1
k_{removal}	Time steps before collided cars remotion	100
d_{view}	Driver's view distance	30
d_ϵ	Driver's safety distance	10
$v_{\max}$	Driver's maximum speed	$\sim U(1.5, 3.5)$
v_Δ	Driver's acceleration (deceleration)	0.1
k_{advance}	Driver's rules forethought time steps	10
$d^{(0)}$	Driver's distance to travel (i.e., initial d)	2000
$\lvert S\rvert$	Number of road sections	5
$\lvert I\rvert$	Number of road intersections	4
$w(p), p \in S$	Number of lanes	$\in \{2, 3, 4\}$
$l(p), p \in S$	Section length	$\in \{100, 100\sqrt{2}\}$
n_{car}	Cars in the simulation	$\in \{2, 5, 8, 11, 14, 17, 20\}$
k_{dead}	Dead car removal time steps	100
$k_{\max}$	Simulation time steps	5000

We performed each simulation by maintaining constant the number n_{car} of cars in the graph during the whole simulation. To this end, during the simulation, we removed a car and added a new one in a random position whenever at least one of the two following conditions was met: (a) the driver's state variable d (i.e., the distance the driver still wants to travel) became lower or equal to zero, or (b) exactly k_{dead} time steps passed since the car state s switched from alive to dead, i.e., since the step when the car was involved in a collision. Concerning the former condition, we recall that drivers do not have a specific destination; instead, their goal is to travel for a predefined distance.

Table 4 shows the set of 8 rules that we manually crafted in order to resemble a (simplified) typical road traffic regulation. The set contains rules regulating the driver's behavior w.r.t. lane to stay on (i.e., "stay on the right", rules 1 and 2), rules stating how the driver's should approach intersection (3, 4, and 5), rules imposing a safety distance (6 and 7), and a rule prohibiting speeding (8). We adjusted the numeric values in the rules by exploratory experimentation, with the aim of reducing the number of collisions while not heavily affecting ASR.

Table 4. Hand-written rules.

	Rule	Explanation
1	$\hat{y} \leq 0.0\text{E}0 \vee \Delta x_{-1} \leq 2.0\text{E}1$	Stay on the rightmost free lane.
2	$\hat{y} \leq 1.0\text{E}0$	Stay on the first or second rightmost lane.
3	$\neg\hat{x} \leq 3.0\text{E}1 \vee \hat{v}_x \leq 1.5\text{E}0$	When close to end of (inter)section, proceed slowly.
4	$\neg\hat{x} \leq 2.0\text{E}1 \vee \hat{v}_x \leq 0.5\text{E}0$	When closer to end of (inter)section, proceed more slowly.
5	$\neg\hat{x} \leq 3.0\text{E}1 \vee \hat{y} \leq 0.0\text{E}0$	When close to end of (inter)section, stay on the rightmost lane.
6	$\neg\Delta x_0 \leq 2.0\text{E}1$	Do not travel too close to the preceding car.
7	$\neg\hat{v}_x \leq 0 \vee \Delta x_0 \leq 2.0\text{E}1$	When too close to the preceding car, stop.
8	$\hat{v}_x \leq 2.4\text{E}0$	Do not exceed a maximum speed.

Figure 3 shows the results of our first experimentation (along with the results of the rules inference experimentation, discussed in the next section). The figure shows the value of the two indexes ($1 - \text{ASR}$, left, and CpT, right) vs. the number n_{car} in the simulation, that is, vs. the injected traffic. There is one curve for each of the three following sets of rules: an empty set (no rules), the set of rules of Table 4 (hand-written rules), and a set of generated rules (best GE rules)—we here discuss the results corresponding to the first two curves. Each point of the curve is obtained by averaging the values of the index collected in 10 simulations.

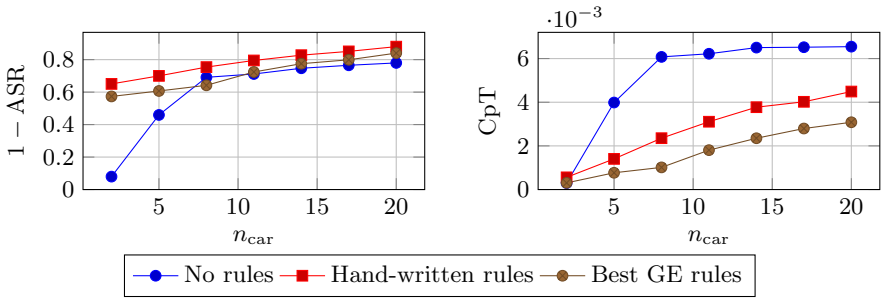


Fig. 3. Values of the two indexes vs. the number n_{car} of cars in the simulation, obtained by performing 10 simulation for each value of n_{car} and each of the three set of rules.

