Vehicular Mobility Model Optimization Using Cooperative Coevolutionary Genetic Algorithms

Sune S. Nielsen
CSC Research Unit,
University of Luxembourg
6, rue Coudenhove-Kalergi
L-1359 Luxembourg
sune.nielsen@uni.lu

Grégoire Danoy
CSC Research Unit,
University of Luxembourg
6, rue Coudenhove-Kalergi
L-1359 Luxembourg
gregoire.danoy@uni.lu

Pascal Bouvry
CSC Research Unit,
University of Luxembourg
6, rue Coudenhove-Kalergi
L-1359 Luxembourg
pascal.bouvry@uni.lu

ABSTRACT

A key factor for accurate vehicular ad hoc networks (VANET) simulation is the quality of its underlying mobility model. VehILux is a recent vehicular mobility model that generates traces using traffic volume counts and real-world map data. This model uses probabilistic attraction points which values require optimization to provide realistic traces. Previous sensitivity analysis and application of genetic algorithms (GAs) on the Luxembourg problem instance have outlined this model's limitations. In this article, we first propose an extension of the model using a higher number of auto-generated attraction points. Then its decomposition on the Luxembourg instance using geographical information is proposed as a way to break epistatic links and hence make its optimization using cooperative coevolutionary genetic algorithms (CCGAs) more efficient. Experimental results demonstrate the significant realism increase brought by both the VehILux model enhancements and the CCGA compared to the generational and cellular GAs.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; D.2.8 [**Software engineering**]: Metrics—*Performance measures*

Keywords

Simulation optimization; Transportation; Coevolution; Genetic algorithms

1. INTRODUCTION

The emergence of *vehicular ad hoc networks* (VANETs) has opened the door to new *intelligent transportation systems* (ITS) that aim at improving driving safety, infotainment and traffic efficiency. Many research challenges have to be faced, such as the development of new protocols, communication algorithms and applications. In order to develop and experiment VANET dedicated solutions, simulation is a must because of technical, cost and reproducibility constraints. Therefore, realistic simulation at both network and mobility levels are required. Previous works indeed demonstrated that the accuracy of the mobility greatly impacts the network connectivity and performance [6]. In this paper we focus on mobility models which aim is to generate traffic traces specifying each vehicles' origin, destination and route between these.

Mobility models can be classified as trace-based, survey-based or simulation-based. In the first class, traces are obtained from GPS data (e.g.[1]). However they do not represent regular traffic as they are limited to commercial vehicles (e.g. taxis or buses) and cannot be modified to generate phenomena like accidents. The second class, survey-based models, permits to generate traces at the macroscopic level, which is of limited interest for VANET applications. Finally the last class, simulator based-traces, focuses on micromobility, i.e. movement is considered at the vehicle level (acceleration, braking, etc.). A recent trend in this field is to combine real-world information such as maps, and statistical data together with microscopic simulation [9, 15].

In this work we focus on the last class, and more precisely on VehILux [11], a simulation-based model that exploits real-world information: geographical map and traffic counts in the considered area. To generate traces from these traffic counts, VehILux relies on a set of probabilistic geographical attraction points to determine destinations. In previous works on the VehILux model for Luxembourg [10], the authors analyzed the model through sensitivity analysis and later optimized VehILux probabilities using three different genetic algorithms (GAs), i.e. a generational GA, a steady-state GA and a cellular GA in [14]. These permitted to outline some limitations of the original VehILux model and previous GAs application: (1) the strong interdependencies between the model parameters, (2) the limited accuracy implied by the low number of attraction areas and (3) the lack of problem knowledge exploitation in the GAs.

This article therefore proposes an extension to the VehILux mobility model, through a more fine grained control over attraction areas of the map. In addition a decomposition of the modified model is defined, in which areas are geographically separated into so-called slices. The performance of a *cooperative coevolutionary GA* (CCGA) exploit-

GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.

ing this decomposition is then compared to the GAs used in the previous work.

The remainder of this article is organized as follows. The next section presents the works related to VehILux usage, analysis and optimization. Then the original VehILux model and the proposed modifications are described in section 3. Section 4 presents the evolutionary approach used for optimizing VehILux original and modified models. Sections 5 and 6 respectively describe the experimental setup and contains the analysis of the obtained results. Finally section 7 contains our conclusions and perspectives.

2. RELATED WORKS

The work presented in this article relies on some previous vehicular mobility model development, analysis and optimization, and more precisely on the *VehILux* model [11].

VehILux is a survey-based mobility model that exploits two real-world data sources i.e. a geographical map and traffic counts from the considered geographical area, with the advantage being that such data is widely available and complete (e.g. [3]). In order to obtain mobility traces from this static data, origins, destinations and routes connecting them are generated using the concept of *repulsion* and *attraction* points. The former constitute typical origin points on the map, while the latter represent popular final destinations. The different attraction points are selected using a probabilistic model, i.e. each of them is assigned with a probability of being chosen as a destination.

