Small or medium-scale focused research project (STREP)

ICT Call 8

FP7-ICT-2011-8

Cooperative Self-Organizing System for low Carbon Mobility at low Penetration Rates

COLOMBO: Deliverable 2.4

Policies for Pedestrians, Bicycles and Public Transport

Document Information	
Title	Delivarable 2.4 - Policies for Pedestrians, Bicycles and Public Transport
Dissemination Level	PU (Public)
Version	1.0
Date	15.07.2015
Status	Approved Version
Authors	Federico Caselli (UNIBO), Alessio Bonfietti (UNIBO), Michela Milano (UNIBO)

Table of contents

1 Introduction

This deliverable contains the results of Task 2.5 on “System Extensions for Pedestrian, Bicycles, public Transports and Emergency Vehicles” of the project’s Work Package (WP) 2. The goal of this task is to extend the developed system to take into account pedestrian and bicycles management and give precedence to the public transports and emergency vehicles.

The algorithms for traffic light control defined in Tasks 2.1 to 2.3 regard only motorised, four-wheeled vehicles and do not distinguish between different modes of transport these vehicles represent, such as public transport. Within Task 2.5, the traffic light control system has been extended with rules that allow adapting the traffic light’s behaviour to pedestrians and rules for prioritising public transport and emergency vehicles.

Therefore, this deliverable is divided into two main parts. The first describes the extensions to the traffic light logic to take into account pedestrian and bicycles management. The second presents the approach adopted in the controller to give precedence to special vehicles.

1.1 Document Structure

This deliverable is structured into three main chapters.

Chapter 2 briefly describes the structure and the behaviour of the traffic light controller.

Chapter 3 illustrates how the system takes into account the management of pedestrian and bicycles.

Chapter 4 describes the extensions to the developed controller giving precedence to the public transports and emergency vehicles. The last section of this chapter presents the experimental results. These results show that the system correctly gives precedence to special vehicles and that it is robust to low penetration conditions.

Finally, the conclusions are given in Chapter 5.

2 Background

In this section we briefly recall the algorithm developed within the Work Package 2.

2.1 The traffic light control algorithm

The main goal of Work Package 2 it to design a self-organizing traffic light control algorithm that is able to sense the traffic condition and automatically select the policy which best fits the current traffic. The algorithm is structured into two levels: the macroscopic level consists in an algorithm (called "swarm") that, using the information on the current traffic condition, chooses a proper low level policy. The microscopic level is therefore modelled through a set of policies that could take short-term decisions: when the green light should go on and off and on which lanes, the duration of the red period and so on. These decisions are mainly taken depending on two factors: the number of waiting cars and the time these cars are waiting.

The selection of a microscopic policy is probabilistic and depends on the sensed traffic condition. Given the uncertainty in the definition and measurement of "high" vs "low" traffic, we use the natural metaphor of pheromone to abstract the traffic conditions. Each traffic light controller calculates its own three pheromone levels:

- pheromone for the incoming lanes (pheromone_in, or φ_{in})
- pheromone for the outgoing lanes (pheromone_out, or φ_{out})
- dispersion of the pheromone among the approaching lanes (pheromone_dsp, or φ_{dsp})

Each microscopic level policy is mapped in the pheromone space (i.e. $\varphi_{in} \times \varphi_{out} \times \varphi\_dsp$) using a stimulus function. A stimulus function specifies how to compute the current level of stimulus for every policy a traffic light controller is able to execute, with respect to pheromone levels: the more desirable the policy, the higher the stimulus. As stated before, the selection of a microscopic policy is probabilistic: the higher the stimulus function for a policy, the higher the probability that the policy will be selected.

Further details on the model and the implementation can be found in the Deliverables 2.1 to 2.3.

3 Policies for Pedestrians And Bicycles

This section describes the implementation details to make the system aware of pedestrians and bicycles. In the developed algorithm, the pedestrians/bicycles stage is performed concurrently with a corresponding chain of the controlled intersection. For instance Figure 3.1 depicts a simple intersection (i.e. “Richtlinie für Lichtsignalanlagen” (guidelines for traffic light systems), or RiLSA) with four incoming/outgoing edges plus two sets of zebra crossing edges. The scenario shows that the pedestrians/bicycles stage for west-east and east-west crossing is called when the traffic light chain west-east and east-west is in Green stage.

