Parallel Computing-based Calibration for Microscopic Traffic Simulation Model

Lanyue Tang, Master Student
Department of Transportation Engineering, Tongji University Shanghai, China, 201804
Email: 2133393@tongji.edu.cn

Duo Zhang, PhD Candidate
Department of transportation engineering, Tongji University Shanghai, China, 201804
Email: zhangduo@tongji.edu.cn

Yu Han, Associate Professor
Department of transportation engineering, Southeast University
School of Transportation, Southeast University, Nanjing, China, 211189
Tel: +86-13718047574, E-mail: yuhan@seu.edu.cn

Auhui Fu, Master Student
Department of Transportation Engineering, Tongji University Shanghai, China, 201804
Email: 2133397@tongji.edu.cn

He Zhang, PhD Candidate
Department of transportation engineering, Tongji University
Email: zhanghe_e@126.com

Ye Tian, Associate Professor
Department of Transportation Engineering, Tongji University Shanghai, China, 201804
E-mail: tianye@tongji.edu.cn

Lishengsa Yue, Assistant Professor
The Key Laboratory of Road and Traffic Engineering, Ministry of Education
College of Transportation Engineering, Tongji University
4800 Cao'an Road, Shanghai 201804, P.R of China
E-mail: 2014yuelishengsa@tongji.edu.cn

Di Wang, Senior Engineer
SAIC Motor Corporation Limited, Shanghai, China, 201804
E-mail: wangdi01@saicmotor.com

Jian Sun, **Corresponding Author,** Professor
Department of Transportation Engineering, Tongji University Shanghai, China, 201804
E-mail: sunjian@tongji.edu.cn

ABSTRACT

Microscopic traffic simulation is vital to assess the performances of various traffic operation and management schemes. Microscopic traffic simulation is usually not parameter-free, and it relies on independent parameters to predict traffic evolution. Thus, parameter calibration is indispensable to conveying trustworthy simulation results. Heuristic algorithms are widely used for parameter calibration. Its logic is for achieving iterative optimization through continuous trial-and-error simulations. This process is time-consuming and usually takes several hours, making the calibration unable to meet the requirements of fast and efficient. In recent years, Parallel Computing Technology (PCT) has been gradually applied in the engineering realm, which makes rapid calibration possible. Following the three steps of parallel framework selection, algorithm bottleneck identification, and subtask load balancing, this paper designs and implements the parallelization of Genetic algorithm (GA) and Particle Swarm Optimization (PSO) calibration algorithms. Finally, the proposed parallel framework is applied to simulation parameter calibration of a section of a 5 km long highway in Australia, and the effectiveness of parallel computing is evaluated from the two dimensions of reduction in calibration computational time and scalability. The results show that the proposed parallel calibration algorithm can shorten the 5-hour calibration process to less than 1 hour, reducing the calibration time by 80%. The parallel PSO calibration algorithm has better scalability, and its acceleration effect is better when more processors are used.

Keywords: model calibration, traffic simulation, parallel computing, genetic algorithm, particle swarm optimization.

INTRODUCTION

Micro-traffic simulation has become a necessary technical support tool to assess and predict the performances of various traffic operation and management schemes. In the microscopic traffic simulation model, the driver behavior, the characteristics of traffic flow, and the operation of the traffic system are all described by the calculus of many internal independent parameters, so the ability of the simulation model to reproduce the actual traffic flow mainly depends on the value of the parameters. Micro-traffic simulation platforms often set default values of parameters based on the traffic flow characteristics of the country where the platform is developed. However, these default values often do not match with the specific application scenarios, resulting in low simulation accuracy. Therefore, in practical applications, parameter calibration has become a prerequisite for all subsequent works.

The parameter calibration of microscopic traffic simulation is essentially a combined optimization problem with multiple objectives. The trial-and-error method, proxy models, and heuristic algorithms have been used to solve this essential engineering problem. Gardes et al. (*1*) and Gomes et al. (*2*) used the trial-and-error method to calibrate the PARAMICS and VISSIM simulation models. This kind of method is only suitable for small-scale road networks. The calibration process is highly subjective, resulting in low efficiency.

Some researchers used the proxy model to calibrate the parameters. Osorio et al. (*3*) proposed a simulation-based meta-model calibration method. This method used the structural information of the problem to establish an analytical approximation of the input-output mapping between the calibration parameters and the simulation output. Ištoka Otkovic et al.(*4*) applied the neural network to the parameter calibration of microscopic traffic simulation. They used the neural network to replace the process of simulation evaluation. These methods are currently often used for the simulation and calibration of large and complex networks (*5; 6*).

At present, compared with other methods, heuristic algorithms such as GA and PSO have been more widely used. Siddharth (*7*) et al. adopted Elementary Effects to perform sensitivity analysis and realized simulation parameter calibration based on GA. Focusing on heuristic optimization algorithms, Abdalhaq et al. (*8*) compared the calibration effects of GA, Tabu Search (TS), PSO, and Simultaneous Perturbation Stochastic Approximation (SPSA) based on the SUMO. Experimental results showed that the SPSA algorithm had certain limitations in terms of parallelization, while PSO could be highly parallelized, and thus they had good potential in solving high-latitude problems. The application of the heuristic algorithm, which derives the optimal combination through iterative evolution, speeds up the process of parameter calibration to a certain extent.

However, the heuristic algorithm realizes iterative optimization through continuous trial-and-error simulation, which is very time-consuming, making the calibration based on the heuristic algorithm unable to meet the requirements of time cost and efficiency. To make the accuracy of the simulation model reach an acceptable level, it still takes several hours(*7-9*).

In recent years, with the popularity of multi-core computers, Parallel Computing Technology (PCT) has gradually become a research hotspot in the computer field. Parallel computing divides the problem into subtasks and solves them on multiple processors at the same time. It has been successfully applied in fields such as big data mining and the modeling of complex phenomena(*10-13*), which improves the computational efficiency of programs.

There have been researches on applying PCT to parameter calibration in recent years. Hou et al.(*9*) proposed a multi-threaded optimization Quasi-monte carlo Particle Swarm (QPS) calibration method, realizing parallel computing. However, their research focused on the application of Quasi-Monte Carlo (QMC) sampling. The proposed QMC is an efficient and robust method that allows modelers to filter out redundant samples and find the global optimum in time. Dadashzadeh et al. (*14*) applied PCT to the heuristic algorithm by realizing the parallel of loop sentences, and then developed a fast calibration strategy.

However, the studies so far still have the following problems. First, these studies only realized simple applications of parallel computing, without consideration of the entire algorithm design from the perspective of parallel computing. Secondly, researchers only focused on the reduction of calibration computational time when evaluating the efficiency improvement of parallel computing. In fact, when computing resources increase, different parallel computing algorithms obtain different acceleration capabilities, and this ability to utilize the increased computing resources is named scalability. A comprehensive parallel computing efficiency evaluation system that takes into account the reduction of computing time and the scalability of the algorithm itself has been missing for a long time.

Therefore, this paper applies PCT to the parameter calibration of microscopic traffic simulation. To make up for the shortcomings of the current research, as summarized above, our work is as follows:

- Following three steps of parallel framework selection, computing bottleneck identification, and load balancing design, the parallelization of the GA and PSO are designed and implemented to realize parameter calibration.
- A SUMO simulation model containing high-density traffic flow is used, and the established parallel algorithms are applied to the simulation model for verification, which proves the effectiveness of the algorithms in terms of accuracy and efficiency.
- The performance of parallel computing on the two calibration algorithms is evaluated and compared from both dimensions of calibration computational time and scalability.

The rest of this article is organized as follows. In the part of problem formulation, the mathematical problem behind the calibration of micro-traffic simulation models is described in detail from the perspective of objective functions and the parameter space for optimization. The methodology part introduces the overall framework of the proposed parameter calibration method using PCT. In the case study section, based on the SUMO simulation platform, the proposed algorithms are used to calibrate the traffic flow of a section of an expressway in Australia. The findings and conclusions of the experiment are discussed in the section of the conclusion.