These probabilities highly influence the accuracy, and therefore realism, of VehILux. In order to quantify this phenomenon, a sensitivity analysis of the model parameters was conducted using FAST99 and Morris approaches in [10]. On the considered instance, i.e. Luxembourg, it was shown that one of the parameters, the inner traffic ratio, has a predominant impact on the fitness, and that all parameters in general have strong interactions. This means that first order effects of parameters are relatively small compared to their combined effect in the model. Furthermore it was shown that the effect of parameter changes is subject to large amount of noise and that no obvious conclusions could be made on 'good' probability ranges, except for one (the inner traffic ratio).

Finding the best set of probabilities of VehILux for a given problem instance (i.e. a geographical area and its corresponding traffic counts) is a hard task and has been modeled as an optimization problem in [14]. The performance of three GAs - generational, steady-state and cellular - was compared on the same Luxembourg instance. The sensitivity analysis results were used to reduce the range of possible values for some variables (i.e. the inner traffic ratio). The results obtained with these metaheuristics have demonstrated the limitations of the original VehILux model. Indeed, all algorithms converged to some low quality solutions, i.e. the delta between the real counts and the counts obtained from the generated traces remains high.

Finally, in [8] the authors proposed to further enhance the realism of VehILux traces with of a new route assignment algorithm, i.e. the Gawron's algorithm [7]. To this end, the best VehILux traces previously obtained using GAs have been used in the SUMO microscopic traffic simulator [5]. This permitted to outline another limitation of the VehIlux model, as in realistic road conditions the simulator is unable to reproduce all the theoretical traffic due to road network congestion. One of the reasons is the limited number of attraction areas of VehILux that constrains the generation of traces to a limited set of routes.

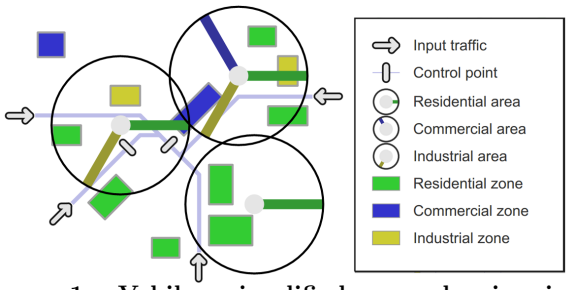

Figure 1: Vehilux simplified map showing input flows, control points, areas and zones.

The objective of this work is therefore to improve VehILux accuracy by proposing a new route assignment model that uses a larger number of attraction areas and further optimize its parameters through the exploitation of its decomposition capabilities with cooperative coevolutionary GAs.

3. PROBLEM DESCRIPTION

This section describes the problem of flow generation in general and how it was solved in the original VehILux work. Then the improvements of the current work to the model itself and how it is fitted specifically to a coevolutionary algorithm is described.

3.1 VehILux Original Model

The VehILux mobility model relies solely on traffic volume counts and geographical map data. The instance considered in this paper consists of a geographical map of Luxembourg city with its surrounding region as well as real traffic volume counts collected every hour in 28 locations within the region.

Entering points are selected from the 15 locations positioned at the edge of the map, as the problem instance addressed is concerned with commuting hours. These points and their volume counts are used as to initiate the flow of vehicles into the model, referred to as *outer traffic*, while the remaining 13 count locations positioned towards the center of the region are used to validate the quality of the generated traffic flows. In addition to the outer traffic, a certain portion of the traffic is generated inside the map, referred to as *inner traffic*.

From the geographical map data, information about the different zones in the region can be derived, and is used to predict the destinations of the vehicles. This is done by making a destination distribution model for the vehicles by assigning to each zone a probability of being chosen as a destination for a vehicle.

Route assignment concerns the selection of a destination zone for each origin and the selection of a route between the origin-destination pair. The selection of a destination is based on the notion of *zone types* and *attractivity areas* where possible types are *commercial*, *residential* and *industrial*. Each zone type is assigned with an overall global probability of being selected as a destination type, noted P_T where $T \in \{R, C, I\}$ is the zone type. However, zones of the same type can exist in several locations spread over the map where some locations are more popular than others and should hence be associated with a higher selection probabil-

ity. This property is modeled by the concept of *attractivity areas*, which can cover any smaller or bigger part of the map and are defined independently for each zone-type.

Every zone location is associated with a specifically defined attractivity area, and if not, at least with a default attractivity area.

In Fig. 1 the concept is shown in a simplified example. The top right circular attractivity area contains zones of all three types, as shown by the three wheel spokes which denotes three overlapping areas with the same center and radius. Top left, a commercial zone is shown that is not covered by any specific area, therefore this zone will belong to the default commercial area of the map.

We define $P()$ and $S()$ as probability- and surface-functions respectively, and the probability $P(z)$ of the selected zone z becomes:

$$P(z) = P_T \times P(z.a) \times \frac{S(z)}{S(z.a)}, \quad (1)$$

where $P(z.a)$ and $S(z.a)$ are respectively the probability and surface of the attractivity area $z.a$ to which the zone z was assigned. These probabilities are essentially what we have to tune in order to generate realistic flows with the model.

As mentioned, a fraction of the flow is also generated inside the map. This generation is controlled by the parameter *inner traffic ratio* which is the probability of a trace having its origin located in a random residential zone within the map instead of in an input flow location on the edge of the map. When a destination has been chosen for a given origin, the route between the origin-destination pair is generated using Dijkstra's shortest path algorithm. Speed limits and traffic lights on roads are used to adjust the weight of each road in the generation process.

