Australasian Transport Research Forum 2013 Proceedings
2 - 4 October 2013, Brisbane, Australia
Publication website: http://www.patrec.org/atrf.aspx

A NOVEL METHODOLOGY FOR EVOLUTIONARY CALIBRATION OF VISSIM BY MULTI-THREADING

Kayvan Aghabayk[1]*, Majid Sarvi[1], William Young[1] & Lukas Kautzsch[2]

[1] Institute of Transport Studies, Department of Civil Engineering, Monash University, Australia

[2] PTV Planung Transport Verkehr AG, Karlsruhe, Germany

* Corresponding Author: Kayvan.Aghabayk@monash.edu

Abstract

Traffic micro-simulation models have become very popular in transport studies and are extensively used in research and by industry. Traffic simulation models are especially very useful in reflecting the dynamic nature of transportation system in a stochastic manner, which is beyond the capability of some classic methods. Nevertheless, one of the major concerns of micro-simulation users has been the appropriate calibration of this software. An inappropriate calibration may end in a wrong conclusion and decision which could result in irreparable costs or problems. This paper develops an efficient methodology to improve the calibration procedure of traffic micro-simulation models. It applies the method to VISSIM. It provides a methodology for auto-tuning of VISSIM as one of the well-known and commercially available traffic micro-simulations. More specifically, it looks at the car-following and lane changing models as they form the main component of any traffic micro-simulation. This approach uses particle swarm optimisation (PSO) method as an evolutionary algorithm through the VISSIM COM interface and parallel optimisation technique to reduce the cost of auto-tuning and calibration. This paper could be of interest to transport experts in particularly those who are using traffic micro-simulation and looking for auto-calibration approach.

Keywords: Traffic micro-simulation, Calibration, Auto-tuning, Evolutionary algorithm, Parallel optimisation, Multithread, Particle Swarm Optimisation (PSO)

1. Introduction

Traffic simulation models have become an important and popular tool in modelling transport systems, in particular, owing to advent of fast and powerful computers. One of the supreme advantages of using such tools is to assess different alternates and scenarios prior to their implementations. Traffic simulation models could be divided into three categories including microscopic, macroscopic, mesoscopic simulation models. First category simulates the movement of individual vehicles in a traffic stream. Car-following and lane-changing models are the two fundamental components in traffic micro-simulations. The second (macroscopic) category simulates transportation network section-by-section rather than by tracking individual vehicles. The relationships between flow, speed, and density of traffic stream form the fundamental basis of this category. Mesoscopic traffic simulation models combine the properties of the first and second models.

Along with the increasing popularity and use of traffic simulations, an essential concern has been raised about their proper applications in the study they are used for; or more specifically their appropriate calibration and validation. As no single model can comprise the whole universe of variables, every model must be adapted for local conditions using real world data. The performance of the model should be also evaluated through independent data sets. These processes are known as calibration and validation. More specifically, calibration of the traffic simulation normally refers to computing the magnitude of the parameters embedded in the simulation models to match the real traffic and local driving behaviour.

The outputs of traffic simulations may not be accurate and reliable without appropriate calibration. However, the calibration process, especially for microscopic simulations, could be a complex and time-consuming task because of the large number of unknown parameters (Toledo et al. 2004). Some studies (e.g. Gardes et al. 2002, Chu et al. 2003, Park and Schneeberger 2003, Moridpour et al. 2012) used the generic procedure to calibrate traffic micro-simulations using sensitivity analysis and trial-and-error which could be very resource-intensive and time-consuming. Some other studies (e.g. Lee et al. 2000, Ma and Abdulhai 2002, Park and Qi 2005, and Menneni et al. 2008) used an evolutionary algorithm like Genetic Algorithm (GA) for calibration purposes. However, the process may be still time-consuming due to the significant computational load associated with large-scale traffic simulation runs.

This study introduces a methodology to calibrate traffic micro-simulations based on an evolutionary algorithm known as Particle Swarm Optimisation (PSO). To overcome the long running time, it applies multi-thread technique and implements Parallel PSO algorithm. This method can use several CPUs and runs several simulation instances in parallel which can shorten the run time significantly. VISSIM (2012) traffic micro-simulation was used in this study for implementation of the algorithm.

Section 2 explains the VISSIM interface and defines the important parameters that should be considered for calibration procedure. In particular, the parameters related to the driving behaviours are discussed in this section. Section 3 explains about the optimisation techniques and the particle swarm optimisation (PSO) algorithm is discussed in details. Section 4 presents the implementation and parallelisation of the PSO algorithm for auto-calibration of the traffic micro-simulation. The paper is closed by providing some conclusions and further remarks for future work in Section 5.

2. VISSIM and Calibration Parameters

The micro-simulation which was used in this study is VISSIM (2012), version 5.40. The name is derived from “Verkehr In Städten - SIMulationsmodell” (German for “Traffic in cities - simulation model”). The software was developed at the University of Karlsruhe, Karlsruhe, Germany, during the early 1970s. Commercial distribution of VISSIM began in 1993 by PTV Transport Verkehr AG, which has continued to distribute and maintain VISSIM till now. VISSIM is one of the latest traffic micro-simulations available and provides significant enhancements in terms of driver behaviour, multi-modal transit operations, interface with planning / forecasting models, and 3-D simulation.

VISSIM (2012) is a microscopic, time step and behaviour based simulation model developed to analyse private and public transport operations under constraints such as lane configuration, vehicle composition, traffic signals and so on. Access to model data and simulation is provided through a COM interface, which allows VISSIM to work as an Automation Server and to export the objects, methods and properties. The VISSIM COM interface supports Microsoft Automation and thus the program can be implemented in any of the RAD (Rapid Application Development) tools ranging from scripting languages like Visual

Basic Script or Java Script to programming environments like Visual C++ or Visual J++. Also, internal driving behaviour can be replaced by a fully user-defined behaviour using the External Driver Model DLL Interface of VISSIM.