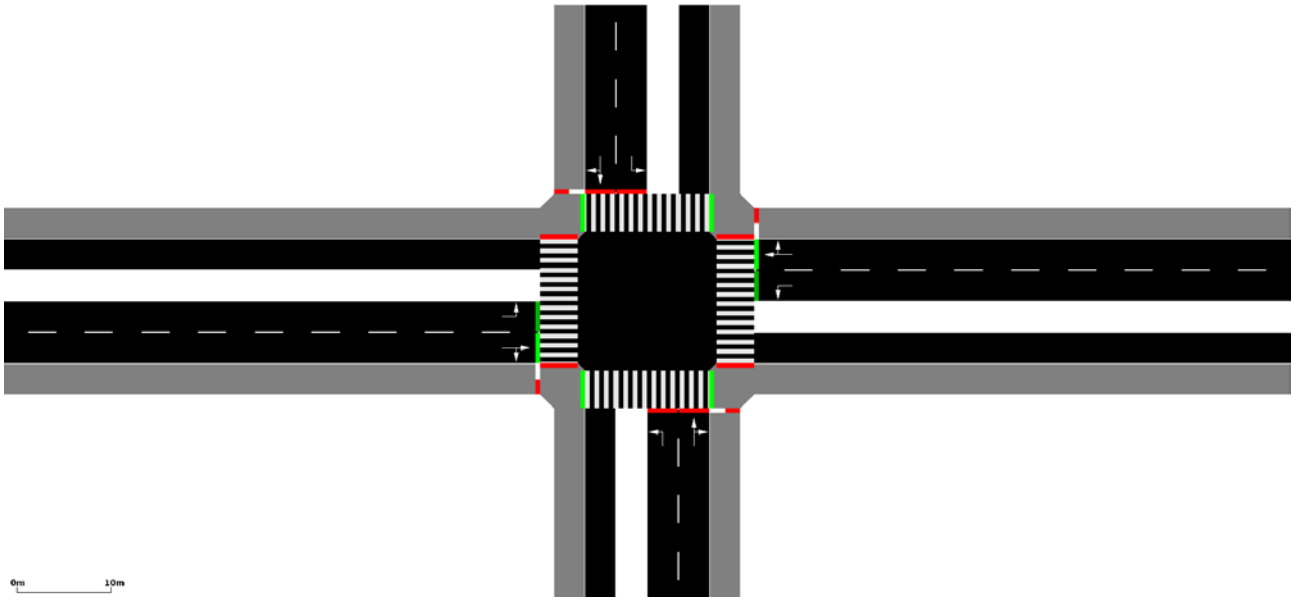

Figure 3.1: The RiLSA example with green pedestrian/bycicles stage on the west-east direction.

The implemented controller models and manages the pedestrians in the same way like bicycles. In the following, the term “pedestrians” is therefore used for both pedestrians and bicycles.

3.1 Pedestrians and Bicycles implementation

As stated before, the duration of the current green phase is decided by the selected microscopic policy. Depending on the logic of the specific policy, the controller might keep the current green until a vehicle waiting on a lane served by a red is detected. This behaviour can clearly lead to long waits for the pedestrian waiting at a crossing.

In our model, each pedestrian can demand a green phase using a pushbutton placed in correspondence of every crossing. Whenever the system detects a request, it notifies the presence of pedestrians to the selected microscopic policy.

The development of the pedestrian support was carried out in two phases: in the first phase we extended the SUMO simulator adding the pushbutton control while in the second phase we defined and implemented the management logic.

3.1.1 Pushbutton Implementation

To simplify the implementation of the pedestrian logic in the system we assumed that every pedestrian presses the pushbutton at its arrival at a red crossing. The request will also stay active, advising the presence of waiting pedestrians, until they are allowed to cross the intersection.