3.2 Model Modifications

To further enhance the quality of the produced flow, a number of modifications have been implemented and later tested.

3.2.1 Time-frame Reduction

The originally proposed VehILux model instance tackled the entire interval from midnight and 11 one-hour time slots ahead. An attempt to better match the real-life traffic was to look only at the traffic flow of three time-slots, covering peak-commuting hours from 6AM to 9AM. This is motivated by the idea that achieving optimal settings in different time slots would require different model parameters for each time slot. Another motivation is that the simulation is quite CPU-intensive and proportional to the number of vehicles injected into the model. Therefore by reducing the time-frame to 3 hours, we can optimize the model in a little more than one third of the time. The two instances of the model are called the *3-hour*- and *11-hour*-instances respectively.

3.2.2 Additional attractivity areas

The original VehILux instance considered fewer rather large attractivity areas situated around and the center of the map almost covering the city-center. In an attempt to enhance the model performance, we replace the original attractivity areas by many smaller areas placed systematically to cover the city-center as well. This is done by placing areas of the three types, residential commercial and industrial on the map in a grid covering the central region. When doing this an area is only placed if there is actually a zone of the same type within its circumference. This can be seen in Fig. 2, which represents the real Luxembourg instance tackled in our experiments. Some attractivity areas have up to three radial spokes representing the types of zones within the area, and on the other hand blank positions are found where there are no zones. If a zone is situated within the circumference of two or more overlapping attractivity areas, it is simply assigned to the first placed area.

This modification alone should in theory provide grounds for generating much better flows closer to real-life measurements with the drawback being an important increase of model variables to tune. Our expectations are that a coevolutionary algorithm will cope well with this large amount of variables.

3.2.3 Geographical Model Decomposition

In addition to generating attractivity areas systematically, we also group them depending on the geographical location on the map. In Fig. 2 this group assignment is shown by the numeric labels in the center of areas. We group attractivity areas by slicing the map, and the three slice borders or cuts are also seen in the figure.

We expect that optimising an attractivity area probability will have larger impact on the generated flow within its slice, and hence control points in the slice, rather than in other slices.

The main reason for this separation into groups is that eventually, each group of areas will be optimized in their own separate sub-population of a coevolutionary genetic algorithm, as described by Potter in [12]. In [13] Potter describes the ideal case where a problem can be decomposed into independent subcomponents. Here each sub-component could be solved independently without regards to the others. On the contrary when changing interdependent subcomponents it can be described as a "deforming" or "warping" of the fitness landscapes associated with each of the other subcomponents, which is to be avoided.

In addition to grouping the auto-generated attractivity areas by their slice location, the *default* attractivity areas are also divided by slice. The original VehILux model instance had three default areas, one per zone-type essentially containing all zones around the city center. To achieve as much independence between the slices as possible, these areas are also split into a total of 9 areas: three default areas per slice which are then optimized by the sub-population assigned to the slice.

4. METHODOLOGY

In this work, we compare the performance of three different GAs that rely on different population structures (panmictic, cellular and islands) in order to optimize VehILux original and modified models. We present in sections 4.2 to 4.3 the three algorithms used: generational GA, cellular GA and coevolutionary GA. Then the objective function is defined in 4.4, followed by the solution encoding an normalization in 4.5 and 4.6 respectively.

4.1 Generational GA

Generational GAs (genGAs) are a type of panmictic algorithm, i.e. individuals are grouped into a single structure-

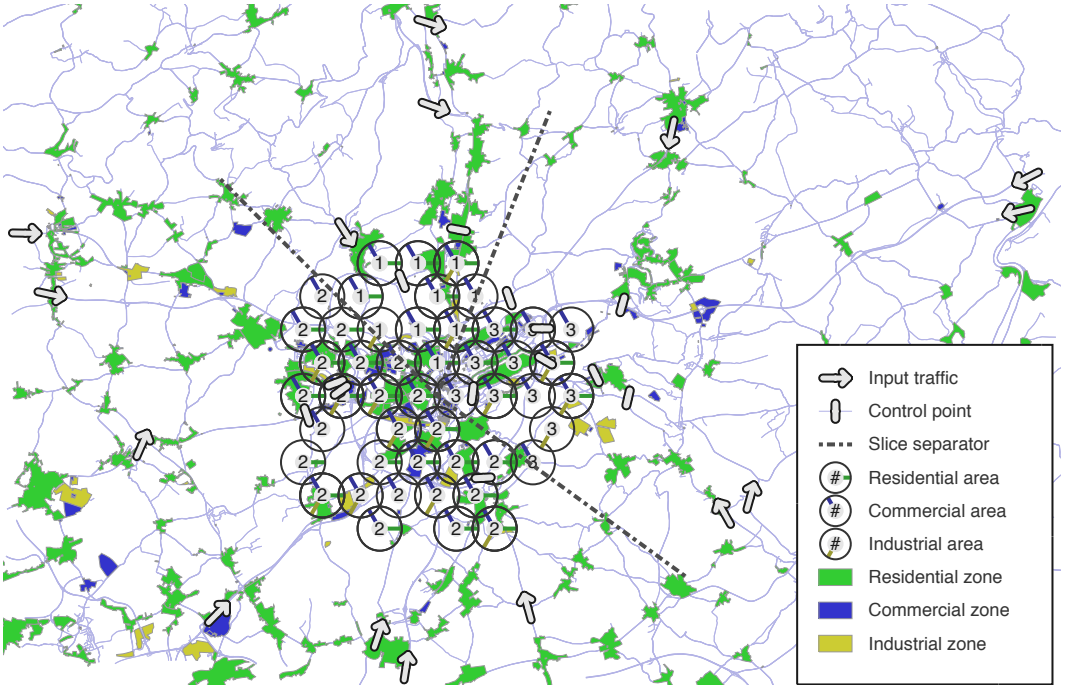

Figure 2: Real Vehilux map of Luxembourg with 102 auto-generated attractivity areas and its decomposition.