The accuracy of the traffic flow simulation model is highly related to the accuracy of estimating vehicles' movements in the network. The driving behaviour in the micro-simulation is linked to each link by its behaviour type. The traffic flow model in VISSIM is a discrete, stochastic, time step based, microscopic model with driver-vehicle-units as single objects. The model contains a psycho-physical car-following model for longitudinal vehicle movement and a rule-based algorithm for lateral movements. The model is based on the continued work of Wiedemann (1974) for car-following process and Wiedemann and Reiter (1992) for lane-changing manoeuvres. The car-following and lane changing models and their associate parameters are explained in the following sub-sections. These parameters affect the vehicle interactions directly and cause substantial differences in simulation results and thus should be considered in the calibration procedure particularly. However, the calibration parameters are not limited to the parameters associated with the car-following and lane changing model. Some further parameters which could be considered are used to define the maximum and desired acceleration/deceleration of vehicles, desired speed distributions, and signal control. Here two fundamental components within the traffic micro-simulations are presented as an example, but the rest of parameters can be also considered in calibration procedure. The proposed algorithm allows setting as many parameters as desired for calibration. However, more parameters will result in more complex process and thus will require more running time.

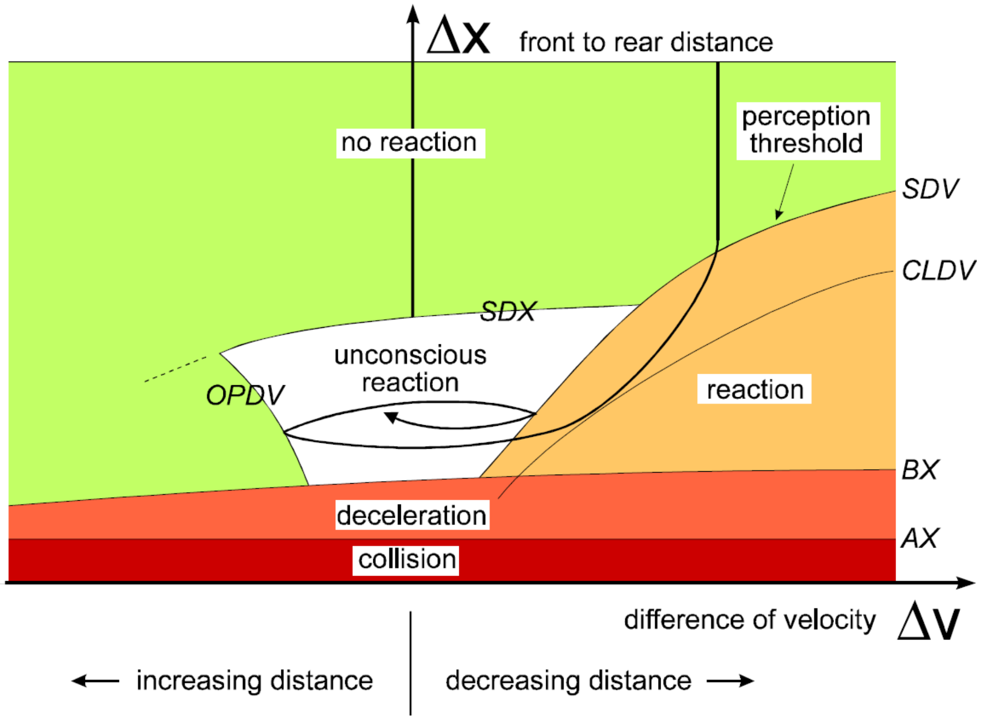
2.1. Car-following model

VISSIM uses the psychophysical car-following model developed by Wiedemann (1974). The concept of this model is that the faster moving vehicle drivers approaching slower vehicle start decelerating when they reach their own individual perception threshold. However, the speed may become smaller than the lead vehicle speed as the results of driver's imperfection in the estimation of the lead vehicle speed. This means the driver will accelerate slightly again after reaching another threshold. This results in an iterative process of acceleration and deceleration due to drivers' imperfections to determine the exact speeds of the lead vehicles. Figure 1 shows a typical car-following behaviour of a vehicle based on the logic explained above.

There exist two car-following models in VISSIM: Wiedemann74 and Wiedemann99 (VISSIM 2012). The former one is suggested to be applied for urban arterial roads and the later one is more suitable for freeways. The basic idea of the models is the assumption that a driver is in one of the four driving modes: Free driving, Approaching, following or braking. These modes are determined by the following six thresholds (also shown in Figure 1):

- AX: the desired distance between two stationary vehicles
- BX: the minimum following distance which is considered as a safe distance by drivers
- CLDV: the points at short distances where drivers perceive that their speeds are higher than their lead vehicle speeds
- SDV: the points at long distances where drivers perceive speed differences when they are approaching slower vehicles
- OPDV: the points at short distances where drivers perceive that they are travelling at a lower speed than their leader
- SDX: The maximum following distance indicating the upper limit of car-following process

Figure 1: a typical car-following behaviour of a vehicle (VISSIM 2012)



More details about these thresholds can be found in Wiedemann (1974) or VISSIM (2012). For each mode, the acceleration could be determined as a result of speed, relative speed, space headway and the individual characteristics of driver and vehicle.