A zebra crossing in SUMO is a particular edge that lets pedestrians across a road. It is preceded and followed by a walking area, which is an edge that allows only pedestrians passage. For every crossing edge, the system creates two different pushbuttons, one on both sides of the road. Each of these buttons is identified by a pair `<walking area, crossing edge>`. Since we suppose that a pedestrian pushes the button when it starts waiting, the request can be considered active if at least a pedestrian, currently on the related walking area, is waiting to cross the corresponding crossing edge.

Note that usually each phase has more than one pushbutton connected, since any of its controlled edges can be crossed by a crosswalk. If at least one pushbutton gets a request, the policy is informed so that it can take the appropriate actions.

The only information collected by the system is the state of the pushbuttons (active or not active), without any indication about the number of people waiting at the crossing or how many times the pushbutton has been pressed.

In the simulator, each pushbutton is created with the attribute *crossingEdges* of the element *edge*, specified in the SUMO road network file[1]. An *edge* with function of crossing can have this attribute listing which edges it crosses. During the initialization phase, while parsing the network file, the implemented system loads the set of *crossingEdges*, using it to create the coupled pushbuttons. For each phase of a traffic light, SUMO checks every controlled edge defined in the phase itself to find if a crossing edge is associated to it. SUMO instantiates the appropriate pushbuttons, if a match is found.

3.1.2 Logic definition

Initially, the logic implemented (called *default logic*) managing pedestrians was similar to the static one for vehicles. In particular, when a pushbutton was active, an upper bound on the duration of the current decisional phase was imposed equal to the static duration. No action was taken if the selected policy decided to change the phase before this duration was reached, otherwise the phase change was mandatory. If the static duration was already exceeded when a pushbutton became active, then the logic would force the change of phase as next step.

This *default logic*, using the static stage duration, might give a higher priority to the waiting pedestrians w.r.t. the vehicles. In fact, for some policies, the phase duration can be significantly reduced. Depending on the vehicles flow, the traffic light position in the road network, and the flow of pedestrians, this *default* behaviour could drastically increase the waiting time of the vehicles.

An improved version of the logic has therefore been implemented by introducing a configurable *scale factor* for the static duration. Low values give priority to pedestrians, while high values give priority to the vehicles. In particular, values lower than one decrease the upper bound duration of a

[1] Please refer to the SUMO wiki for a detailed description: http://sumo.dlr.de/wiki/Networks/SUMO_Road_Networks

phase when the pushbutton is active, therefore reducing the waiting time of the pedestrians; values higher than one have the opposite effect, reducing the waiting time of the vehicles.

3.2 Experimental results

The experiments highlight the impact of the values adopted for the *scale factor* on the overall waiting time of the vehicles.

The designed algorithm has been evaluated on a synthetic, but realistic network. The network consists in a grid of 16 RiLSA traffic lights, where the swarm algorithm controls the four central ones, while the outer 12 use the SUMO actuated method.

The zebra crossing are placed in correspondence with every traffic light, but only the one controlled with the swarm algorithm takes into account the presence of the pedestrians since the SUMO actuated controller does not alter its behaviour if there are pedestrians waiting.

The presented results are obtained by averaging the sum of the waiting times of the vehicles on every edge over the number of vehicles with the formula below:

$$avgWait = \frac{\sum_{i=1}^{n} waitingTime_{i-edge}}{carNumber} \qquad 3.1$$

The Figure 3.2 shows how the average waiting time of the vehicles varies for different values of the scale factor. While the leftmost configuration does not take into account the pedestrian requests, the other use this information and use different values for the *scale factor*, respectively, from left to right, the values are 2, 1.75, 1.5, 1, 0.75 and 0.5. Decreasing the values of this parameter has a significant effect on the overall performance of the system.