CALIBRATION PROBLEM FORMULATION

objective function

The calibration of micro traffic simulation is essentially a combinatorial optimization problem with objective functions, and the parameters are constrained. The ultimate goal is to minimize the difference between the simulated traffic flow and the actual traffic flow. In order to make the traffic flow velocity of the simulation model achieve high accuracy, and at the same time simulate the formation and dissipation of bottlenecks in the actual traffic flow as much as possible, this paper refers to the research of Song et al. (*15*) . The $RMSE_{speed}$ （root mean square error of the speed） and the bottleneck range matching index C_1 are simultaneously adopted as the objective functions of parameter calibration, and the specific formulas are as follows:

Minimize $$RMSE_{speed} = \sqrt{\frac{\sum_{i=1}^{N_{detector}} \sum_{t=1}^{T} (Sv(i,t) - Rv(i,t))^2}{N_{detector} \times T}} \quad (1)$$

Maximize $$C_1 = \frac{2\sum_{i=1}^{N}\{(\sum_{t=1}^{T}[BS_S(i,t) \wedge BS_R(i,t)]) \cdot (l_{i+} - l_i)\}}{\sum_{i=1}^{N}\{(\sum_{t=1}^{T}[BS_S(i,t) \vee BS_R(i,t)]) \cdot (l_{i+} - l_i)\}} \quad (2)$$

Restriction $$X = \{x_1, x_2, \dots, x_j\}\ j = \{1, 2, \dots, K\},\ Lb_{x_j} \le x_j \le Ub_{x_j} \quad (3)$$

In the formula, X represents the parameter vector. Lb_{x_j}, Ub_{x_j} are the lower and upper limits of parameter $x_j (j = \{1, 2, \dots K\})$. $Sv(i,t)$ is the speed collected by the i-th $(i = \{1, 2, \dots N_{detector}\})$ detector of the simulation model in the t-th $(t = \{1, 2, \dots T\})$ time period, while the speed collected by the actual field detectors is denoted as $Rv(i,t)$. T is the number of time periods during which the detectors collect data (simulation duration is $120\,min$, collected every $2\,min$, $T = 120/2 = 60$). Furthermore, $N_{detector}$ denotes the total number of coil detectors.

C_1 is a quantitative index that reflects the matching degree of the actual and simulated traffic flow's spatio-temporal bottleneck range. First, the binary speed space-time maps of the simulated and actual traffic flow need to be calculated and generated, which represent the binary form of the average speed value of the ith detector at the tth time period :

$$\text{if } Sv(i,t) < V_{th},\ BS_S(i,t) = 1,\ \text{else } BS_S(i,t) = 0 \quad (4)$$
$$\text{if } Rv(i,t) < V_{th},\ BS_R(i,t) = 1,\ \text{else } BS_R(i,t) = 0 \quad (5)$$

In the formula: V_{th} is the congestion threshold. When the speed value is less than V_{th}, it is considered that there is congestion on the expressway at this spatio-temporal point, which is 45 km/h in our study.

In formula (2), $BS_S(i,t) \wedge BS_R(i,t)$ represents the intersection of matrices, which is calculated mathematically by multiplying the internal elements of two matrices, while $BS_S(i,t) \vee BS_R(i,t)$ represents the union. When $C_1 = 1$, it means that the actual and simulated traffic flow bottleneck areas are completely matched. $l_{i+} - l_i$ represents the distance between two coils.

Parameter space for optimization

The driving behavior parameters of the SUMO simulation software are mainly divided into three modules, including car following model, lane changing model, and speed distribution module. In this paper, the Intelligent Driver Model (IDM) was selected as the car following model, which was first proposed by Treiber et al. (*16*) based on the generalized force model. IDM has few parameters which are present clear meaning. This model has been widely used, and it often produces consistent results with empirical observations. There are two lane changing models of SL2015 (*17*) and CL2013 (*18*) in SUMO simulation platform. SL2015 is a sub-lane model, which is only applicable to the simulation scenarios of lane splitting. Therefore, the CL2013 lane change model for lane selection was adopted in this paper.

In SUMO, the driving speed of the simulated vehicles is modeled by assigning to each vehicle an individual multiplier which gets applied to the road speed limit. This multiplier is called the "*individual speedfactor*". The product of the road speed limit and the individual speed factor gives the desired free flow driving speed of a vehicle. The expected value of the "*individual speedfactor*" is denoted as "*SpeedFactor*", while, the "*SpeedDev*" represents the deviation of the "*SpeedFactor*". These two parameters form a speed distribution module that controls the sampling of vehicle speed. The specific meanings of the two parameters are shown in **Table 1**.

TABLE 1 Parameters of the speed distribution module

Parameter names	Definition	default value
SpeedFactor	The vehicles' expected multiplier for lane speed limits	1.0
SpeedDev	The deviation of the SpeedFactor	0.1

We followed several principles to determine the ranges of the parameters to be calibrated. First of all, the ranges of parameters must comply with the basic regulations of sumo's official documents. Secondly, constraints between parameters needed to be considered. For example, from common sense, the value of emergency deceleration would be larger than that of normal deceleration. Then the ranges of parameters with similar meanings in previous studies also needed to be referred to. Finally, we also refer to the default values given in SUMO, making the ranges around the default values.

After an initial selection, 19 parameters were used for calibration, far exceeding the expected number of calibration parameters. Too many calibration parameters would lead to high dimensionality and computational complexity of the parameter calibration problem. Therefore, in order to save computing resources and reduce the dimension of the optimization problem, avoiding the trap of local optimal solution, we used the sobol sensitivity analysis (*19*) to realize the selection of key parameters.

Sobol sensitivity analysis is based on the basic assumption that variance can well represent the uncertainty of the model's output. This method uses the sobol sequence for sampling, which can be divided into two steps: the value acquisition using sobol sequence and the sample generation.

Sobol sequence is essentially a quasi-random low-discrepancy sequence used to generate uniform samples of parameter space, which is constructed by linear recurrence relations in finite fields. In this paper, we generated a basic sobol sequence matrix $\mathrm{M}_{\mathrm{basic}}$ of size *(N_{sample}+skipvalue, 2D)* based on the sobol sequence generator proposed by Frances et al. (*20*). N_{sample} is the number of sample points taken, and D is the number of random variables for sensitivity analysis. Since the initial points of the sobol sequence had some repetitions, *skipvalue*, a positive integer, was set to skip these points in the previous part. Therefore, starting from the *skipvalue+1* bit of the basic sobol sequence, a basic sobol sequence matrix M_{sobol} of size *(N_{sample}, 2D)* was generated. We scaled the sequence value to the sample space formed by the thresholds of all parameters according to a uniform distribution, then obtained a sample parameter matrix M_{para} with N_{sample} rows and *2D* columns.

Latin Hypercube Sampling (LHS) was first described by Michael McKay(21) in 1979, which is a statistical method for generating near-random samples of parameters from multidimensional distributions. The sampling process of LHS consists of two parts: interval sampling and sample sorting. LHS firstly stratifies the sample space according to the cumulative probability density curves of random variables, then conducts sampling in each sample space. Finally, LHS uses the sorting algorithm to make the correlation between random variables of the sampling result consistent with the one between the real random variables.

We took the sampling in a one-dimensional sample space as an example. With 300 sample points generated in the $X \in [0, 1]$ interval, the results of sobol sequence sampling are compared with those generated by LHS which is commonly used, as shown in **Figures 1 and 2**. The horizontal axis represents the serial number of the sample point. A total of 300 sample points were generated. The vertical axis is the value of the sample point. As can be seen from the figure, sample points generated by sobol sequence sampling are more evenly distributed, with wide coverage and better diversity.