This study explains the Wiedemann99 car-following model with more details as it is more suitable for freeways and contains more parameters causing more difficulties for calibration. Further, the relation between the calibration parameters and the perceptual thresholds are required to be investigated in details. Figure 2 is a snapshot of VISSIM interface showing the parameters associated with Wiedemann99 car-following model. The parameter explanations are presented below.

The 'Look ahead distance' parameters determine the minimum and maximum distances as well as the number of vehicles in front of a driver in the same link that can be observed and thus influence driver's reaction accordingly.

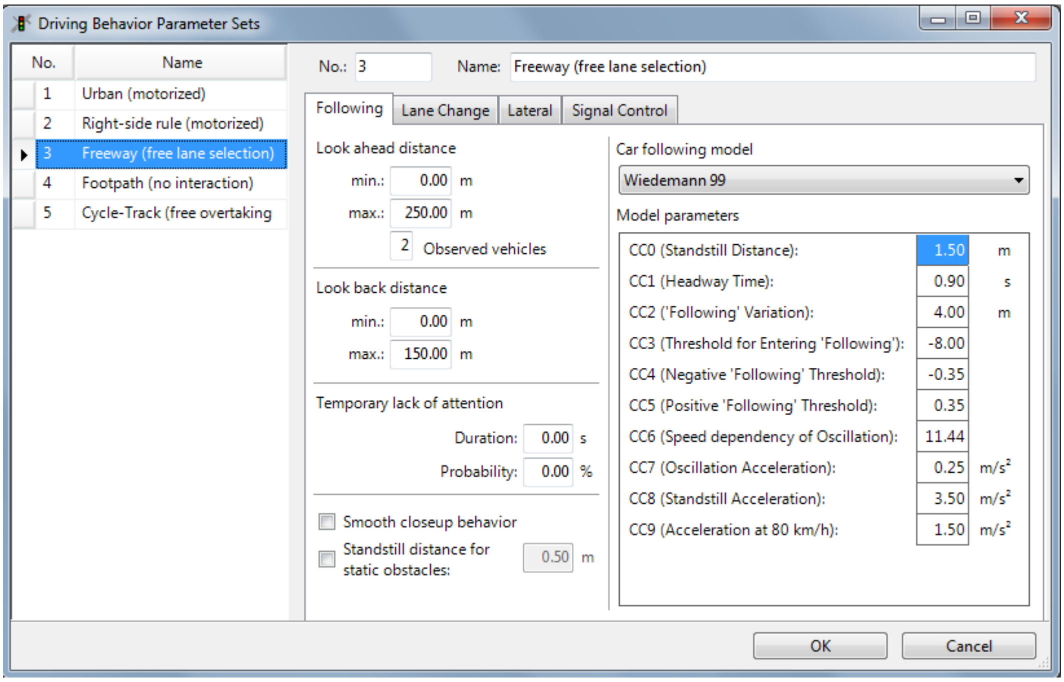
The 'Look back distance' parameters determine the maximum and minimum distances that a driver can see backwards within the same link in order to react to other vehicles behind.

The 'temporary lack of attention' parameters determine the probability and time duration in which a driver does not react to the behaviour of lead vehicle (except for emergency braking).

The 'smooth closeup behavior' parameter determines whether or not drivers slow down more smoothly when approaching standing obstacles. If it is checked, drivers will prepare to stop behind the obstacle from the maximum look ahead distance. If it is not checked, drivers will have the normal following behaviour and will not consider the obstacle from the long distance.

The 'standstill distance for static obstacles' parameter determines the distance that drivers keep while standing in upstream of all static obstacles. The distance can be fixed by checking the box otherwise it would be a random value following a normal distribution with the mean of $0.5\ \mathrm{m}$ and variance of $0.15\ \mathrm{m}^2$.

Figure 2: Parameters of Wiedemann99 car-following model in VISSIM



In the box of car-following model parameters, there exist ten parameters (CC0-CC9). The first seven parameters (CC0-CC6) are used to determine the car-following thresholds and the rest have different roles. The relation between the parameters and thresholds are defined by Equations 1 to 6.

$$AX = L + CC0 \qquad Eq.(1)$$

where L is the length of the lead vehicle

$$BX = AX + CC1 \times v \qquad Eq.(2)$$

where v is equal to subject vehicle speed if it is slower than the lead vehicle; otherwise, it is equal to lead vehicle speed with some random errors. The error is determined randomly by multiplying the speed difference between the two vehicles by a random number between -0.5 and 0.5.

$$SDX = BX + CC2 \qquad Eq.(3)$$

$$(SDV)_i = -\frac{\Delta x - (SDX)_i}{CC3} - CC4 \qquad Eq.(4)$$

where Δx is the space headway between the two successive vehicles calculated from front bumper to front bumper.