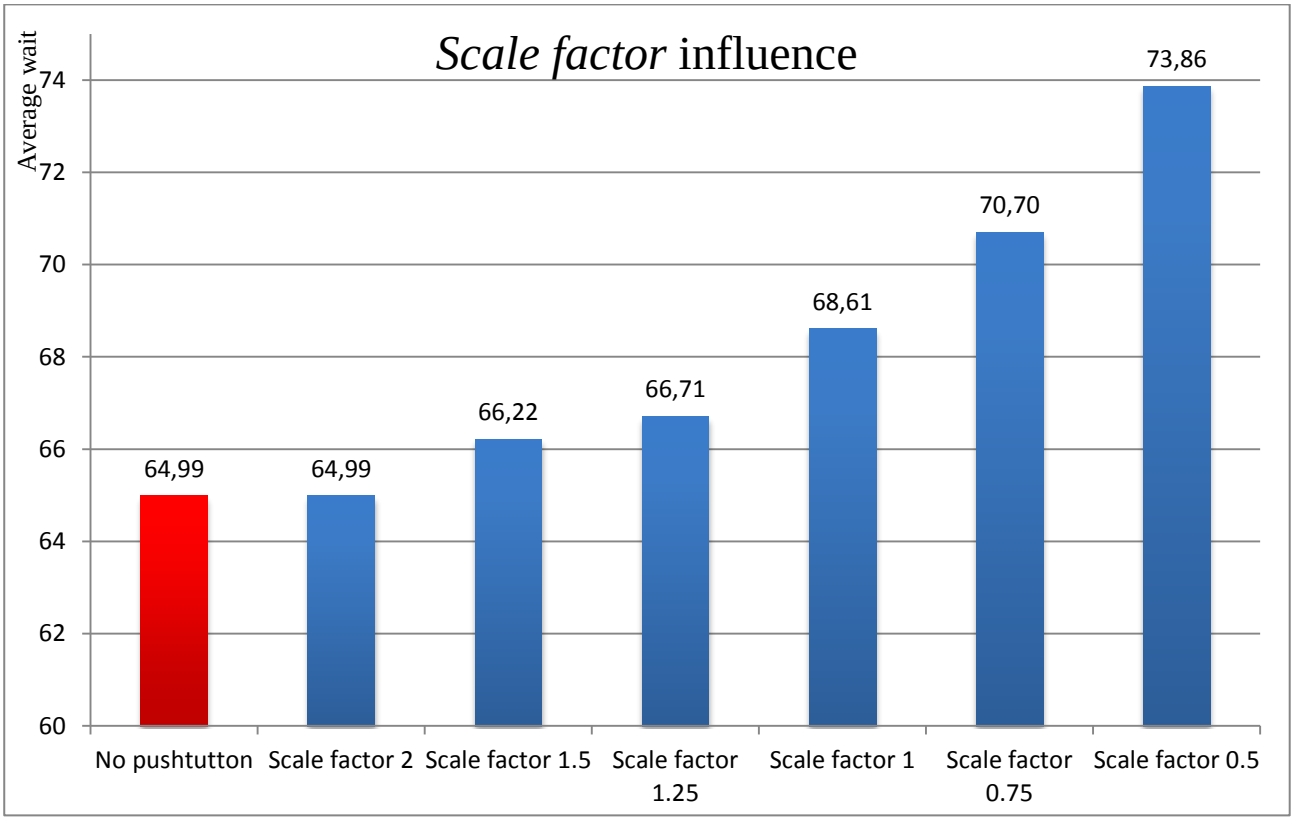

Figure 3.2: Average waiting steps with different values for the scale factor.

4 Policies for Public Transport and Emergency Vehicles

This section presents the extensions to the traffic light control system for the management of public and emergency vehicles. In the actual implementation of the SUMO simulator a vehicle is assigned to a type by using the attribute `type`[2] in the route file. The type specifies a collection of properties common to a set of vehicle (e.g. the physical dimension, the maximum speed,…). In our approach, we have defined two types of vehicles: the *standard* ones (i.e. the private vehicles) and the *special* ones (i.e. Public Transport and Emergency Vehicles).