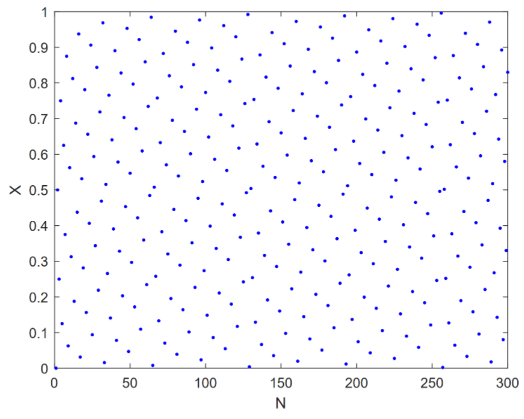


FIGURE 1 Sobol Sampling

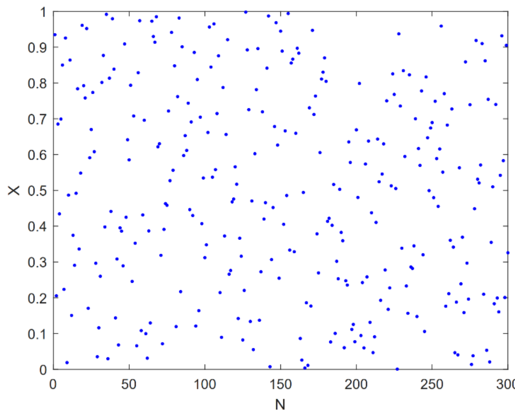


FIGURE 2 LHS Sampling

After the sample parameter matrix M_{para} with N_{sample} rows and *2D* columns was obtained, the sample matrixes for sobol sensitivity analysis needed to be generated. Took the first *D* column of the matrix M_{para} as matrix *A*, and the last *D* column as matrix *B*. Then, we replaced the *d* th column of matrix *A* with the *d* th column in matrix *B* to construct the matrix with N_{sample} rows and *D* columns AB^d *(d=1, 2, 3, ..., D)*. So far, a total of *D+2* sample matrices of *A, B,* AB^1, AB^2, ...AB^D had been constructed for sobol sensitivity analysis. The obtained sample matrixes were put into the simulation model to get $Y_A, Y_B, Y_{\mathrm{AB}^1}, Y_{\mathrm{AB}^2}, \dots, Y_{\mathrm{AB}^D}$. In this paper, each value of the *Y* matrix was the first objective function value $RMSE_{speed}$ (**formula 1**) obtained after the corresponding parameter combination was simulated. The global sensitivity index ST_d of dth parameter was calculated according to the following formula:

$$ST_d = \frac{E_{X_d}\left(\mathrm{Var}_{X_d}(Y|X_d)\right)}{\mathrm{Var}\,(Y)} \tag{6}$$

Within the formula:

$$E_{X_d}\left(\mathrm{Var}_{X_d}(Y \mid X_d)\right) = \frac{1}{2N_{sample}} \sum_{m=1}^{N_{sample}} \left(f(\mathrm{A})_m - f(\mathrm{AB}^d)_m\right)^2 \tag{7}$$

$$\mathrm{Var}\,(Y) = \mathrm{Var}\,(Y_A + Y_B) \tag{8}$$

In the formula, $f(.)_m$ denotes the value in the mth row in the matrix $Y_{(.)}$, representing the $RMSE_{speed}$ (**formula 1**) obtained after the m th parameter combination in the corresponding sample matrix was simulated. Referring to previous studies (*22*), when the global sensitivity index ST_d of dth parameter is greater than 2%, this parameter can be considered as a key parameter and usually needs to be calibrated. Therefore, according to the results of the sobol sensitivity analysis, we sorted the global sensitivity index values of all parameters from largest to smallest, and selected the top 7 parameters as the key parameter set to be calibrated. **Table 2** shows the final selected calibration parameters and their value ranges.

TABLE 2 Calibration Parameters

Parameter names	**Definition**	**Total Sensitivity index ST**	**Range**
Tau	Minimum driving interval expected by the drivers	0.766	[1, 4]
SpeedFactor	The vehicles' expected multiplier for lane speed limits	0.431	[0, 1]
SpeedDev	The deviation of the speedFactor	0.372	[2, 9]
IcCooperative	Willingness of drivers to cooperate in lane changing. A lower value means less cooperation between vehicles.	0.275	[0, 0.5]
Accel	Acceleration capacity of vehicles	0.139	[0.5, 1.5]
Decel	Deceleration capacity of vehicles	0.097	[0.8, 1.5]
IcAssertive	Willingness to accept lower front and rear gaps on the target lane. The required gap is divided by this value.	0.049	[1, 7]

METHODOLOGY

In the methodology section, two GA and PSO, which are two classical heuristic algorithms widely used in calibration problems, are presented in detail. Then, the parallel frameworks of the two calibration algorithms are designed and implemented following three steps: the parallel framework selection, computing bottleneck identification, and load balancing design.

Genetic Algorithm and Particle Swarm Optimization

Genetic algorithm

Genetic algorithm (GA) is a non-deterministic quasi-natural algorithm, first proposed by Holland(*23*) in 1975. The GA first composes a certain number of genetically encoded individuals into the initial population. According to the principle of survival of the fittest, individuals are selected to generate new populations according to their fitness. The selected outstanding individuals are combined, crossed, and mutated to achieve individual reorganization. The above process continues to iterate and loop, producing a better approximate solution in the evolution.

The GA used in this article is as follows:

(1) The generation of the initial population. We randomly sample within the value range of each parameter to generate the initial population. Then the Gray coding is used to map the parameter sets from the solution space to the search space that the genetic algorithm can handle.

(2) Fitness function. In this paper, the Root Mean Squared Error $RMSE_{speed}$ (**formula 1**) of the speed and the index *C1* (**formula 2**) is used as the basis of the fitness function. We use exponential change to make it conform to the convention that the smaller the simulation error, the greater the adaptability.

(3) Selection. Based on the fitness values of different individuals, outstanding individuals are selected to produce offspring populations in accordance with the evolutionary principle. This paper adopts the method of tournament selection. We take a certain number of individuals from the population each time, using sampling with replacement, and then select the best one to enter the progeny population. Repeat this operation until the new population size reaches the original value.

(4) Crossover. Two individuals for evolution are randomly selected. The same position of the two carries out gene exchange according to the crossover probability P_c to produce a new individual.

(5) Mutation. The mutation operation in this paper is to invert the corresponding bit of the individual code according to the set mutation probability P_m.

Repeat the above process until individuals who meet the fitness value requirements are obtained. When designing and selecting the hyperparameters of GA, in order to ensure the fairness of the comparison, the population size of each generation was determined to be equal to the number of particles in each generation in PSO. We mainly considered the crossover probability P_c and the mutation probability P_m when adjusting the GA. An orthogonal experiment with two Factors and five Levels was designed for the two parameters in the GA, and 25 parameter combinations were generated. In order to avoid the influence of the randomness and random errors of the microscopic traffic simulation model on the test results, 5 groups of experiments were repeated for each parameter combination, and the mean values of the objective function values were compared. The following GA parameters have given the best results: crossover probability P_c = 0.8, mutation probability P_m = 0.01. Therefore, we have adopted this hyperparameter combination in GA in subsequent experiments.

Particle swarm optimization

The PSO algorithm is a swarm intelligence algorithm proposed by Eberhart et al. (*24*) in 1995. The algorithm regards individuals as random particles, and guides particles to move to the optimal solution through the local and global optima.

We used the PSO algorithm with M particles to solve a problem with a search space of dimension K. In $x_{i,n}^{j}$, n, i, j respectively represent, in the nth $(1 \leq n \leq N)$ iteration, the position in $jth\,(1 \leq j \leq K)$ dimension of the $ith\,(1 \leq i \leq M)$ particle. Then in the nth

iteration of PSO, the current position and current velocity of the ith $(1 \le i \le M)$ particle can be denoted as $X_{i,n} = \left(x_{i,n}^1, x_{i,n}^2, \cdots, x_{i,n}^j, \cdots, x_{i,n}^K\right)$,and $V_{i,n} = \left(V_{i,n}^1, V_{i,n}^2, \cdots, V_{i,n}^j, \cdots, V_{i,n}^K\right)$. In the nth iteration, the velocity and position of the ith particle in the jth dimension component are updated using the following formulas:

$$V_{i,n+1}^j \leftarrow wV_{i,n}^j + c_1 r_{i,n}^j\left(pbest_{i,n}^j - x_{i,n}^j\right) + c_2 R_{i,n}^j\left(gbest_n^j - x_{i,n}^j\right) \tag{9}$$

$$x_{i,n+1}^j \leftarrow x_{i,n}^j + V_{i,n+1}^j \tag{10}$$

$$i = 1,2,\dots,M; j = 1,2,\dots,K; n = 1,2,\dots,N$$

In the formula, c_1 and c_2 are acceleration factors of the self-learning and the social learning respectively, which are used to adjust the convergence speed of the algorithm. The vector

$$pbest_{i,n} = \left(pbest_{i,n}^1, pbest_{i,n}^2, \cdots, pbest_{i,n}^j, \cdots, pbest_{i,n}^K\right)$$

is the position with the best fitness value of the ith particle from initialization to the nth iteration, which also known as the individual optimal position. After each iteration, the $pbest_{i,n}$ of each particle is updated according to the following formula:

$$pbest_{i,n+1} = \begin{cases} pbest_{i,n}, & f\left(pbest_{i,n}\right) < f\left(X_{i,n+1}\right) \\ X_{i,n+1}, & f\left(pbest_{i,n}\right) \ge f\left(X_{i,n+1}\right) \end{cases} \tag{11}$$

f(.) is the objective function to obtain the fitness value of the corresponding position. The vector

$$gbest_n = \left(gbest_n^1, gbest_n^2, \cdots, gbest_n^j, \cdots, gbest_n^K\right)$$

in **Equation (9)** is the position with the best fitness value among all $pbest_{i,n}$ until the nth iteration, $gbest_n$ is also called the global optimal position. $r_{i,n}^j$ and $R_{i,n}^j$ are random numbers that satisfy the uniform distribution from 0 to 1, denoted as $r_{i,n}^j, R_{i,n}^j \sim U(0,1)$, which are set to raise the randomness of the search, running iteratively. They are updated as each particle in each generation of the population moves through each dimension. So, for the ith particle of the nth generation swarm, when it is updated in the jth dimension, the random items are denoted as $r_{i,n}^j, R_{i,n}^j$.