$$CLDV = \frac{CC6}{17000} \times (\Delta x - L)^2 - CC4 \qquad Eq.(5)$$

$$OPDV = -\frac{CC6}{17000} \times (\Delta x - L)^2 - \delta.CC5 \qquad Eq.(6)$$

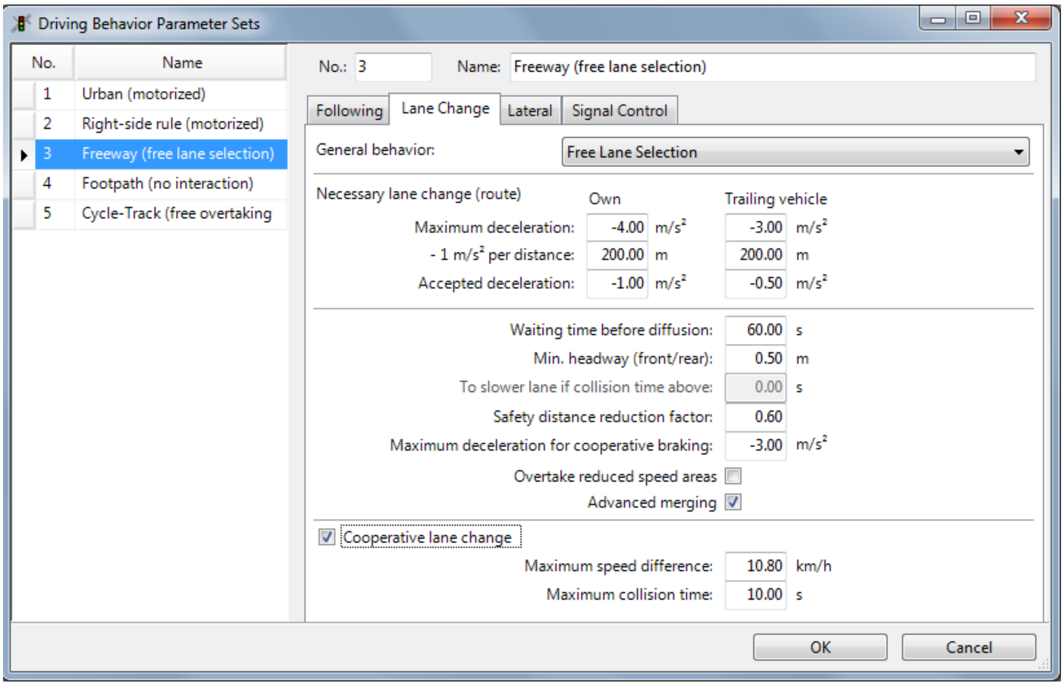
where δ is a dummy variable which is equal to 1 when the subject vehicle speed is greater than CC5 and 0 else.

The CC7 parameter defines the actual acceleration during the oscillation process; The CC8 parameter defines the desired acceleration when starting from standstill condition; and the CC9 parameter determines the desired acceleration at the speed of 80 km/h.

2.2. Lane changing model

There are two types of lane changing models used in VISSIM: necessary lane change and free lane change. Necessary lane change (Figure 3) considers the situations in which drivers should change their lane in order to reach the next connector of a route. The driving behaviour parameters in this type of lane change contain the maximum acceptable deceleration for the lane changing vehicle (own) and the vehicle will be its follower in the target lane (trailing vehicle). The target lane is the lane that the driver wishes to move into. The free lane change considers a lane change of a vehicle in order to obtain speed advantages or more space. This lane change will take place if the desired safety distance in the target lane is satisfied. This safety distance is determined by speed of lane changing vehicle and trailing vehicle in the target lane. The parameters associate with the lane changing model are outlined briefly below. . For more details, the readers are referred to the VISSIM user manual (VISSIM 2012).

Figure 3: Parameters of lane changing model in VISSIM



The 'general behavior' defines the way of overtaking and contains two options: 'free lane selection' and 'right side rule' / 'left side rule'. The overtaking is allowed in any lane by using the first option but it will have some limitations if the second option is selected. As the parameters are the same in the two options, this study focuses on the parameters and does not explain the limitations

The aggressiveness of lane changing manoeuvre is defined by deceleration thresholds for the lane changing vehicle (own) and its following vehicle in the target lane (trailing vehicle).

The thresholds are determined by the six parameters appeared under the ‘necessary lane change (route)’ section.

The ‘waiting time before diffusion’ describes the maximum time that a vehicle wait at an emergency stop position to find a gap to change lanes and stay on its route. If the vehicle cannot change its lane within this time, it will be taken out from the network and an error message will be appeared in the VISSIM error file.

The ‘min. Headway (front/rear)’ determines the minimum gap in front of the vehicle that should be available for a lane change in standstill condition.

The Safety distance reduction factor considers a deduction in the safety distances associated with vehicles involved in the lane changing manoeuvre. Smaller values result in more aggressive lane changing behaviour. For instance, the default value (0.6) results in 40 precent deduction in the safety distances.

The ‘maximum deceleration for cooperative braking’ parameter determines the maximum deceleration the trailing vehicle driver will accept for cooperation helping the lane changing vehicle to execute its manoeuvre. A higher absolute value will result in more cooperation and thus lane changing.

The ‘overtake reduced speed areas’ is unchecked by default. However, it can be checked resulting in model lane-dependent speed restrictions which are considered by the vehicles for lane changing.

The ‘advanced merging’ is checked by default and considers any necessary lane change belongs to the next link or connector along with the current connector. This means that the strategic plan of vehicles can be considered and they may change lanes earlier. This may decrease the waiting time for lane changing and increase the capacity of the road.

If the ‘cooperative lane change’ is checked, the trailing vehicle in the target lane may move to the other side in order to make room for the lane changing vehicle However, the second lane change will occur if it is safe and suitable for the trailing vehicle based on its own route plan. Further it does execute a cooperative lane change if the defined ‘maximum speed difference’ and ‘maximum collision time’ are not satisfied.

3. Particle Swarm Optimisation

The determination of the abovementioned parameters requires a trial-and-error or an optimisation procedure. This section outlines the procedure used to develop the optimal values for the parameters.

In mathematics, optimisation refers to the problem of finding the extremum value of a function. So, it can refer to both minimization and maximization. Since the maximization of an arbitrary function f is equivalent to the minimization of its negative function(–f), the minimization and optimisation terms are usually used interchangeably.