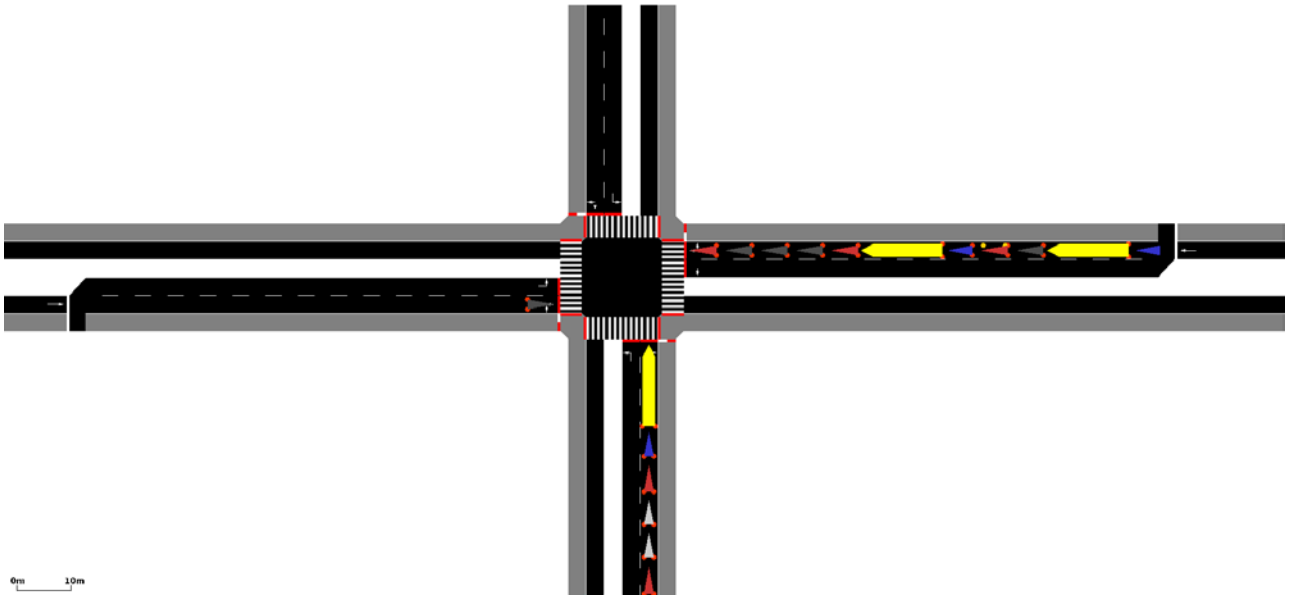

Figure 4.1: The RiLSA example with standard and special vehicles.

The vehicles belonging to the *special* type are graphically plotted as long yellow vehicles (see Figure 4.1).

4.1 Special Vehicles Management

As detailed in Section 2, the microscopic level policy decisions are based on the number of waiting cars and on the waiting times. In particular, the approach relies on the definition of a threshold that is proportional to the number of vehicles waiting at a red light and the time they have been waiting, for each incoming lane. A high value means that there are many cars waiting for a long time on that lane. Hence, the lane with the highest value has the higher probability of being chosen for the next green stage. Furthermore, these values influence the length of the green lights; in fact the lanes with higher values usually are served with longer green phases. We call this value the *carsNumber*.

In the initial implementation, the traffic light controller considered all the vehicles *as equal* when selecting the duration of the green phases. By increasing the weight of the vehicles according to their type, we can promote such vehicles, so that their average waiting time decreases.

[2] Please refer to the SUMO wiki for a detailed type description:
http://sumo.dlr.de/wiki/Definition_of_Vehicles,_Vehicle_Types,_and_Routes#Vehicle_Types

Whenever a microscopic level policy takes a decision, the algorithm checks if, on any edge not served by the current phase (meaning that the corresponding light is red), the *carsNumber* is over a defined threshold. If any, the policy can react accordingly: for instance by deciding to change the chain in order to serve a lane with a higher *carsNumber*.

The increase of the weight of a vehicle directly affects the computation of the *carsNumber* value of the edge where the special vehicle is waiting, boosting its growth. The higher the weight assigned to a type, the faster the threshold on its edge will be reached and exceeded.

Recalling the previous definition of *standard* and *special* type, a *standard* type has the default weight of one, while a *special* type has a weight greater than one, usually defined by the parameter tuning procedure.

4.1.1 Implementation

Instead of a fixed set of weights for a predetermined collection of vehicles types, we added a parameter in the configuration file of the traffic light controller, which lets us specify a list of types with their corresponding weights. The default weight of every vehicle is one; by setting the weight of the type *bus* for instance to 10, the system will count it as if it were 10 vehicles.