Algorithm 1 Pseudo-code of a canonical genGA

```
1: proc Evolve(genga)      // Parameters of the algorithm in 'genga'
2: GenerateInitialPopulation(genga.pop);
3: Evaluation(genga.pop);
4: while ! StopCondition() do
5:    for i←0 to genga.popSize do
6:       parents ← Selection(genga.pop);
7:       offspring ← Recombination(genga.Pc,parents);
8:       offspring ← Mutation(genga.Pm,offspring);
9:       Evaluation(offspring);
10:      Add(auxiliary_pop,offspring);
11:   end for
12:   genga.pop ← auxiliary_pop;
13: end while
14: end proc Evolve
```

Algorithm 2 Pseudocode for a canonical cGA

```
1: proc Evolve(cga)      //Algorithm parameters in 'cga'
2: while ! StopCondition() do
3:    for individual ← 1 to cga.popSize do
4:       n_list←Get_Neighborhood(cga,position(individual));
5:       parents←Selection(n_list);
6:       offspring←Recombination(cga.Pc,parents);
7:       offspring←Mutation(cga.Pm,offspring);
8:       Evaluation(offspring);
9:       Add(position(individual),offspring,cga);
10:   end for
11: end while
12: end proc Steps_Up;
```

less population also referred to as panmixia. Individuals can thus mate with any other individual in the population.

The genGA is a (μ, λ)-GA, where the newly generated individuals are placed in an auxiliary population which will replace the current population when it is completely filled, i.e., when the number of newly generated solutions is equal to the size of this auxiliary population. In our case, the sizes of both the auxiliary and the current population is the same ($\mu = \lambda$).

Algorithm 1 presents the pseudo-code of the generational GA. The population is first randomly initialized (line 2). Each generated individual is then evaluated using the fitness function defined for the tackled problem (line 3). The genetic loop then starts in line 4 until some predefined termination condition is met, e.g. a number of fitness function evaluations. "Parent" individuals are selected using some stochastic selection operator to construct the mating pool (line 6). Genetic variation is then ensured by the crossover (also called recombination) and mutation operators, both applied with some probability, which permit to visit other search space regions (line 7 and 8). The obtained offspring is then evaluated and inserted into the auxiliary population (line 9 and 10). The offspring population will become the current one once full (line 12).

4.2 Cellular Genetic Algorithm

Cellular genetic algorithms (cGAs) [4] are a kind of GA with a structured population in which individuals are spread in a two dimensional toroidal mesh, and they are only allowed to interact with their neighbors.

A canonical cGA follows the pseudo-code included in Algorithm 2. In this basic cGA, the population is usually structured in a regular grid of d dimensions ($d = 1, 2, 3$), and a neighborhood is defined on it. The algorithm iteratively considers as current each individual in the grid (line 3), and individuals may only interact with individuals belonging to their neighborhood (line 4), so parents are chosen among the neighbors (line 5) with a given criterion. Crossover and mutation operators are applied to the individuals in lines 6 and 7, with probabilities P_c and P_m, respectively. Afterwards, the algorithm computes the fitness value of the new offspring individual (or individuals) (line 8), and inserts it (or one of them) instead of the current individual in the population

Algorithm 3 Pseudocode of the CCGA

```
1: gen = 0
2: for all species_s do
3:    Pop_s(gen) = randomly initialized population
4:    evaluate fitness of each individual in Pop_s(gen)
5: end for
6: while termination condition = false do
7:    gen = gen + 1
8:    for all species_s do
9:       select Pop_s(gen) from Pop_s(gen − 1) based on fitness
10:      apply genetic operators to Pop_s(gen)
11:      evaluate fitness of each individual in Pop_s(gen)
12:   end for
13: end while
```

(line 9) following a given replacement policy. This loop is repeated until a termination condition is met (line 2).

4.3 Cooperative Coevolutionary GA

In addition to the cellular model, another common way for structuring the population of GAs consists of splitting the whole population into several subpopulations. Each subpopulation independently evolves a GA, and exchange some information among them during the run. We study in this paper an algorithm following this model, namely CCGA, a cooperative coevolutionary GA. Instead of considering a population of similar individuals representing a global solution, like classical genetic algorithms, CCGA considers the coevolution of subpopulations of individuals representing specific parts of the global solution.

The CCGA considered here is based on the model proposed by Potter and DeJong [12], in which a number of subpopulations explore different decompositions of the problem. CCGA involves several species independently evolving with a GA a subset of the global solution vector. In order to evaluate complete solutions on the global problem, all subpopulations need to cooperate by exchanging representatives of their respective subpopulations.