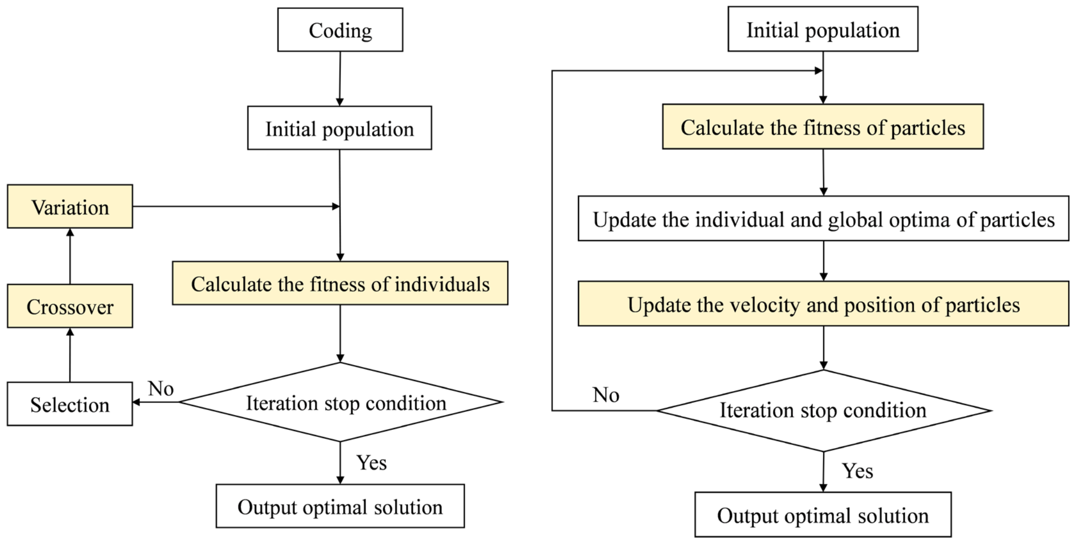
Acceleration coefficients C_1, C_2 represent the weighting of the stochastic acceleration terms that pull particles towards pbest and gbest. *When* C_1*=0, the algorithm converges quickly,* but the algorithm's local search ability is poor, which means that the algorithm is easy to fall into the dilemma of local optimization. On the contrary, when C_2=0, the algorithm only has the ability of self-learning, which will cause the algorithm to converge slowly and fail to find the global optimal solution. J. Kennedy et al. suggested statically setting the values of C_1 and C_2 to 2 (24). Since then, many authors have followed this advice in their PSO studies(25; 26). The current main purpose of this paper is to compare the difference in the application effect of PCT on GA and PSO. For the fairness of the comparison, the prototypes of the two algorithms both are standard forms. Therefore, in the selection of the acceleration coefficients, we refer to the common value of $C_1 = C_2 = 2$, without considering the more complicated mechanism of controlling the acceleration coefficients. The inertia weight w is used to control the change range of the particles. If w is low (e.g. 0.3-0.4) means that the system is more dissipative. When the w is high (e.g. 0.8-0.9), the particles would move in a medium of low viscosity and do an extensive exploration of the space of parameters. While a larger value of w makes the PSO algorithm may have better global search capabilities to avoid falling into the trap of local optima. The parameter w can also be set larger than 1, however, this is not advisable as the swarm would turn out to be unstable. Through multiple experiments, Shi et al. (27) suggested that the inertia weight w should take a value in the interval $[0.8, 1.2]$. We tried several common values and finally found that when $w = 0.9$, the results of the PSO calibration algorithm were relatively better.

Parallel Computing Design

The selection of parallel framework

The selection of parallel architecture needs to consider the specific characteristics of the problem to be solved. Problems suitable for parallel computing technology often have the following salient features:

- The problem can be decomposed into discrete pieces that can be executed concurrently;
- The different discrete fragments obtained by decomposition have no requirement on the execution order, which means that they are independent fragments.



(a) GA **(b) PSO**

FIGURE 3 The framework of the calibration algorithm

The genetic algorithm calibration process is shown in **Figure 3**(a). The order of execution of these main steps is fixed. The main steps are dependent on each other seriously, meaning that they cannot be parallelized. However, the three steps of running simulation to calculate fitness, crossover, and variation can be decomposed into independent discrete segments that can be executed concurrently.

The main steps of the PSO calibration algorithm are shown in **Figure 3**(b) There is also a strong dependency between the four main parts. The steps of updating the individual and global optima require a lot of communication between particles. Therefore, the two steps of evaluating particle fitness and updating particle position and velocity can be divided into discrete segments that could be executed by PCT.

The computer framework can be divided into four categories from the two dimensions of instruction flow and data flow: Single Instruction Single Data (SISD), Single Instruction Multiple Data (SIMD), Multiple Instruction Single Data (MISD), and Multiple Instructions Multiple Data (MIMD). SISD is a serial machine in the standard sense. The other three categories belong to the framework of parallel computing in terms of their operation logic. In the MISD architecture, different processing units can independently execute different instruction streams, but they receive the same single data stream. The SIMD architecture is a typical parallel architecture. All processing units execute the same instruction in any clock cycle, and each processing unit processes different data elements. MIMD is a parallel computing framework in the highest sense, in which different processors can process different instruction streams and different data at the same time.

According to the above analysis of the two calibration algorithms, in the GA and PSO calibration algorithms, a certain main step can be divided into discrete segments that are executed concurrently. In these steps, the instructions executed by the discrete segments are the same, which need to be applied to different data, and the results are returned to the main thread for information aggregation. Therefore, in this study, SIMD is an applicable parallel framework.

The identification of calculation bottleneck

This step is to analyze and identify which part of the whole process has completed most of the work of the program through a program analyzer or performance analysis tool. Based on the results of the analysis, we made the parallel computing technology focus on these key points of the program while ignoring the rest that takes up a small amount of Central Processing Unit (CPU) computing resources.

The flame graph is a visualization tool that shows the proportion of CPU computing resources occupied by each function during the program operation. The flame graph is drawn based on the stack traces, which list all the functions being executed by the CPU at any given time. The flame graph shows the time distribution of program execution from a global perspective. Each column represents a call stack, and the bar is used to represent a specific function. The vertical axis arranges functions from bottom to top according to the calling relationship. On the horizontal axis, the flame graph aggregates many call stacks in alphabetical order, which does not represent a chronological order.

The length of each bar represents the frequency of occurrence of the function in the sample. Therefore, the longer the bar, the longer the execution time of the function, which is most likely to be the calculation bottleneck of the algorithm.

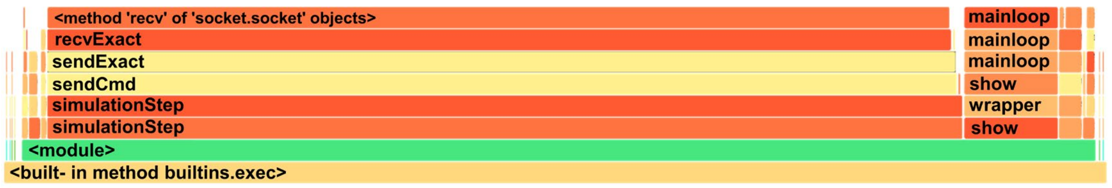


FIGURE 4 Flame Diagram of Single-thread GA Calibration *The bar represents a specific*

function. The vertical axis arranges functions from bottom to top according to the calling relationship. On the horizontal axis, the flame graph aggregates many call stacks in alphabetical order. The length of each bar represents the frequency of occurrence of the function in the sample. SimulationStep function of the GA calibration algorithm occupies most of the computing resources during the entire calibration process, which is extremely time-consuming.