Below are shown the parameters of the additional file declaring the weights of the vehicles.

```
<param key="USE_VEHICLE_TYPES_WEIGHTS" value="1" />
<param key="VEHICLE_TYPES_WEIGHTS" value="bus=10; police=50" />
```

Parameters that control the weight of the vehicles

The first parameter is a *boolean* value specifying if the logic uses or not different weights for the vehicle types. The second one is a list of types with their respective weight, encoded with the syntax `"type=weight"`. As shown in the code, the parameter can contain a list of multiple types using a semicolon `";"` as separator.

4.2 Experimental Results

The goal of the experiments is to highlight that the developed approach is viable and robust for low penetration scenarios.

The designed algorithm has been evaluated on the same network described in the chapter 3.2.

For each penetration rate (100%, 50%, 25%, 10%, 5%, 2.5%, and 1%) we generate 10 realistic traffic flows. The configurations used for the swarm controller is unique for each penetration rate, and has been obtained via automatic parameter tuning.

The quality of the method is assessed using the average waiting time of the vehicles during the simulation. These values are computed using the edge-based traffic measures available in SUMO[3] considering all vehicles or only the set of special ones (by employing the attribute `vTypes` to select only the *special* types).

As before, the presented results have been obtained with the equation `3.1`, but in this case $waitingTime$ and $carNumber$ are relative to all vehicles or only to the *special* ones. For the presented experiments we set up two cases:

[3] More information can be found in SUMO wiki: http://sumo.dlr.de/wiki/Simulation/Output/Lane-_or_Edge-based_Traffic_Measures

- *Standard*: where all the vehicles have weight 1
- *Improved*: where the special vehicles have higher weights (the value is decided from the tuning). This case represent the developed improvement.

The Figure 4.1 shows the average waiting time of the special vehicles in both cases. The figure shows that the average waiting step of the special vehicles of the improved case is certainly lower w.r.t. the *standard* one. We resemble that each penetration has a different configuration and runs on different traffic flows; hence the results for different penetrations cannot be compared.