A pseudocode of CCGA is provided in Algorithm 3). In the initial generation *(t=0)* individuals from a given subpopulation are matched with randomly chosen individuals from all other subpopulations. A fitness for each individual is evaluated, and the best individual in each subpopulation is found. The process of *cooperative coevolution* starts form the next generation *(t=1)*. For this purpose, in each generation a cycle of operations is repeated in a round-robin fashion. Only one current subpopulation is active in a cycle, while the other subpopulations are frozen. All individuals from the active subpopulation are matched with the best values of frozen subpopulations. When the evolutionary process is completed, a composition of the best individuals from each subpopulation represents a solution of a problem.

Two ways of evaluating the fitness of an individual, referred to as credit assignment 1 and 2 (CCGA-1, CCGA-2), are considered in this work. CCGA-1 evaluates an individual with the best individual of the other subpopulations, as presented in Algorithm 3. CCGA-2 additionally evaluates the individual with a random individual from each other subpopulation and the best fitness among the two obtained is kept. In this work however, the CCGAs are not frozen in turn, but evolve in parallel and exchange individuals at the beginning of each generation.

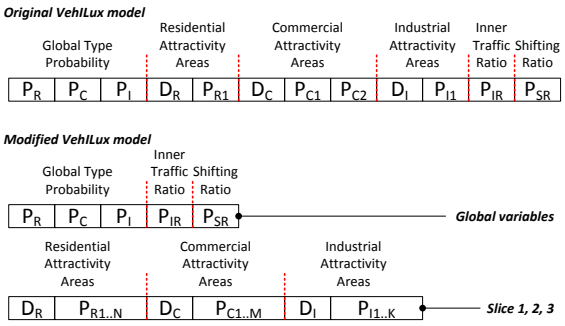

Figure 3: Individual encoding of original and modified model.

4.4 Objective function

As mentioned, the 13 of the 28 real-life traffic count locations positioned closer to the center of the region are reserved to validate the flow generated by the model. The quality of this flow is evaluated by comparing the generated traffic counts with real traffic volume counts at the control points.

The evaluation or fitness function F is the following:

$$F = \sum_{c=1}^{C} \sum_{t=1}^{T} |r_c(t) - c_c(t)|\,, \tag{2}$$

where $r_c(t)$ is the real traffic volume count at control point c in time slot t, $c_c(t)$ is the number of vehicles at control point c derived from the generated traffic flows in time slot t, C is the number of control points and T is the number of time slots. The smaller this sum of absolute differences between the real traffic volume counts and the estimated ones is, the better the model estimated the real flow.

Traffic flows are generated independently for each one hour time slot t and control point c, i.e. $c_c(t)$, as real traffic count data are typically collected on per hour basis. However, there is no precise departure time related to each origin-destination pair, except the 60 minutes slots (e.g. vehicle i departs from point a to point b between 9 and 10AM). Such a function is proposed in [10]. It assumes that a vehicle passes through a control point at the same hour as it initiates the trip. This is not always true as the vehicle may pass through the control point in the next hour. To include this "delay" effect, a *shifting ratio* parameter is introduced, which defines the ratio of vehicles whose trips are scheduled to start in time slot t, but will pass through the control point in the slot $t+1$. The estimated number of vehicles that pass through control point c within time slot t ($c(t)$) is calculated as follows:

$$c_c(t) = p_c(t) \times (1 - \alpha) + p_c(t-1) \times \alpha, \tag{3}$$

where $p_c(t)$ is the number of all vehicles generated in time slot t that pass through control point c, $p_c(t-1)$ is the number of vehicles generated in time slot $t-1$ that pass through control point c in time slot t, and α is the shifting ratio.

4.5 Solution Encoding

Fig. 3 shows how the parameters of the model are encoded in the original and modified model respectively. Global zone probabilities are noted P_T where $T \in \{R, C, I\}$ denotes the

zone type. Default attractivity areas probabilities are noted D_T and remaining conventional areas are noted P_{Ti} where i is the index. In the modified version, the global parameters are assigned to the first sub-population and the remaining three sub-populations are responsible for optimizing only attractivity areas within their corresponding slice.

4.6 Solution Normalization

As the destination distribution model works by selecting destinations based on their probability, one important requirement is that the sum of all zone probabilities per type are 1. Therefore the probabilities of a slice are normalized according to the total surface of the zones in that slice compared to total surface of all zones. With $P()$ and $S()$ again being the probability- and surface-function respectively, the probability $P(a)$ of each area a in a slice $\mathcal{S}$ of the map $\mathcal{M}$ is normalized as follows:

$$\forall a \in \mathcal{S}, P'(a) = \frac{P(a)}{P(\mathcal{S})} \times \frac{S(\mathcal{S})}{S(\mathcal{M})}. \quad (4)$$

5. EXPERIMENTAL SETUP

This section presents the parameters of the genetic algorithms and problem instances used in our experiments. These are summarized in Table 1 .