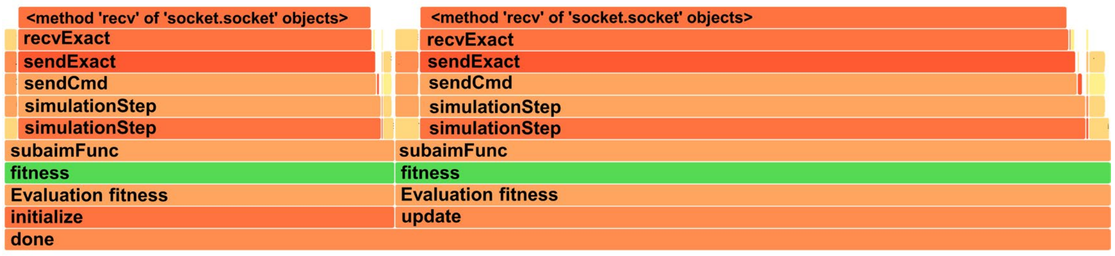


FIGURE 5 Flame Graph of Single-thread PSO Calibration *The bar represents a specific function. The vertical axis arranges functions from bottom to top according to the calling relationship. On the horizontal axis, the flame graph aggregates many call stacks in alphabetical order. The length of each bar represents the frequency of occurrence of the function in the sample. SimulationStep, initialize, update function of the PSO calibration algorithm occupies most of the computing resources during the entire calibration process.*

When both algorithms run for 2 generations and simulate 5 times for each generation, the flame diagrams of the GA calibration algorithm and the flame diagram of the PSO calibration algorithm are shown in **Figures 4 and 5**. It can be seen from the graphs that the SimulationStep function of the two calibration algorithms occupies most of the computing resources during the entire calibration process, which is extremely time-consuming. The SimulationStep function writes the generated parameter sets into the SUMO simulation platform and controls the simulation model by using Traffic Control Interface (TraCI). While the TraCI is a traffic control interface that realizes the interaction between SUMO and external control algorithms. The interactions implemented in this article include controlling the start of the simulation and acquiring data in the SUMO traffic simulation environment. Therefore, based on the results of the flame graph analysis, we applied PCT to the SimulationStep function in the GA and PSO calibration algorithms, which means to realize the parallelization of simulation operation and the calculation of evaluation indexes. The structures of the two parallel calibration algorithms are shown in **Figures 6 and 7**.

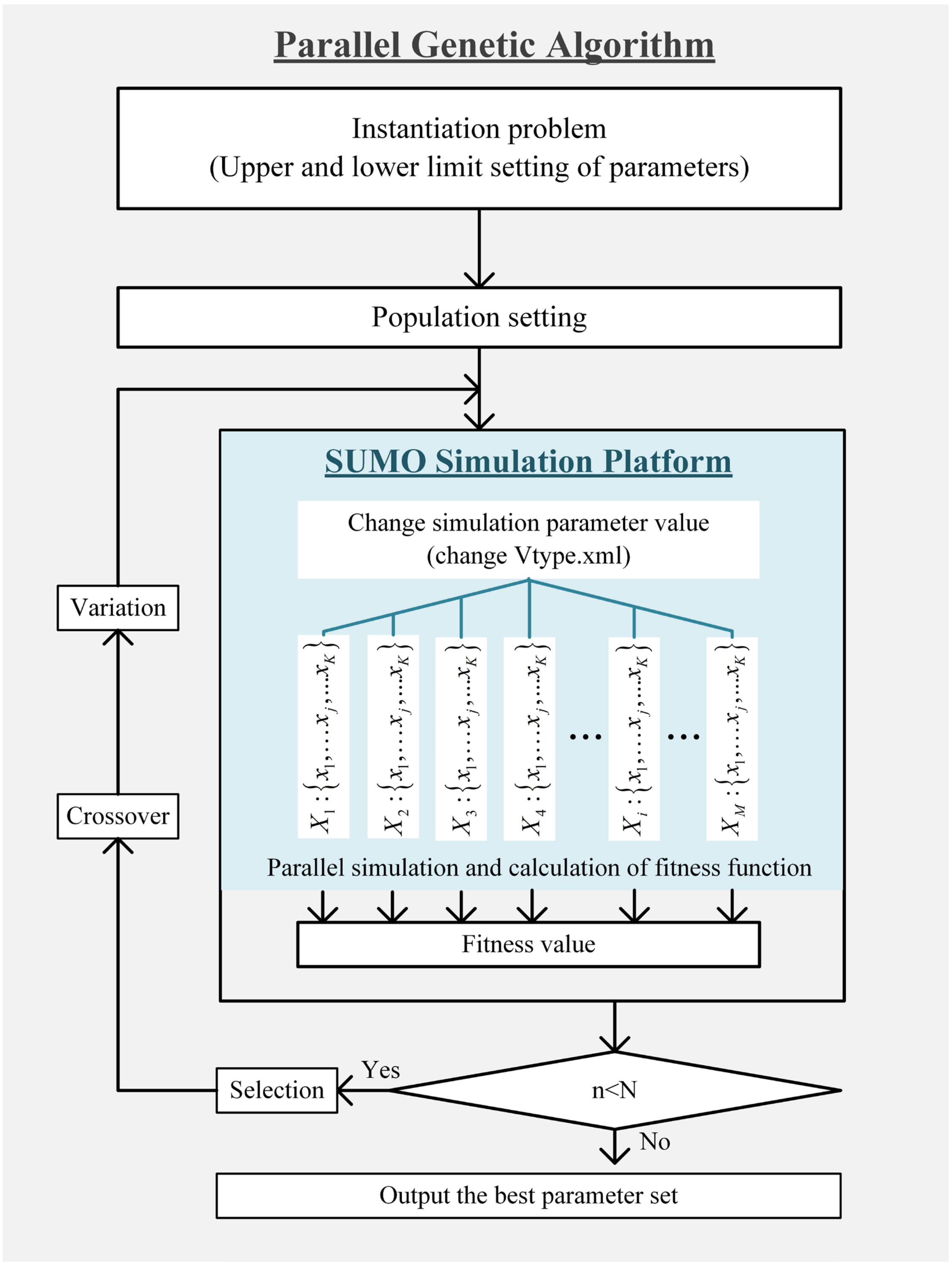

Parallel Genetic Algorithm
Instantiation problem
(Upper and lower limit setting of parameters)
Population setting
SUMO Simulation Platform
Change simulation parameter value
(change Vtype.xml)
$X_1 : \{x_1, \ldots x_j, \ldots x_K\}$
$X_2 : \{x_1, \ldots x_j, \ldots x_K\}$
$X_3 : \{x_1, \ldots x_j, \ldots x_K\}$
$X_4 : \{x_1, \ldots x_j, \ldots x_K\}$
$X_i : \{x_1, \ldots x_j, \ldots x_K\}$
$X_M : \{x_1, \ldots x_j, \ldots x_K\}$
Parallel simulation and calculation of fitness function
Fitness value
Variation
Crossover
Selection
Yes
n<N
No
Output the best parameter set

FIGURE 6 Framework for Parallel GA Calibration *The calibration algorithm and the SUMO simulation platform are marked with a gray background and a blue background, respectively. The values of the simulation parameters that need to be calibrated are changed in the vType.xml script through the traffic control interface (TraCI). In each generation,* $X_1, X_2, \ldots, X_M$ *refer to the position of all individuals in the problem optimization space with K.*