Optimisation methods can be categorized into different classes based on different points of view (Sedlaczek and Eberhard 2006, Engelbrecht 2007, Wang and Li 2009, Zheng and Wan 2012, Komori et al. 2012). From one perspective we can classify them into constrained optimisation and non-constrained optimisation methods as in some circumstances the underlying problem imposes several constraints and limitations to the variables (e.g. limitation on the boundaries of variables or optimising the function in a sub-area of the space). In such cases, the optimisation problem should be solved satisfying all the constraints of the problem (i.e. equality and inequality constraints). In fact, the solutions that do not satisfy the constraints are not acceptable. This kind of optimisation problems has its own developed methods. For example, Lagrangian optimisation method (Rockafellar 1993) is a well-known method that is widely used in this context. Non-constrained optimisation refers to the global optimisation of the function without any limitations. In fact, the feasible space is equal to the whole search space in non-constrained optimisation problems.

Optimisation has been extensively studied in mathematics literature (Rockafellar 1993, Snyman 2005, Polyak 2007, Wang and Li 2009, Narushimaa and Yabeb 2012, Zheng and Wan 2012). Using the derivatives of the underlying function leads to a family of methods called gradient-based methods (Snyman 2005, Narushimaa and Yabeb 2012). These methods start from an initial point and make use of the gradient of the function to improve the solution in each step. Some of these methods use the first order derivatives while some others make use of higher order derivatives. For example, gradient descent, conjugate gradient and several other methods use the first order derivatives but Newton's method (Polyak 2007) uses the second order derivatives and uses the Hessian matrix in order to improve the rate of convergence (Boyd and Vandenberghe 2004).

Mathematical optimisation methods have their own limitations when the underlying function is non-convex. When the function is convex, it has definitely a single minimum point (best solution) and several convex optimisation methods could be used to find this point. These methods guarantee that they find this single minimum point of the function starting from any arbitrary initial point. Although they differ from each other in terms of convergence rates and the number of iterations, all of them are able to find the global minimum of the function. However, several challenges and difficulties arise when the underlying function is non-convex which may have several local minimums. In these cases, mathematical optimisation methods are sensitive to the initial points and may result in a local minimum instead of the global minimum of the function (Szu 1986). Further, in some optimisation problems the goal is to achieve the best parameters for a system in which there is no specific mathematical function. Therefore, the mathematical methods are not applicable in these problems.

To overcome the abovementioned problems, another class of optimisation methods could be applied to find the best solution. This category consists of evolutionary algorithms using computational intelligence or soft computing methods. They are inspired from biological evolution or swarm intelligence (collective animal behaviour). The two most well-known methods in this category are Genetic Algorithms (Holland 1975) and Particle Swarm Optimisation (Kennedy and Eberhart 1995). These methods use several random search elements in parallel to avoid local minima.

Genetic algorithms use two main operator named mutation and cross-over. The mutation operator causes random movements in the population and causes exploration and avoiding local minimums. The cross-over operator leads to fine movements and exploitation around the current position of population. Although there exist continuous extensions to genetic algorithms, it is mainly used in binary and discrete optimisation.

Particle swarm optimisation (PSO) was first introduced by Kennedy and Eberhart (1995, 2001). The idea behind the PSO algorithm came from the collective motion of a flock of particles: the particle swarm. In an n-dimensional function, each particle is presented as an n-dimensional vector representing a single point in the search space. The position of the i^{th} particle ($\vec{x}_i$) can be presented as:

$$\vec{x}_i = (x_{i1}, x_{i2}, \dots, x_{in}) \qquad Eq.(7)$$

Each particle moves through the multi-dimensional search space and its position and velocity are updated by two elastic forces defined based on its own experience and the experience of the swarm. The procedure continues until the swarm as a whole converges to an optimum value.

Each particle memorizes the position vector and velocity vector as well as the best individual (personal) point which has produced the minimum value of the cost function. This point is called personal best (p^{best}). The best personal experience of the i^{th} particle can be expressed as:

$$\vec{p}_i^{best} = (p_{i,1}^{best}, p_{i,2}^{best}, \dots, p_{i,n}^{best}) \qquad Eq. (8)$$

The particles can share their experience in the flock and can be affected by the best experience of the swarm. The best experience of the swarm is the point which has produced the minimum value of the cost function within the whole population. This point is called global best (g^{best}).

The basic concept of the PSO algorithm lies in updating the position of each particle towards its p^{best} and g^{best} points at each iteration by adding an increment vector called velocity. The velocity and position of the i^{th} particle at the $(k+1)^{th}$ iteration can be determined by:

$$\vec{v}_i(k+1) = \omega . \vec{v}_i(k) + c_1 . \vec{r}_1(k)\left[\vec{p}_i^{best}(k) - \vec{x}_i(k)\right]$$
$$+ c_1 . \vec{r}_2(k)[\vec{g}^{best}(k) - \vec{x}_i(k)] \qquad Eq. (9)$$

$$\vec{x}_i(k+1) = \vec{x}_i(k) + \vec{v}_i(k+1) \qquad Eq. (10)$$

where ω, c_1 and c_2 are positive constants; $\vec{r}_1(k)$ and $\vec{r}_2(k)$ are vectors of random numbers from $U(0,1)$ in the $(k)^{th}$ iteration.

Figure 4 shows the flow chart of the particle swarm optimisation algorithm. The main steps can be summarized as follows.

(1) Initialisation: the positions and velocities of the particles are initialised with random values in the n-dimensions of the search space.

(2) Fitness calculation: the value of cost function is calculated for each particle

(3) Find p^{best} of each particle: the fitness of each particle is compared with its previous personal best fitness (p^{best}) ; if the new fitness value is smaller than the previous p^{best} , the new value is considered as the new $\mathrm{p}^{\mathrm{best}}$; otherwise the personal best will remain unchanged.