5.1 Algorithms Parameters

Three different GAs have been used to tackle the optimization of the VehILux mobility model, a generational GA, a cellular GA and two coevolutionary GAs (CCGA-1 and CCGA-2). The genGA uses a single population of 100 individuals, the cGA a 10×10 population and the CCGAs 4 subpopulations of 25 individuals each. For all algorithms, the termination condition was limited to 8000 fitness function evaluations, due to the computationally heavy fitness evaluation. The recombination operator (the single point crossover—SPX) was used with the probability $p_c = 1.0$. The uniform mutation operator was applied with the probability $p_m = \frac{1}{chrom\_length}$. The binary tournament was used for the selection of the two parents except for the cGA where one parent is the central individual. Other cGA specific parameters are the neighborhood, we used C9 (9 closest individuals measured in Manhattan distance), and the replacement strategy. Finally, all the obtained results are averaged over 30 independent runs.

5.2 Problem Instances Parameters

In order to evaluate the VehILux mobility model, simulations have been based on a detailed map of Luxembourg from OpenStreetMap [2]. The considered area surface is 1700 square kilometers (47×36 kilometers). Traffic volume counts from 28 locations were obtained from [3]. 15 locations were selected as entering points and 13 as control points (see Fig. 2). For each control point, values corresponding to each hour were taken. Two different instances have been tackled, the 12 hours that was used in [14] to optimize the original VehILux model, and a new smaller instance of only 3 hours. During this period, i.e. morning hours, most of the travel involves commuting, therefore, the entering points are selected from the locations positioned at the edges of the map. Three zone types are defined: *residential*, *commercial* and *industrial*. Information about the number, position and surface of these zones was extracted from the OpenStreetMap.

Table 1: Algorithms and VehILux parameters

Genetic Algorithms	Population size	100 (genGA) 10 × 10 (cGA) 4 x 25 (CCGA)
	Termination Condition	8000 function evaluations
	Number of runs	30
	Selection	Binary tournament (BT) BT + Center individual (cGA)
	Neighborhood	C9 (cGA)
	Replacement Strategy	replace if better (cGA)
	Crossover operator	SPX, p_c=1.0
	Mutation operator	Uniform, $p_m = \frac{1}{chrom\_length}$
VehILux	Simulation area	1700 km^2 (47 × 36 km)
	Number of entering points	15
	Number of control points	13
	Simulated time periods	12AM–11AM (11 time slots) 6AM–9AM (3 time slots)

6. NUMERICAL RESULTS

In this section we present the results of our experiments and demonstrate the performance increase obtained thanks to the proposed modifications of the model. The results are discussed and compared for four different algorithms: GenGA, cGA, CCGA-1 and CCGA-2. We included CCGA-2 because of its increased explorative characteristics, in order to estimate its performance on the more challenging 11−hour instances. For each experiment with CCGA-1 and CCGA-2 on the modified model, we included experiments where the attractivity areas to sub-populations are randomly assigned (CCGA-1 shfl and CCGA-2 shfl). The objective is to evaluate the advantage of the problem decomposition based on geographical information.

Experimental results are summarized in Tables 2 and 3 showing best and average fitness with standard deviation for all algorithms. Additionally, for each problem instance, the overall best result and best average fitness among the algorithms is shown in bold font. Statistical confidence in our comparisons is assessed by performing the Wilcoxon test [16]. Finally convergence plots are presented in Fig. 4 to 7.

6.1 The 11-hour Instance

We start out by showing the results of the 11−hour instance, with convergence plots in Fig. 4 and 5 which is also the scenario used in previous works. A first straightforward conclusion is that the modified model is significantly better: in general almost 4000 points with a standard deviation 20 times less - regardless of the algorithm used.

When looking at the original model with 12 alleles we notice that the best result and best convergence is achieved with the CCGA-1, but that the best average is achieved with a regular GenGA. However, based on the Wilcoxon test, statistical confidence in this difference is only true between GenGA and CCGA-2. The latter being also significantly worse than the CCGA-1.

For the 107 alleles in the modified model, the coevolutionary algorithms seem to have an advantage in general, here Wilcoxon indicates that the CCGAs are both significantly better than all other GAs. The CCGA-1 algorithm is again superior in convergence and final result, but the explorative features of CCGA-2 almost allows it to catch up towards the 8000th evaluation.

Table 2: Summary of results for the 11-hour model

	Algorithm	*Best fitness*	*Avg. fitness*
Modified model	GenGA	10602	$10803.000_{\pm 129.813}$
	cGA	10426	$10838.367_{\pm 157.585}$
	CCGA-1	**10320**	$\mathbf{10716.533}_{\pm 280.271}$
	CCGA-2	10419	$10722.333_{\pm 251.953}$
	CCGA-1 shfl	10415	$11024.667_{\pm 429.214}$
	CCGA-2 shfl	10558	$11072.933_{\pm 332.150}$
Org. model	GenGA	14330	$\mathbf{14427.667}_{\pm 55.079}$
	cGA	14326	$14441.500_{\pm 76.905}$
	CCGA-1	**14250**	$14431.833_{\pm 106.041}$
	CCGA-2	14312	$14484.833_{\pm 96.905}$