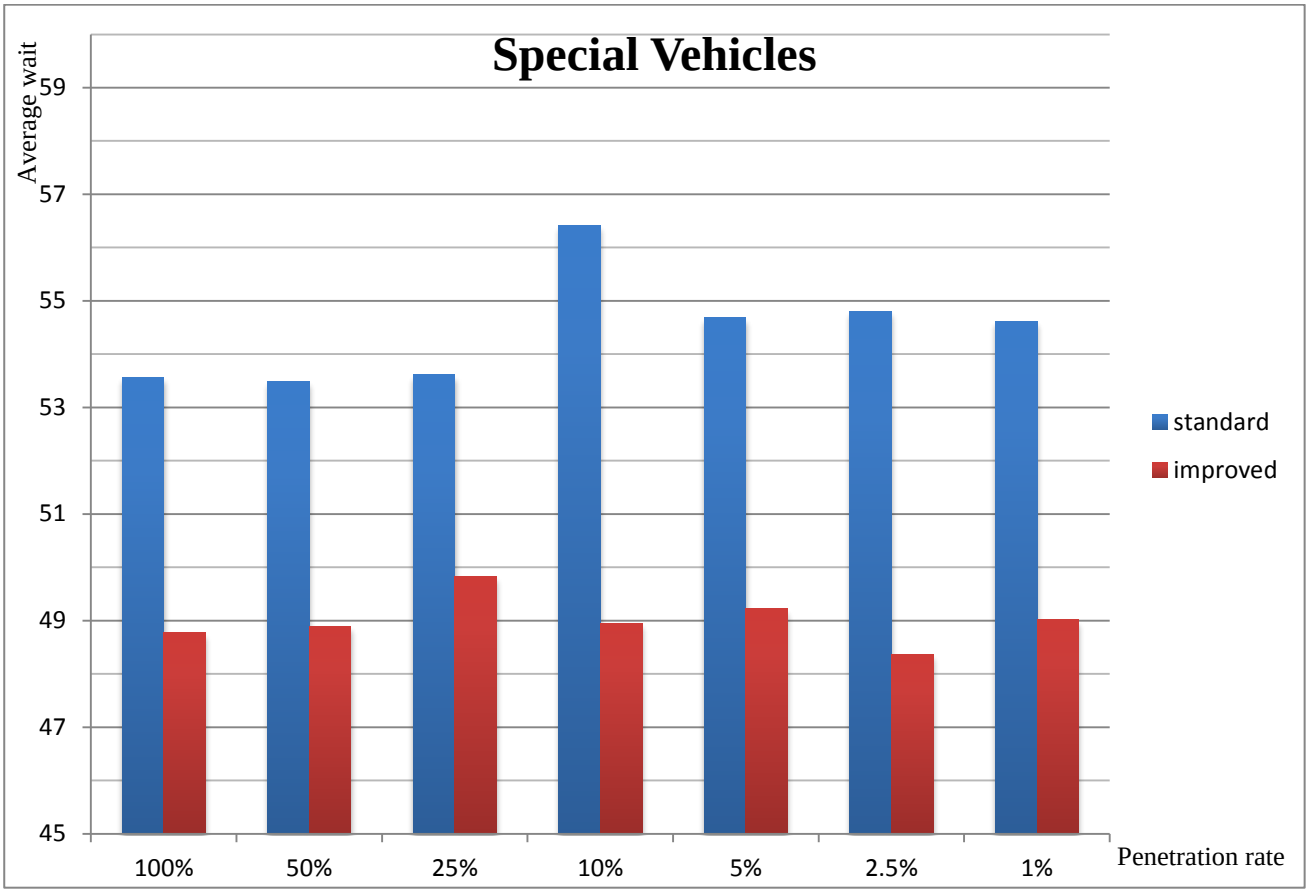

Figure 4.2: Average Waiting steps of Special Vehicles.

Figure 4.2 presents the average waiting time considering all vehicles. This value is important since the method used to favour the special vehicles could in principle penalize the other vehicles. The figure shows that even if special vehicles have priority, overall the vehicles are not penalized. The results of both the *standard* and the *improved* case are similar. Surprisingly the *improved* scenarios show better results, especially in low penetration rate experiments.