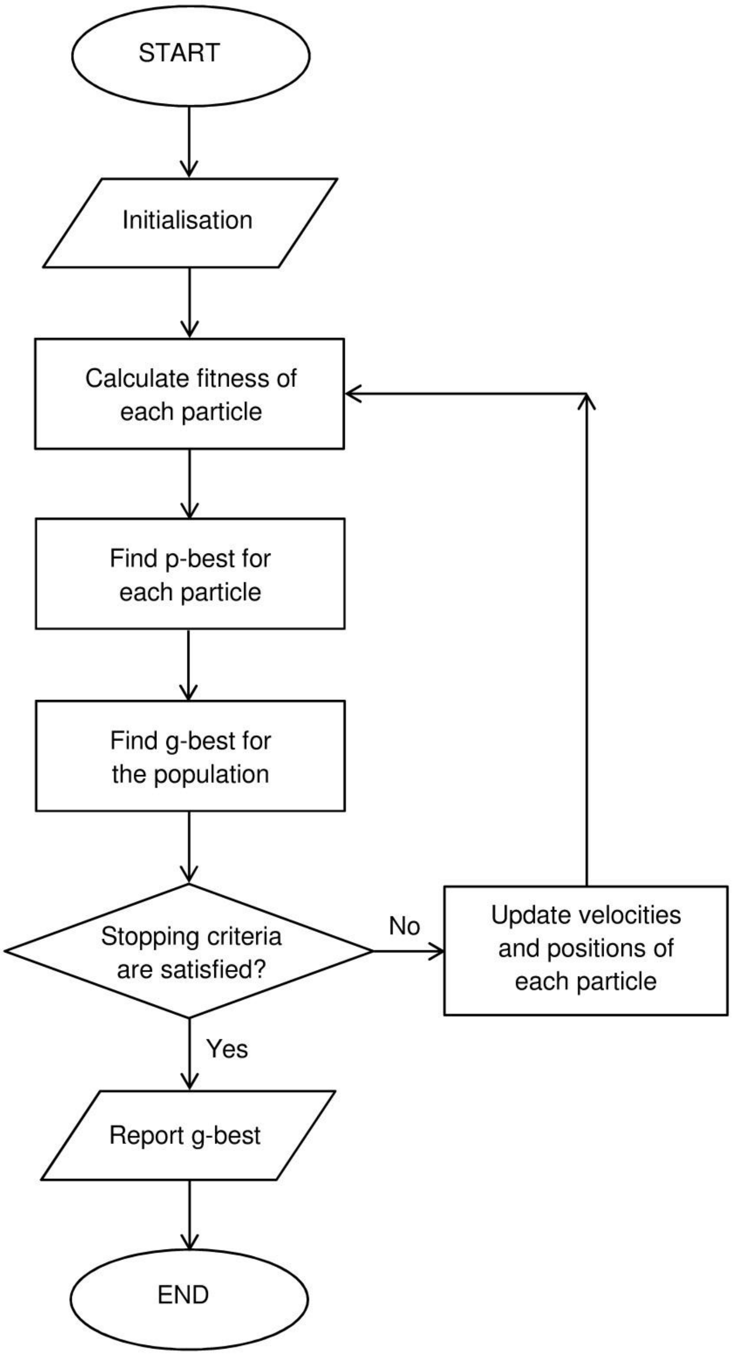
(4) Find g^{best} of the population: the p^{best} of all particles in the population are compared to each other. The smallest value is considered as the new global best (g^{best}).

(5) Check the stopping criteria: if the stopping criteria are satisfied, go to (7); else go to (6).

(6) Change velocities and positions: the velocity and position of each particle are updated by Equation 9 and Equation 10 respectively. Then go to (2).

(7) Report g^{best} : the best fitness which has found so far, is reported as the optimum value and the algorithm stops.