Table 3: Summary of results for the 3-hour model

	Algorithm	*Best fitness*	*Avg. fitness*
Modified model	GenGA	3007	$3272.367_{\pm 136.323}$
	cGA	2961	$3207.500_{\pm 109.123}$
	CCGA-1	**2884**	$\mathbf{3189.700}_{\pm 145.070}$
	CCGA-2	2927	$3227.200_{\pm 138.607}$
	CCGA-1 shfl	3006	$3303.900_{\pm 158.914}$
	CCGA-2 shfl	3007	$3330.200_{\pm 210.293}$
Org. model	GenGA	4869	$4954.633_{\pm 43.615}$
	cGA	4865	$4950.800_{\pm 53.351}$
	CCGA-1	**4848**	$\mathbf{4933.067}_{\pm 86.745}$
	CCGA-2	4864	$4990.733_{\pm 99.371}$

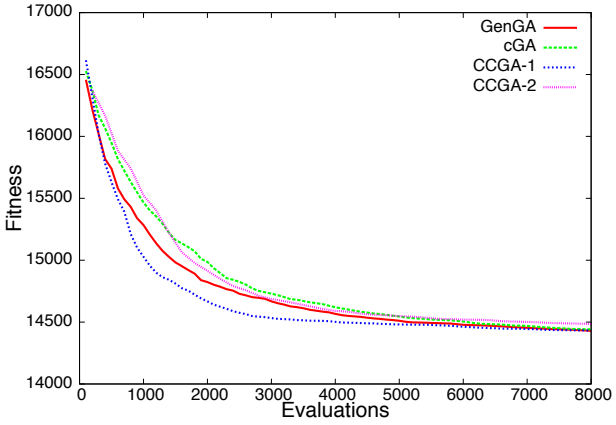

Figure 4: Convergence of the 11−hour original model.

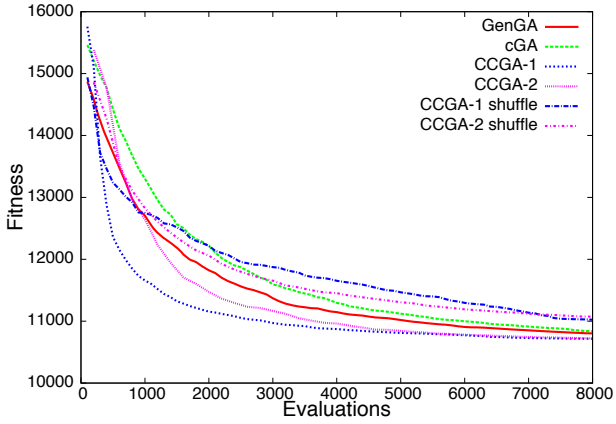

Figure 5: Convergence of the 11−hour modified model.

6.2 The 3-hour Instance

Table 3 shows the same experimental results as in Table 2, for the shorter 3−hour time-frame and with corresponding convergence plots in Fig. 6 and 7. The Wilcoxon tests indicate less difference amongst the algorithms in the modified model, and the only significant conclusion is that CCGA-1 is significantly better than GenGA. CCGA-1 still provides the overall best solution with 2884. We expect this is due to the less complex nature of the problem with a reduced number of time slots.

On the original model however, CCGA-1 is significantly better than all other algorithms, even CCGA-2 according to the Wilcoxon tests. In terms of initial convergence and best results achieved, the CCGA-1 is dominating in both original and standard models with 3− and 11−hour instances.

6.3 Comparing 3h- and 11h- Instances

To compare the best results achieved in the 3− and 11− hour instances better, we calculate the ratio of the best found individuals of the two models to the total flow injected into the model. The total number of vehicles injected for 3− and 11− hour instances respectively are 17342 and 46121. This results in the ratios $16,63\%$ and $22,38\%$ for the best found results of 2884 and 10320. Clearly the shorter time-frame allows the algorithms to fit the flow better.

6.4 Shuffled vs. grouped sub-populations

Another important aspect to study was the effect of grouping attractivity areas and assign them to sub-populations based on their geographic location in the model, i.e. model knowledge based decomposition. For both 3− and 11−hour instances we observe noticeable worse results when the sub-populations are shuffled. Considering the average fitness, the shuffled model represents the worst results for both instances, which is also the case for the best found solutions, except the 3−hour model where it represents the $2nd$ worst. Overall the average performance of the CCGA-1 algorithm becomes 3.58% and 2.88% worse for the 3− and 11−hour shuffled models respectively and Wilcoxon tests indicate that the shuffled CCGA-1 is significantly worse than any non-shuffled version. Further the experiments with shuffled variables, have the highest variation in their final results indicating that some decompositions have been very unfavorable. These observations all underline the advantage of the proposed decomposition of the model into geographically determined slices each optimized by their respective coevolutionary sub-population.

7. CONCLUSIONS AND PERSPECTIVES

In this paper we presented a number of changes to the VehILux mobility model to firstly make the model more suitable for optimization with a Cooperative Coevolutionary Genetic Algorithm (CCGA), secondly to increase the solution quality in general. To reduce epistatic links among coevolving sub-populations, model knowledge is used to decompose the variables and group them according to geographical locations in the model. This effort was shown to have significant impact when comparing the same experiments to a randomized decomposition. By shuffling members of the sub-populations the performance of the CCGA-1 algorithm

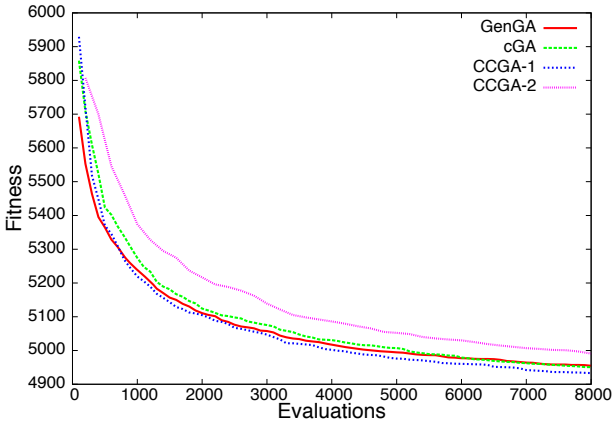

Figure 6: Convergence of the 3−hour original model.