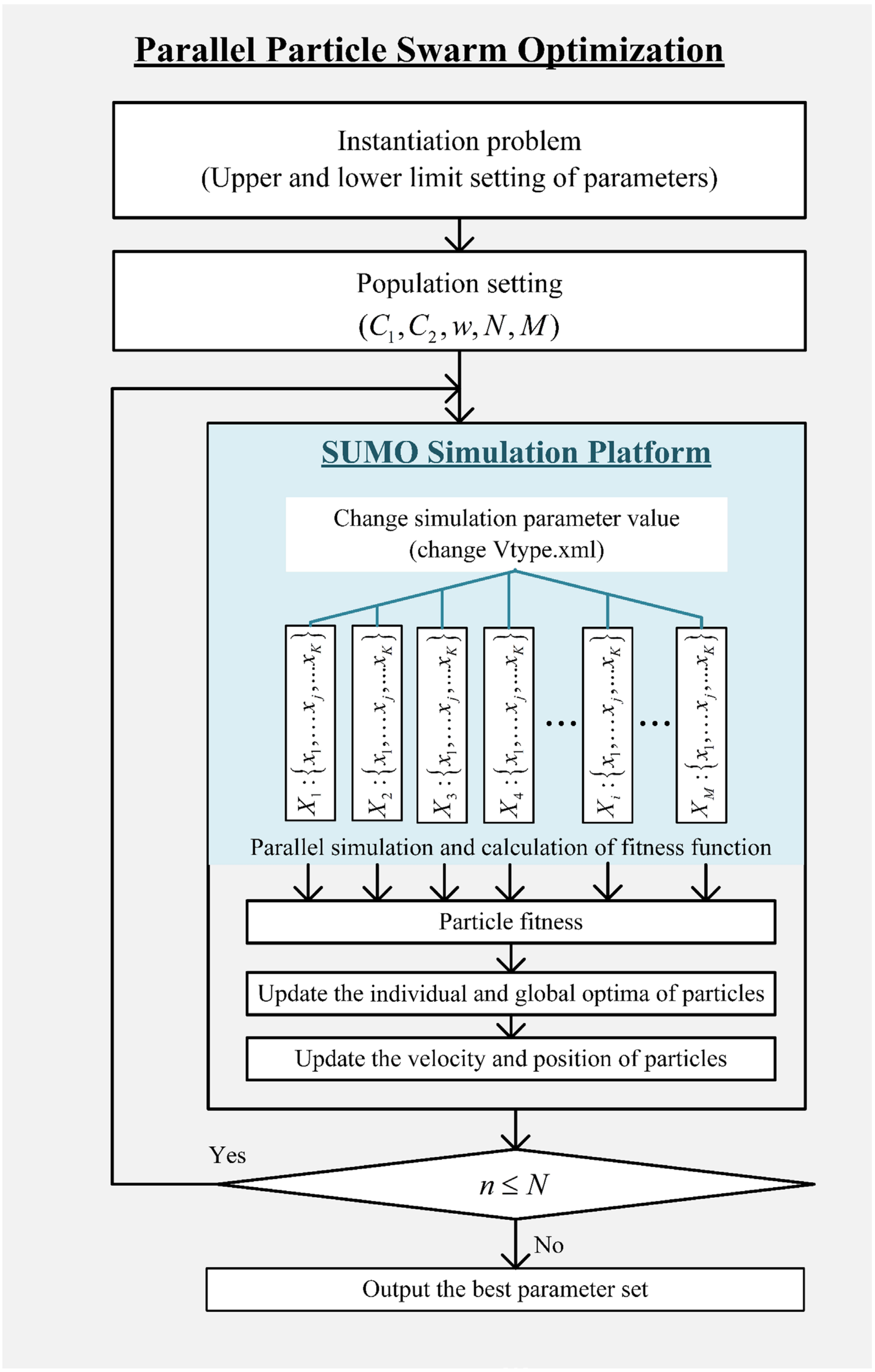

Parallel Particle Swarm Optimization
Instantiation problem
(Upper and lower limit setting of parameters)
Population setting
(C_1, C_2, w, N, M)
SUMO Simulation Platform
Change simulation parameter value
(change Vtype.xml)
$X_1 : \{x_1, \ldots x_j, \ldots x_K\}$
$X_2 : \{x_1, \ldots x_j, \ldots x_K\}$
$X_3 : \{x_1, \ldots x_j, \ldots x_K\}$
$X_4 : \{x_1, \ldots x_j, \ldots x_K\}$
$X_i : \{x_1, \ldots x_j, \ldots x_K\}$
$X_M : \{x_1, \ldots x_j, \ldots x_K\}$
Parallel simulation and calculation of fitness function
Particle fitness
Update the individual and global optima of particles
Update the velocity and position of particles
Yes
$n \le N$
No
Output the best parameter set

FIGURE 7 Framework of Parallel PSO Calibration *The calibration algorithm and the SUMO simulation platform are marked with a gray background and a blue background, respectively. The values*

of the simulation parameters that need to be calibrated are changed in the vType.xml script through the traffic control interface (TraCI). In each generation, $X_1, X_2, \dots, X_M$ *refer to the position of all particles in the problem optimization space with K.*

The realization of the subtasks' load balancing

After the first two steps, the frameworks of the GA and PSO parallel calibration algorithm have been built. According to the characteristics of discrete segments, this step considers the design of load balancing for parallel computing. Load balancing refers to distributing approximately equal amounts of computing work on each processor, so that all processors remain busy at all times, minimizing the idle time of all processors. Load balancing is an important performance indicator of parallel programs. There are two main methods to realize load balancing of parallel computing, including equal distribution of tasks and dynamic task distribution.

In the calibration problem of the microscopic traffic simulation model, the parameter combinations are input into the simulation model. Due to the difference in parameter selection, the running time and computational complexity of each simulation are not exactly the same. There are even some cases where the simulation exits due to the fact that the parameters are too biased. If the method of evenly distributing tasks is adopted in this problem, each processor would run the same number of simulations, making the unbalanced load inevitable.

Therefore, in our work, the mode of task pool is tailored to realize the dynamic allocation of tasks, as shown in **Figure 8**. There are the main process and worker processes in the task pool mode. The main process is the entrance to the program execution, which can be understood as the main function of the program. PCT divides a big problem into small tasks, and the work processes are the program entities that perform the subtasks. Usually, there is only one main process and multiple worker processes. The task pool is set with a maximum number of processes and belongs to the main process. When a new work request is submitted to the task pool, if the pool is not full, a new work process will be created to execute the task. But if the number of processes in the pool has reached the upper limit, then the request will wait. Once a certain task in the pool ends, it will enter the task pool and be executed by the work process. This means that the fastest work process will complete more subtasks.

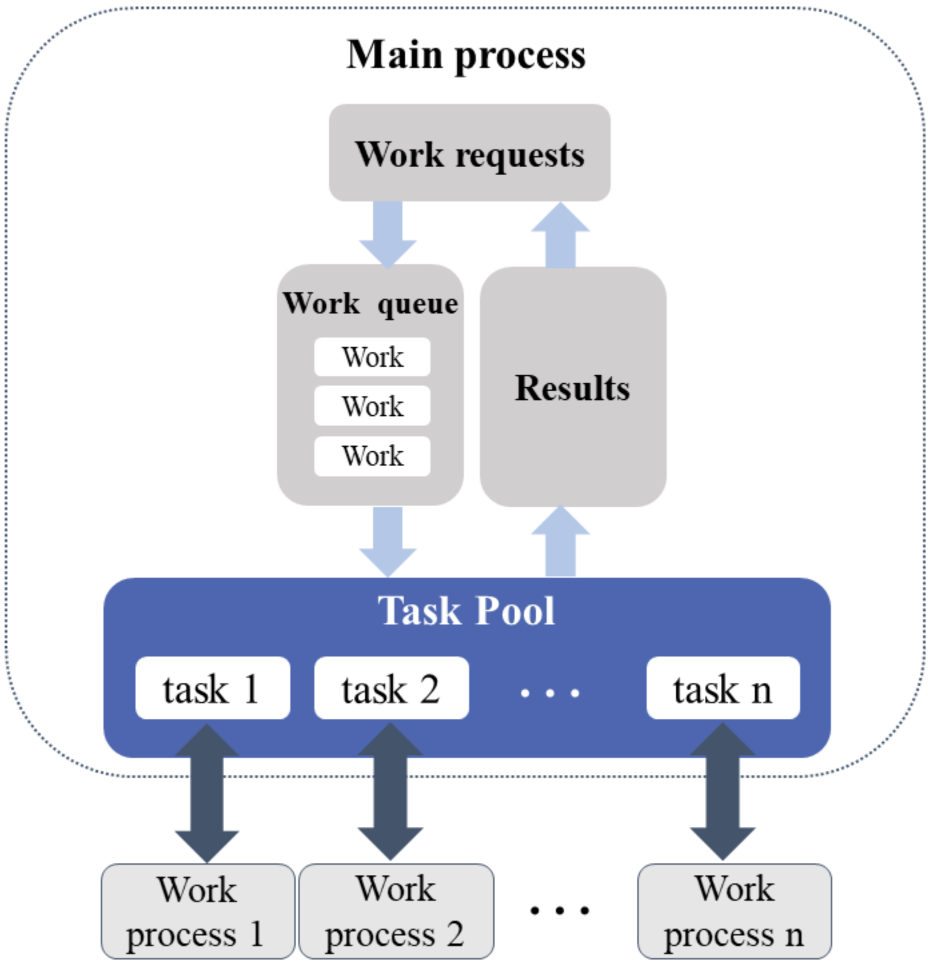


FIGURE 8 The Mode of Task Pool

In the context of the heuristic algorithm calibration problem, the subtask is the simulation operation under a certain set of parameters. The execution results returned to the main process are the accuracy evaluation indexes calculated according to the simulation model. The task pool size is the number of computing cores used in parallel computing.