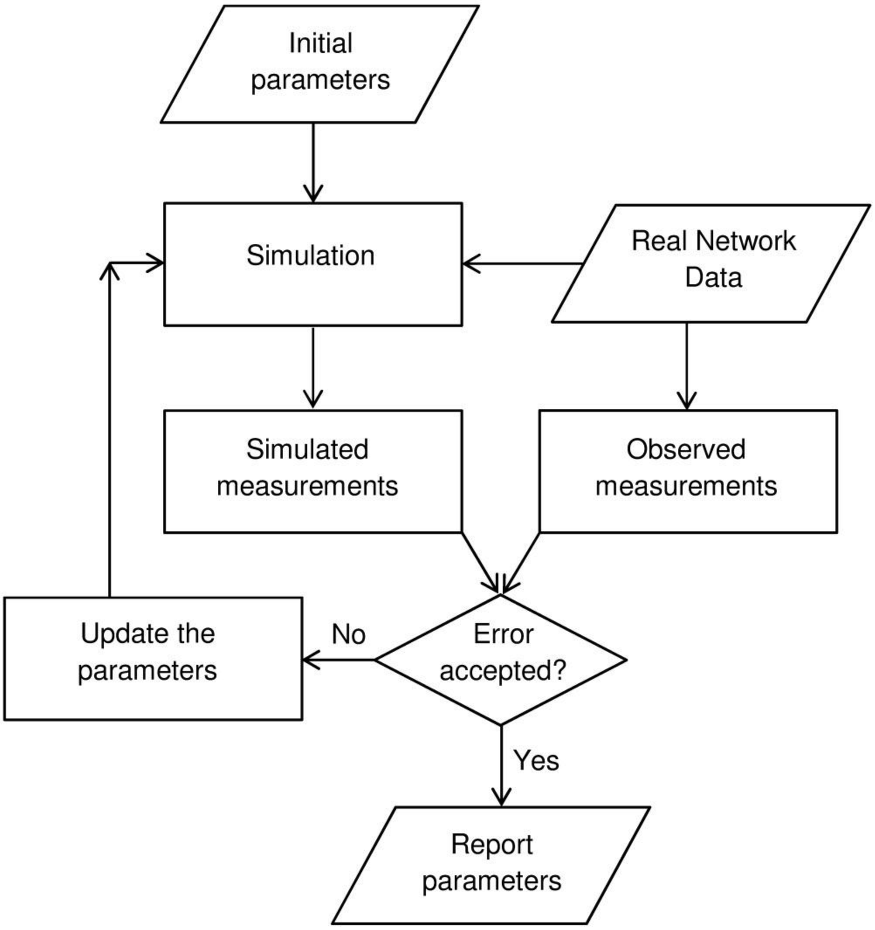
Figure 4: Flow chart of PSO algorithm



4. Implementation and Parallel Optimisation

Calibration is referred to the adjustment of model parameters enhancing a model's capacity to replicate driving behaviour and traffic characteristics. Figure 5 shows a general framework which could be used for calibration. According to this framework the calibration process is an iterative process by which the outputs of traffic micro-simulation become close enough to the real world measurements. Indeed, the calibration process can be considered as an optimisation problem which seeks to minimize a measure of the deviation between observed and corresponding simulated measurements.

Figure 5: General framework of calibration procedure



The simulated traffic measurements (STM) are a function of parameters associated with driving behaviour(α), route choice (β) and Origin-Destination (O-D) flows (γ); because it is almost impossible to segregate the impacts of driving behaviour, route choice and O-D flows in the total error. Therefore, the simulated traffic measurements (STM) can be expressed as:

$$STM = f(\alpha, \beta, \gamma) \qquad Eq. (11)$$

where f is the simulation process with parameter set of α, β and γ.

Now, the optimisation problem can be written as follow.

$$\min_{\alpha,\beta,\gamma} \Delta(STM, OTM) \qquad Eq. (12)$$

where Δ is a function that determines the difference between the simulated traffic measurements (STM) and the observed traffic measurements (OTM) and so-called error. The optimisation problem is thus defined to find α, β and γ by which the error is minimised.

The optimisation problem can be solved by the PSO algorithm as explained in Section 3. However, when the PSO algorithm is implemented for calibration, it is required to run the simulation once for each particle to calculate its fitness. Therefore, the run time of the calibration algorithm will be prolonged due to the long running time of simulation. To overcome this problem, multithread technique can be applied and a program can be written (e.g. in C++) using the COM Interface of VISSIM to share this time consuming task with the other CPUs. Based on the licence, VISSIM may allow theatrically opening unlimited instances for simultaneous runs. However, the number of instances is limited by the number of available CPUs.

The basic idea behind most parallel programs is to divide a large problem into smaller tasks; the tasks are solved simultaneously on multiple processors. When a multi-threaded program runs on a multi-core system, the operating system can provide each thread with one CPU. Therefore the threads can run on separate CPUs simultaneously which results in parallel execution or parallelisation. This technique can reduce the run time significantly. The deduction time depends on the number of available CPUs. Below the algorithm of this technique is presented.

The parallel PSO can be used to speed up the calibration procedure. Therefore, the calibration algorithm is modified as follows.

(1) Initialisation: positions and velocities of particles are generated randomly similar to original PSO. Several threads are initialised to calculate the population fitness. The number of threads should be equal or less than the number of available CPUs. Assume the number of threads is n and the population has m particles by which $d = m/n$. To increase the efficiency of the program, it is recommended to determine the number of particles such that d becomes an integer number.

(2) Parallel tasks:

(2-a) Task of each thread: after initialisation of the n threads, one particle is assigned to each of them. In each thread the assigned particle fitness is calculated through the running of the simulation. Then the personal best position and fitness of the particle is updated. The micro-simulation run and the fitness calculation of the n particles are performed in parallel and simultaneously.

(2-b) Repeating the task for d times: when the tasks of all threads are finished, another set of n particles from the remaining population are assigned to new n threads; the particles' fitness and their personal best positions are updated. This iterative procedure repeats d times.

(3) Joint tasks

(3-a) Updating the global best: when all particles' fitness and their personal best positions in the specific population are determined, the global best of the algorithm is updated.

(3-b) Checking the stopping criteria: if the stopping criteria are satisfied, the algorithm will stop and report the global best as the optimum answer, else, the positions and velocities of the particles are updated and the next iteration will start by going to Task 2.

As it can be seen, the time consuming step of calibration process which is the calculation of each particle's fitness, is implemented in parallel. Therefore, it can reduce the execution time of the algorithm significantly.

5. Conclusion and Future Work

This study provided a methodology to improve and facilitate the calibration procedure of traffic micro-simulations. More specifically, it developed an approach for auto-tuning of VISSIM as one of the commercially available traffic micro-simulations. It investigated the parameters within the simulation which may affect the simulation outputs and thus are potentially important to be considered for calibration. Car-following and lane changing models as the two fundamental components of any traffic micro-simulation and their associate parameters were discussed in details.

The calibration process of the traffic micro-simulation was considered as an optimisation problem that should be solved. The parameters should be found by which the deference between the observed traffic measurements and the corresponding simulated traffic measurements is minimised. Several measurements might be considered for this purpose such as travel time and traffic volume. The Particle swarm optimisation (PSO) algorithm was used to solve the optimisation problem and tune the VISSIM parameters automatically. To reduce the run time of the algorithm, a methodology was used to implement the PSO approach in parallel. Detailed explanations about the PSO algorithm, parallelisation and their implementation for auto calibration of VISSIM were presented in this study. The methodology developed here could be of interest to the traffic simulation software developers as well as their users as its application is not limited to VISSIM. This method could be used in any traffic simulations to produce accessory software facilitating the calibration of that specific simulation.