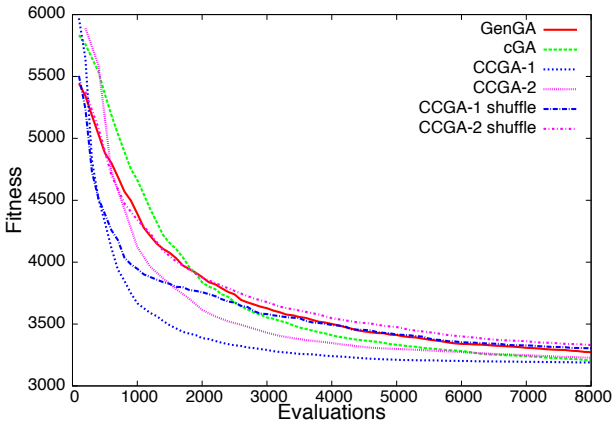

Figure 7: Convergence of the 3− hour modified model.

on average becomes 3.58% and 2.88% worse for the 3− and 11−hour models, with an additional increase in deviation of results indicating bad worst-case performances. As the Wilcoxon tests further confirm that results are statistically significantly better applying the intelligent decomposition by means of model knowledge, we succeeded in reaching the goal set for this work.

By optimizing the model for a shorter time-span, we also reduced the mismatch compared to the total flow from 22, 38% to 16, 63%. The modified model with added attractivity areas was able to decrease the mismatch in flow compared to the original model by 40.5% and 27.6% for the 3− and 11−hour instances respectively.

Future work will be focused on proposing an adaptive control of the mobility models simulation precision from the algorithm's point of view. Finally the newly generated traces will be used in the SUMO microscopic simulator and compared to the original model accuracy.

8. ACKNOWLEDGMENTS

This work has been funded by the UL-EvoPerf project. Sune S. Nielsen acknowledges the support of the National Research Fund of Luxembourg (FNR), with the AFR contract no. 1356145. Experiments presented in this paper were carried out using the HPC facility of the University of Luxembourg.

9. REFERENCES

[1] The cabspotting project. `http://cabspotting.org/`.

[2] OpenStreetMap project. `http://www.openstreetmap.org/`.

[3] Traffic volume counts in Luxembourg, Luxembourg ministry of transportation, [online]. `http://www.pch.public.lu/trafic/comptage/index.html`.

[4] E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*. Operations Research/Compuer Science Interfaces. Springer-Verlag Heidelberg, 2008.

[5] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. SUMO - Simulation of Urban MObility: An Overview. In *Proc. 3rd International Conference on Advances in System Simulation, SIMUL*, pages 63–68, Spain, 2011.

[6] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002.

[7] C. Gawron. An iterative algorithm to determine the dynamic user equilibrium in a traffic simulation model. *International Journal of Modern Physics C*, 9(3):393–407, 1998.

[8] A. Grzybek, G. Danoy, and P. Bouvry. Generation of realistic traces for vehicular mobility simulations. In *Proceedings of the second ACM international symposium on Design and analysis of intelligent vehicular networks and applications*, DIVANet '12, pages 131–138, New York, NY, USA, 2012. ACM.

[9] V. Naumov, R. Baumann, and T. Gross. An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In *7th ACM international symposium on Mobile ad hoc networking and computing*, MobiHoc '06, pages 108–119, 2006.

[10] Y. Pigné, G. Danoy, and P. Bouvry. Sensitivity analysis for a realistic vehicular mobility model. In *ACM international symposium on Design and analysis of intelligent vehicular networks and applications (DIVANet 2011)*, pages 31–38. ACM, 2011.

[11] Y. Pigné, G. Danoy, and P. Bouvry. A vehicular mobility model based on real traffic counting data. In *Proc. 3rd International Workshop on Communication Technologies for Vehicles (Nets4Cars 2011)*, volume 6596, pages 131–142. Springer, LNCS, 2011.

[12] M. Potter and K. De Jong. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature (PPSN III)*, pages 249–257. Springer, 1994.

[13] M. A. Potter. *The design and analysis of a computational model of cooperative coevolution*. PhD thesis, 1997.

[14] M. Seredynski, G. Danoy, M. Tabatabaei, P. Bouvry, and Y. Pigné. Generation of realistic mobility for vanets using genetic algorithms. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

[15] S. Uppoor and M. Fiore. Large-scale urban vehicular mobility for networking research. In O. Altintas, W. Chen, and G. J. Heijenk, editors, *VNC*, pages 62–69. IEEE, 2011.

[16] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.