Observing the plots of Figure 3, it can be seen that the relation between the amount of traffic (n_{car}) and the two indexes looks sound, i.e., consistent with what happens in real world road traffic: the greater the number of traveling cars, the lower the average speed (i.e., the longer the time to destination) and the greater the overall number of collisions. From another point of view, there is a trade-off between efficiency and safety of the road traffic, like in the real world.

Moreover, by comparing the curves related to no rules and hand-written rules, it can be seen that enforcing a road traffic regulation results in a different point

in the above-mentioned trade off: with the hand-written rules traffic is in general less efficient but safer. This findings suggest that our models for the road graph, the car, and the driver are sufficiently consistent with the real word and hence, in our opinion, adequate to investigate the feasibility of an automatic synthesis of road traffic rules.

4.2 Synthesis of traffic rules

In our second experimental campaign, we investigated the feasibility of the automatic synthesis of traffic rules using GE. To this end, we run 30 different evolutionary searches using the parameters shown in Table 5 and, concerning the fitness function (see Equation 1), setting $n_{\text{car}} = 10$, $n_{\text{sim}} = 10$, $\alpha_{\text{time}} = 1$, and $\alpha_{\text{collision}} = 100$. For the weights α_{time} and $\alpha_{\text{collision}}$, we chose the values according to the results of the experimentation discussed in the previous section and reflecting the intuition that minimizing collisions is more important than maximizing the average speed.

Table 5. GE parameters.

Population	100	Crossover rate	0.9
Generations	100	Crossover operator	Two-points
Initial genotype size	512	Mutation operator	Bit flip with $p_{\text{mut}} = 0.01$
Max wraps	10	Selection	Tournament with size 5

We first analyze extensively the set of rules which obtained the best fitness among the final generations of the 30 runs, which is shown in Table 6 and discussed later. We experimentally verified how this set of best GE rules affected the traffic with different amounts of injected traffic by running 10 simulations for each value of $n_{\text{car}} \in \{2, 5, 8, 11, 14, 17, 20\}$. In other words, since GE rules were generated with $n_{\text{car}} = 10$, we evaluated the generalization ability of our rules synthesis approach. The results are shown in Figures 3 and 4.

Figure 3 shows the values of the two indexes ($1 - \text{ASR}$ and CpT) vs. n_{car} for the three set of rules: no rules, hand-written rules, and best GE rules. It can be seen that the generated rules always obtain better index values w.r.t. hand-written rules, for both $1 - \text{ASR}$ and CpT—the difference being always statistically significant ($p < 0.001$ with the Mann–Whitney U test) with the sole exception of CpT for $n_{\text{car}} = 2$, for which $p < 0.05$. That is, GE rules allow to reduce the number of collisions and increase the average speed.

Figure 4 shows the same results of Figure 3 from another point of view, by plotting the overall number of collisions in the simulation (i.e., $\sum n_{\text{collision}}$, on y-axis) against the overall traveled distance in the simulation (i.e., $\sum d_{\text{tot}}$, on x-axis), averaged across simulations with the same n_{car}, one curve for each set of rules. The figure allows to appreciate the trade-off between traffic efficiency and safety: the larger the overall distance, the larger the overall number of collisions.

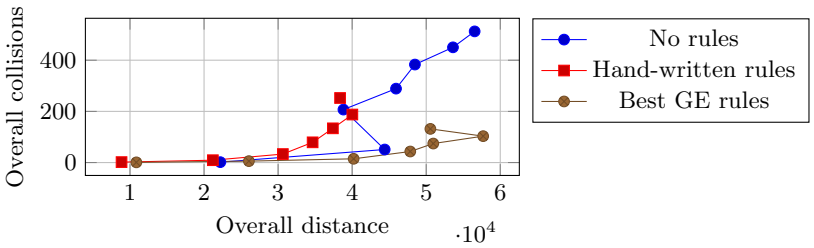


Fig. 4. Overall number of collisions in the simulation against the overall traveled distance in the simulation averaged across simulations with the same n_{car}.

However, it can also be seen that the curve of GE rules strictly dominates both the other two curves (no rules and hand-written rules), hence suggesting that GE rules may be a better way of regulating road traffic regardless of the amount of cars in the graph—i.e., not only for the n_{car} for which the GE rules were generated. This latter finding seems to confirm the good generalization ability of our approach.

Figure 4 also shows another interesting property of our traffic model, namely it highlights the congestion condition. In facts, in both the cases where the traffic is regulated (hand-written and GE rules), there is a maximum number of cars in the graph ($n_{\text{car}} = 17$) after which no further increasing in the overall distance can be obtained, while an increasing in overall number of collisions occurs. Interestingly, despite the fact that the maximum injected traffic before congestion in the two cases is the same, with GE rules the resulting overall distance is greater and the resulting overall number of collisions is smaller.