Acknowledgment

The authors would like to acknowledge the support received from the PTV AG from Germany and the Monash University Postgraduate Publication Award to work on this paper.

References

Boyd, S. & Vandenberghe, L. (2004), *Convex optimization*, Cambridge university press, Cambridge, United Kingdom.

Chu, L., Liu, H., Oh, J., & Recker, W. (2003), ‘A calibration procedure for microscopic traffic simulation’, Proceedings IEEE Conference in Intelligent Transportation Systems, vol. 2, pp. 1574-1579.

Engelbrecht, A. P. (2007), *Computational Intelligence: An Introduction*, John Wiley and Sons, Appendix A, Optimization Theory, pp. 551-579.

Gardes, Y., May, A., Dahlgren, J., & Skabardonis, A. (2002), ‘Freeway Calibration and Applications of Paramics Model’, *Proceedings of 81st Annual Meeting of the Transportation Research Board*, Washington, D.C., USA.

Holland, J. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Michigan, ISBN 0-262-58111-6.

Kennedy, J. & Eberhart, R. (1995), Particle Swarm Optimization, *Proceedings of IEEE International Conference on Neural Networks*, pp. 1942–1948.

Kennedy, J. & Eberhart, R. (2001), *Swarm Intelligence*, Academic Press, San Diego, CA, USA.

Komori, K., Homma, K. & Tsubone, T. (2012), 'Basic Study on Particle Swarm Optimization with Hierarchical Structure for Constrained Optimization Problems', *Neural Information Processing Lecture Notes in Computer Science*, Springer, vol. 7665, pp. 545-551.

Lee, D., Yang, X., & Chandrasekar, P. (2000), 'Parameter Calibration for PARAMICS Using Genetic Algorithm', *Proceedings of 80th Annual Meeting of the Transportation Research Board*, Washington, D.C., USA.

Ma, T., & Abdulhai, B. (2002), 'Genetic algorithm-based optimization approach and generic tool for calibrating traffic microscopic simulation parameters', *Transportation Research Record*: Journal of the Transportation Research Board, vol.1800, no.1, pp. 6-15.

Menneni, S, Sun, C, & Vortisch, P. (2008), 'Microsimulation calibration using speed-flow relationships', *Transportation Research Record*: Journal of the Transportation Research Board, vol.2088, no.1, pp. 1-9.

Moridpour, S., Sarvi, M., Rose, G., & Mazloumi, E. (2012), 'Lane-Changing Decision Model for Heavy Vehicle Drivers', *Journal of Intelligent Transportation Systems*, vol.16, no.1, pp. 24-35.

Narushimaa, Y., & Yabeb, H. (2012), 'Conjugate gradient methods based on secant conditions that generate descent search directions for unconstrained optimization', *Journal of Computational and Applied Mathematics*, vol. 236, no. 17, pp. 4303–4317.

Park, B., & Qi, H. (2005), 'Development and Evaluation of a Procedure for the Calibration of Simulation Models', *Transportation Research Record*: Journal of the Transportation Research Board, vol.1934, no.1, pp. 208-217.

Park, B., & Schneeberger, J. (2003), 'Microscopic Simulation Model Calibration and validation: case study of VISSIM simulation model for a coordinated actuated Signal system', Transportation Research Record: Journal of the Transportation Research Board, vol.1856, no.1, pp.185-192.

Polyak, B.T. (2007), Newton's method and its use in optimization, *European Journal of Operational Research*, vol. 181, no. 3, pp. 1086–1096.

Rockafellar, R. T. (1993), 'Lagrange multipliers and optimality', *SIAM Review*, vol. 35, no. 2, pp. 183-238.

Sedlaczek, k. & Eberhard, P. (2006), 'Using augmented Lagrangian particle swarm optimization for constrained problems in engineering', *Structural and Multidisciplinary Optimization*, vol. 32, no. 4, pp. 277-286.

Snyman, J. A. (2005), *Practical Mathematical Optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms*, Springer, Cambridge, Massachusetts, ISBN 0-387-24348-8.

Szu, H. H. (1986), 'Non-Convex Optimization', *Proceedings of the SPIE, Real-Time Signal Processing IX*, vol. 698, pp. 59-65.

Toledo, T., Ben-Akiva, M., Darda, D., Jha, M., & Koutsopoulos, H. (2004), 'Calibration of microscopic traffic simulation models with aggregate data', *Transportation Research Record*: Journal of the Transportation Research Board, vol.1876, no.1, pp.10-19.

VISSIM (2012), *VISSIM 5.40-01, User Manual*, Planung Transport Verkehr AG, Karlsruhe, Germany.

Wang, C. Y. & Li, D. (2009), 'Unified theory of augmented Lagrangian methods for constrained global optimization', *Journal of Global Optimization*, vol. 44, no. 3, pp. 433-458.

Wiedemann, R. (1974), '*Simulation des Strassenverkehrsflusses*', Technical report, Institut fur Verkehrswesen, Universitat Karlsruhe, Karlsruhe, Germany, In German.

Wiedemann, R. & Reiter, U. (1992), '*Microscopic traffic simulation: the simulation system MISSION*', CEC Project ICARUS (V1052), Final Report.

Zheng, Y. & Wan, Z. (2012), 'A new variant of the memory gradient method for unconstrained optimization', *Optimization Letters*, vol. 6, no. 8, pp. 1643-1655.