CASE STUDY

This section presents a case study. A simulation model of a 5 km long highway section is built for parameter calibration, based on the SUMO micro-simulation (*28*). First, the performance of the parallel GA and the parallel PSO proposed is compared from the perspective of accuracy.

More importantly, this paper selects four quantitative indicators to evaluate the efficiency of parallel algorithms from the perspective of calibration computational time and scalability. In the result analysis part, the simulation computational time of the calibration algorithms with or without PCT is compared. One step further is that we change the number of processors used in parallel and compare the application effects of PCT on GA and PSO based on the established efficiency evaluation indicators.

Simulation Scenario

The study area is in the upstream and downstream sections of the Deception Bay Street exit ramp and the Anzac Street entrance ramp of the Bruce Expressway in Australia, with a total length of 5 km, as shown in **Figure 9**. The study area includes basic highway sections, diverge areas and merge areas. The area where the ramp merges into the confluence area is a typical lane-drop bottleneck.

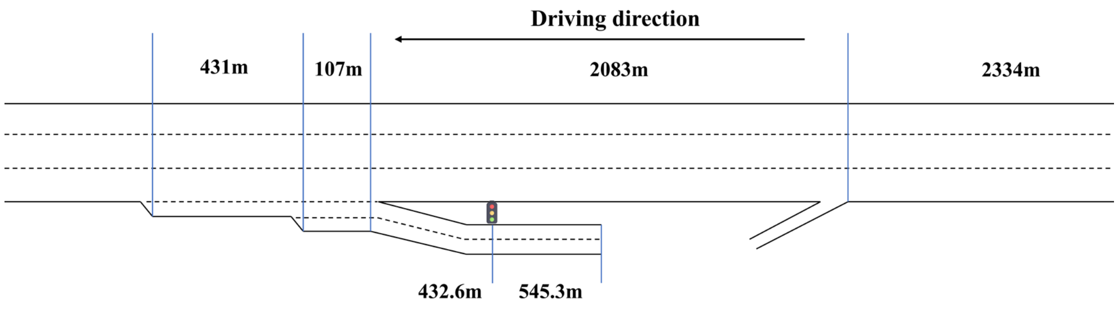


FIGURE 9 The Geometrical Schematic Diagram of The Research Section

Along the study section, there were 10 loop detectors, every 480 meters on average. Cross-sectional data such as traffic, speed, and occupancy rate on the main lane could be obtained. This paper selected the field observation data on September 2, 2020 for parameter calibration research. By looking at the field loop detector data, the traffic entering the road segment began to increase sharply at around 5 AM, and obvious congestion began to appear at the ramp junction at around 6 AM and extended upward. After 8 AM, the congestion began to dissipate. The traffic flow data collected from 6:00 AM to 6:59 AM was used for the calibration experiments. The formation process of bottleneck congestion was included in the calibration period. The observed heatmap for speed during the calibration period is shown in **Figure 10**.

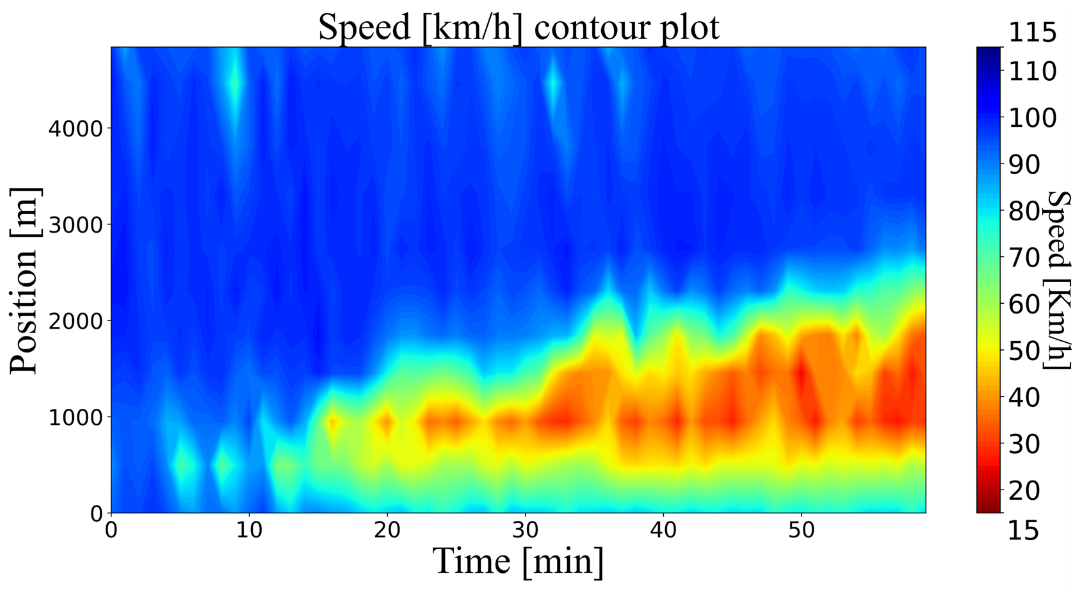


FIGURE 10 Speed Heatmap for mainline (6:00 AM-6:59 AM)

Calibration Accuracy Evaluation Indicators

With reference to Song et al.'s research (*15*), this paper used two quantitative indicators,

$RMSE_{speed}$ and $RMSE_{flow}$, to measure the degree of match between the simulation model and the actual traffic flow. C_1, C_2, and heat diagrams of speed were used to evaluate the degree of bottleneck range matching between the simulation and the observation.

The root mean square error $RMSE_{speed}$ and $RMSE_{flow}$ are shown below.

$$RMSE_{speed} = \sqrt{\frac{\sum_{i=1}^{N_{detector}} \sum_{t=1}^{T} (Sv(i,t)-Rv(i,t))^2}{N_{detector} \times T}} \quad \textbf{(formula 1）}$$

$$RMSE_{flow} = \sqrt{\frac{\sum_{i=1}^{N_{detector}} \sum_{t=1}^{T} (Sf(i,t)-Rf(i,t))^2}{N_{detector} \times T}} \quad (12)$$

The $RMSE_{speed}$ is used as the first objective function (**formula 1**) above. $RMSE_{flow}$ denotes the root mean square error of traffic flow. $Sf(i,t)$ represents the average traffic volume collected by the i-th *(*$i = \{1, 2, \ldots, N_{detector}\}$) detector of the simulation model during the t-th ($t = \{1, 2, \ldots, T\}$) time period, while the volume collected by the actual field detectors is denoted as $Rf(i,t)$. T is the number of time periods during which the detectors collect data (simulation duration is $120\,min$, collected every $2\,min$, $T = 120/2 = 60$). Furthermore, $N_{detector}$ denotes the total number of coil detectors.