Table 6 shows in detail the best GE rules: it can be seen that the set consists of four rules, one of which (the 2nd) is clearly always broken in our simulations, since it tries to impose a minimum linear speed which cannot be reached with the parameters shown in Table 3. Rule 4 is easily understandable and its inclusion in best individual is not surprising. The role, if any, of rules 1 and 3 on the drivers' behavior is not clear. Rule 3 is hard to understand, i.e., hard to translate in natural language: the third disjunction says that the remaining part of the rule applies only when the car is in a section (since if $p \in I \equiv p \notin S$, the rule is true); the first and second disjunctions can be rewritten as $\delta v_0 \in \{\emptyset, -1, 0, 1\}$, hence resulting in the rule being writable as $(p \in S \wedge \delta v_0 = \text{opposite}) \implies \Delta x_0 \leq 3$. However, due to the driver's algorithm (see Table 2) and the value of the parameter d_ϵ (see Table 3), it is unlikely that rule 3 plays an actual role in determining drivers' behavior.

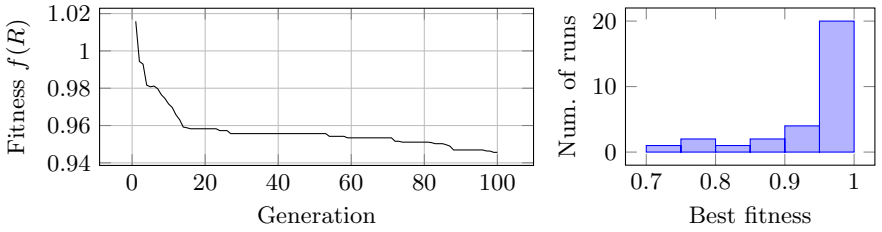
The analysis of the rules of Table 6 may suggest that some mechanism for detecting and cleaning ineffective rules may be beneficial in order to (i) increase the readability/understandability of the generated rules, and (ii) improve the evolutionary search. While we cannot make any meaningful prediction in this respect, we observe that, for the former goal, some automatic and domain-specific heuristic applied after the evolution may suffice—indeed a similar approach have

Table 6. Best GE rules.

	Rule	Possible explanation
1	$\Delta x_{-1} \leq 4.8\text{E}0$	Stay within some distance from the car on right lane.
2	$\neg \hat{v}_x \leq 1.1\text{E}1$	Maintain at least a minimum speed.
3	$\neg \delta v_0 = \text{opposite} \vee \delta v_0 = 1 \vee$ $\neg p \in S \vee \Delta x_0 \leq 0.3\text{E}1$	When in an section, stay close to a car coming on the same lane.
4	$\hat{v}_x \leq 1.2\text{E}0$	Do not exceed a maximum speed.

been applied in [8] in the field of the automatic generation of rules for security policies.

We also analyzed our experimental results by looking at how individuals evolved across the 30 runs. Figure 5 summarizes the results of the evolutionary searches in terms of the fitness of the best individual in the population. In particular, Figure 5a shows how the average (across all runs) best individual fitness varies during the evolution; Figure 5b shows the histogram of the best individual fitness at the end of the evolution.



(a) Average best fitness during evolution. (b) Best fitness histogram.

Fig. 5. Fitness of the best individual: during the evolution, averaged across the 30 runs, on the left; histogram w.r.t. the 30 runs at the end of the evolution, on the right.

It can be seen from Figure 5a that GE is in general able to decrease the fitness of individuals during the evolution: however, after approximately 20 generations, fitness decreases much more slowly. We speculate that some improvement to evolutionary search might be obtained by finely tuning the GE-related parameters, or maybe by using a different variant of GE (e.g., [9, 7]). However, we also note that our scenario is characterized by a remarkably high stochasticity which could, at least, slow down the search of a good solution and, from a more practical point of view, makes experimentation and parameter tuning costly due to the long time needed to compute the fitness. In our case, running a single simulation took $\approx 10\,\text{s}$ on commodity hardware and computing the fitness of a single individual (i.e., a set of rules) consisted in running 10 simulations, precisely to mitigate the impact of the high stochasticity in the simulations.

Figure 5b shows the histogram of the best individual fitness at the end of the evolution across the 30 runs. It can be seen that most of the runs resulted in fitness values close to 1, i.e., the distribution is skewed toward bad values. We analyzed the details of the runs and found that in some of those cases the search got stuck in a local minimum corresponding to a set of rules including one or more rules which, in practice, enforce drivers to stand still. Under those rules, no collision occurs ($\mathrm{CpT} = 0$) and no one moves ($\mathrm{ASR} = 0$), which is, clearly, one extreme of the trade-off between traffic efficiency and safety.