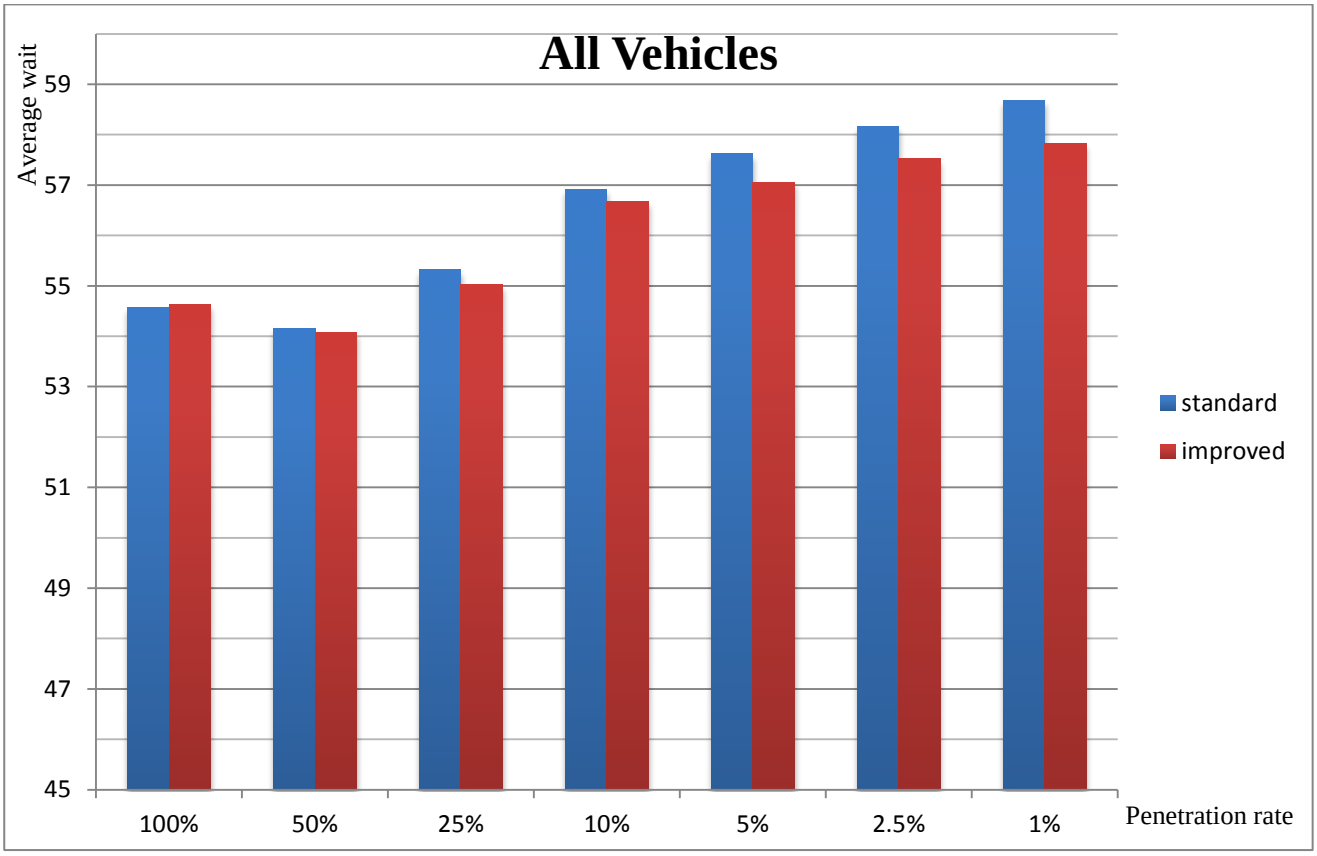

Figure 4.3: Average Waiting steps of all Vehicles.

Penetration Rate	Special Veh. Speed Up %	All Veh. Speed Up %
100%	**8.94%**	**-0.11%**
50%	**8.60%**	**0.15%**
25%	**7.05%**	**0.54**
10%	**13.22%**	**0.42%**
5%	**9.99%**	**0.97%**
2.5%	**11.73%**	**1.12%**
1%	**10.25%**	**1.48%**

Table 4.1: Speed Up of special vehicles and of All vehicles.

Table 4.1 shows the speedup values for the special vehicles and the overall results.

The speedup values presented in the Table 4.1 are computed with the following formula:

$$speedUp_{\%} = \left(\frac{avgWait_{all} - avgWait_{special}}{avgWait_{all}}\right) \times 100$$

Where $avgWait_{all}$ and $avgWait_{special}$ are values obtained from the equation `3.1` considering all vehicles and only the *special* ones, respectively.

5 Summary

This document presents the extension to the COLOMBO system to take into account pedestrian and bicycles management and give precedence to the public transports and emergency vehicles, as required by Task 2.5 of the project's Work Package (WP) 2.

The system has been extended on two sides: firstly, with the implementation of a pushbutton abstraction to signal the presence of pedestrians and the creation of logic to properly manage their requests; secondly, with the ability to assign different weights to a set of *special* vehicles with the aim of reducing their overall wait time.

Both extensions proved to be fully functional in initial operation. Experiments were conducted to analyse the impact of the choice of the scale parameter value and of different vehicle type weight values.

The experimental results show, that the scale factor is effective in controlling the behaviour of the traffic light controller. A value of 2 for the scale factor (meaning high priority for motor vehicles) gives similar average waiting times as if no pedestrians and bicycles were accounted for. A value of 0.5 for the scale factor (meaning high priority for pedestrians) reduces the waiting times for pedestrians but considerably increases the average waiting times of vehicles.

Experiments regarding the weight assignment to different types of vehicles covered the situation of giving equal weight to all vehicles and giving different weights to different types of vehicles (Car = 1, Bus = 10, Police=50). Surprisingly, in the latter case the average waiting time for all vehicle was usually lower in the range of 0% - 1.5% in the experiments. Tests at different penetration rates proved that the algorithms work very well even at a Car2I penetration rate as low as 1%, where the average waiting time measured was only 6% higher than at a Car2I penetration rate of 100%. This result is very promising with respect to the goal of COLOMBO to provide solutions effective in very low Car2X penetration rate environments as expected on European roads for the near future.