5 Concluding remarks and future work

We proposed and assessed experimentally, by an extensive set of simulations, a method for synthesizing automatically a set of road traffic rules with the aim of maximizing such global indexes as road efficiency (high average speed) and safety (low number of collisions). We are motivated by a future scenario in which human and machine-based drivers will coexist on the roads, making current road regulation possibly unsuitable or inefficient. We are not aware of any similar proposal.

Our method is based on GE: individuals are sets of rules written according to a context-free grammar that we designed ad hoc to express quite expressive concepts such as, e.g., "stay on the rightmost free lane" or "slow down when approaching an intersection". The fitness of a candidate set of rules is given by a weighted sum of the traffic efficiency and safety resulting from the application of the rules set in a number of simulations, which makes this GE application highly stochastic.

Results of our experimental evaluation are promising, since generated rules result in simulated road traffic which is more efficient and safer than that regulated by hand-written rules or not regulated at all.

Our work may be extended in different ways, such as: (i) including a more fine-grained model (e.g., concerning intersections); (ii) considering a different way for expressing rules (e.g., with Linear Temporal Logic); (iii) better exploring GE parameters and/or variants. We plan to investigate some of these research lines in future works.

Acknowledgements

The authors are grateful to Lorenzo Castelli for his insightful comments.

References

1. Bartoli, A., De Lorenzo, A., Medvet, E., Tarlao, F.: Syntactical similarity learning by means of grammatical evolution. In: Handl, J., Hart, E., Lewis, P.R., López-Ibáñez, M., Ochoa, G., Paechter, B. (eds.) Parallel Problem Solving from Nature – PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings. pp. 260–269. Springer International Publishing, Cham (2016), http://dx.doi.org/10.1007/978-3-319-45823-6_24

2. Brabazon, A., O'Neill, M.: Evolving technical trading rules for spot foreign-exchange markets using grammatical evolution. Computational Management Science 1(3-4), 311–327 (2004)
3. Drake, J.H., Kililis, N., Özcan, E.: Generation of vns components with grammatical evolution for vehicle routing. In: European Conference on Genetic Programming. pp. 25–36. Springer (2013)
4. Greene, J.D.: Our driverless dilemma. Science 352(6293), 1514–1515 (2016)
5. Kirkpatrick, K.: The moral challenges of driverless cars. Commun. ACM 58(8), 19–20 (Jul 2015), http://doi.acm.org/10.1145/2788477
6. Koza, J.R.: Genetic programming: on the programming of computers by means of natural selection, vol. 1. MIT press (1992)
7. Lourenço, N., Pereira, F.B., Costa, E.: Sge: a structured representation for grammatical evolution. In: International Conference on Artificial Evolution (Evolution Artificielle). pp. 136–148. Springer (2015)
8. Medvet, E., Bartoli, A., Carminati, B., Ferrari, E.: Evolutionary inference of attribute-based access control policies. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) Evolutionary Multi-Criterion Optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, March 29 –April 1, 2015. Proceedings, Part I. pp. 351–365. Springer International Publishing, Cham (2015), http://dx.doi.org/10.1007/978-3-319-15934-8_24
9. O'Neill, M., Brabazon, A., Nicolau, M., Mc Garraghy, S., Keenan, P.: πgrammatical evolution. In: Genetic and Evolutionary Computation Conference. pp. 617–629. Springer (2004)
10. Rizaldi, A., Althoff, M.: Formalising traffic rules for accountability of autonomous vehicles. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems. pp. 1658–1665. IEEE (2015)
11. Ryan, C., Collins, J., O'Neill, M.: Grammatical evolution: Evolving programs for an arbitrary language. In: European Conference on Genetic Programming. pp. 83–96. Springer (1998)
12. Sanchez, J.J., Galan, M., Rubio, E.: Applying a traffic lights evolutionary optimization technique to a real case: "las ramblas" area in santa cruz de tenerife. IEEE Transactions on Evolutionary Computation 12(1), 25–40 (Feb 2008)
13. Tachet, R., Santi, P., Sobolevsky, S., Reyes-Castro, L.I., Frazzoli, E., Helbing, D., Ratti, C.: Revisiting street intersections using slot-based systems. PLoS one 11(3), e0149607 (2016)
14. Tumova, J., Hall, G.C., Karaman, S., Frazzoli, E., Rus, D.: Least-violating control strategy synthesis with safety rules. In: Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control. pp. 1–10. HSCC '13, ACM, New York, NY, USA (2013), http://doi.acm.org/10.1145/2461328.2461330
15. Vanholme, B., Gruyer, D., Lusetti, B., Glaser, S., Mammar, S.: Highly automated driving on highways based on legal safety. IEEE Transactions on Intelligent Transportation Systems 14(1), 333–347 (2013)
16. Wen, W.: A dynamic and automatic traffic light control expert system for solving the road congestion problem. Expert Systems with Applications 34(4), 2370–2381 (2008)