C_1 and C_2 are defined as the degree of matching between the simulation and the observation in the bottleneck range, from the perspective of space and time. When C_1=1, it means that the bottleneck area obtained by simulation completely matches with observation. If C_2=1, the bottleneck area and speed value obtained by simulation are completely consistent with observations. The mathematical expressions are as follows:

$$C_1 = \frac{2\sum_{i=1}^{N}\{(\sum_{t=1}^{T}[BS_S(i,t) \wedge BS_R(i,t)]) \cdot (l_{i+} - l_i)\}}{\sum_{i=1}^{N}\{(\sum_{t=1}^{T}[BS_S(i,t) \vee BS_R(i,t)]) \cdot (l_{i+} - l_i)\}} \quad \textbf{(formula 2)}$$

$$C_2 = 1 - \frac{2\sum_{i=1}^{N}\{(\sum_{t=1}^{T}[BS_S(i,t) \wedge BS_T(i,t)] \cdot |Sv(i,t)-Rv(i,t)|) \cdot (l_{i+} - l_i)\}}{\sum_{i=1}^{N}\left\{\left(\sum_{t=1}^{T}[BS_S(i,t) \wedge BS_T(i,t)] \cdot (Sv(i,t)-Rv(i,t))\right) \cdot (l_{i+} - l_i)\right\}} \quad (13)$$

The $C_1$is used as the second objective function (**formula 2**) in the problem formulation section. First, the binary speed space-time maps of the simulated and actual traffic flow need to be calculated and generated as **formula (4-5)**, which are denoted as $BS_S(i,t)$ and $BS_R(i,t)$. $BS_S(i,t) \wedge BS_R(i,t)$ represents the intersection of matrices, which is calculated mathematically by multiplying the internal elements of two matrices, while $BS_S(i,t) \vee BS_R(i,t)$ represents the union. $l_{i+} - l_i$ represents the distance between two coils.

In addition to the four quantitative evaluation indicators, the accuracy of the simulation model can also be evaluated from an intuitive visual point of view by referring to the speed heat diagram.

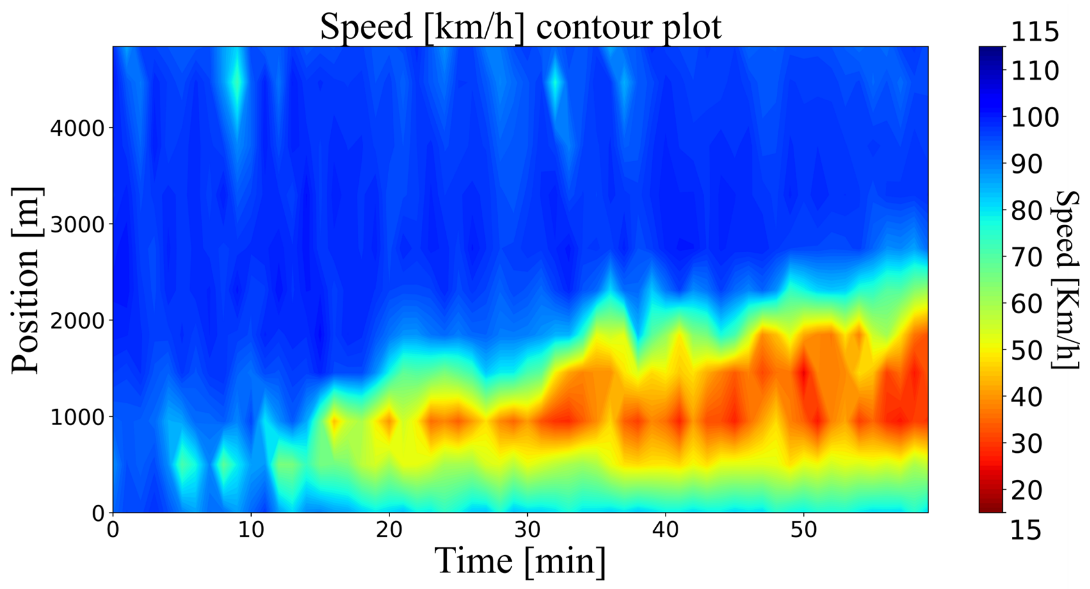
RESULT ANALYSIS

Calibration result

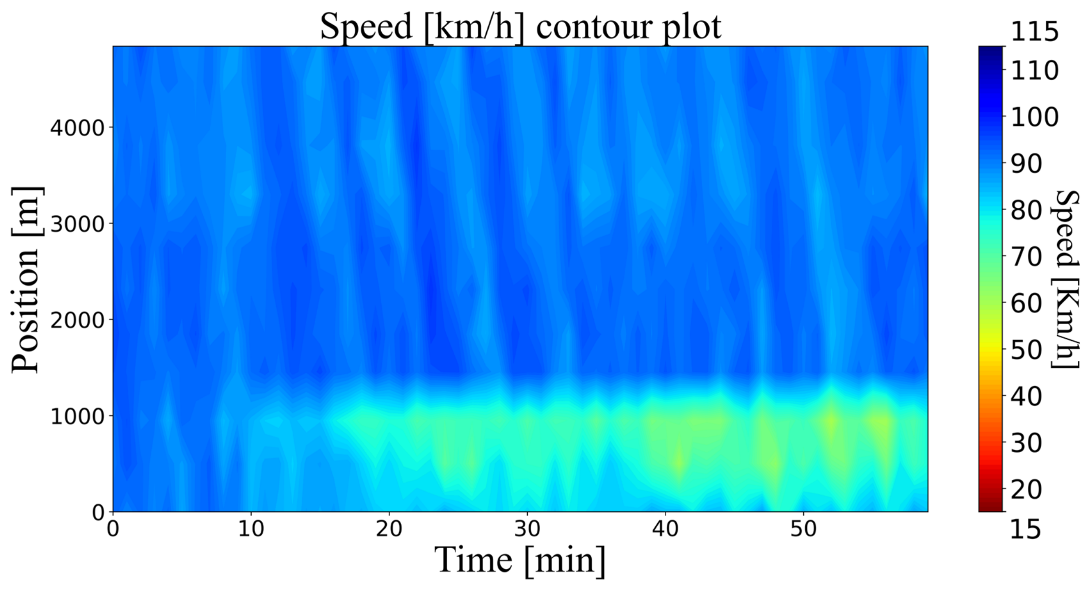
In the case study, the IDM model was selected as the car-following model, and the LC2013 as the lane changing model. Two indicators of $RMSE_{speed}$ and C_1 were selected as the fitness evaluation functions of the GA and PSO calibration algorithms. In the parallel GA calibration algorithm, we set the population size to 32 and the number of iterations to 30. In the parallel PSO calibration algorithm, the number of particle swarms was set to 32, and the number of iterations was also 30. We controlled both evaluation times to be 960 times, which meant that a total of 960 simulation evaluations are carried out during the calibration process.

TABLE 3 Accuracy comparison of calibration results

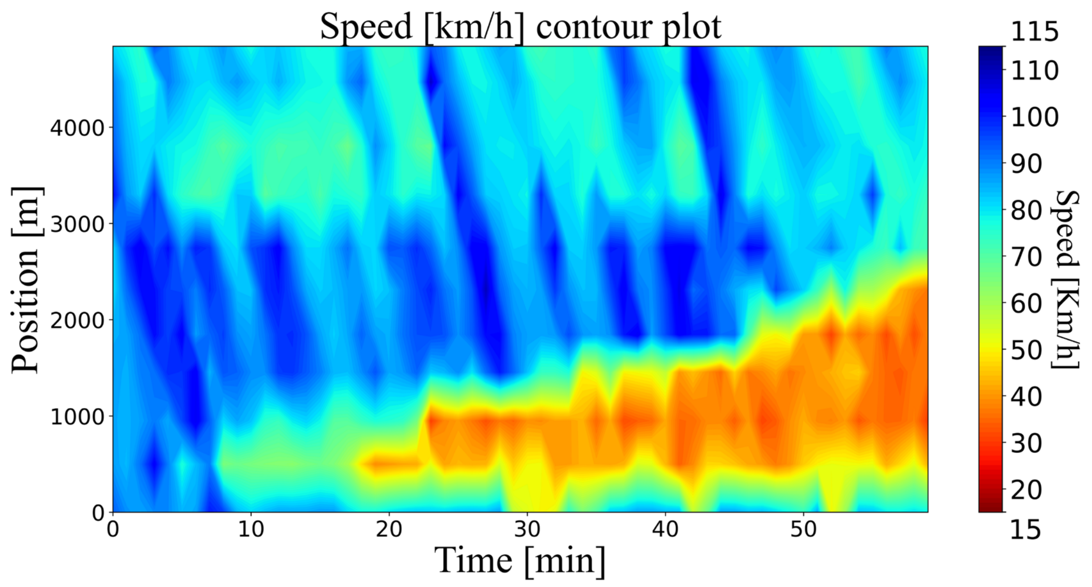
Indicator	$RMSE_{speed}$	$RMSE_{flow}$	C_1	C_2
Default	27.71	30.35	0	0
Parallel GA	18.17	24.71	0.95	0.78
Parallel PSO	19.80	25.04	0.90	0.67



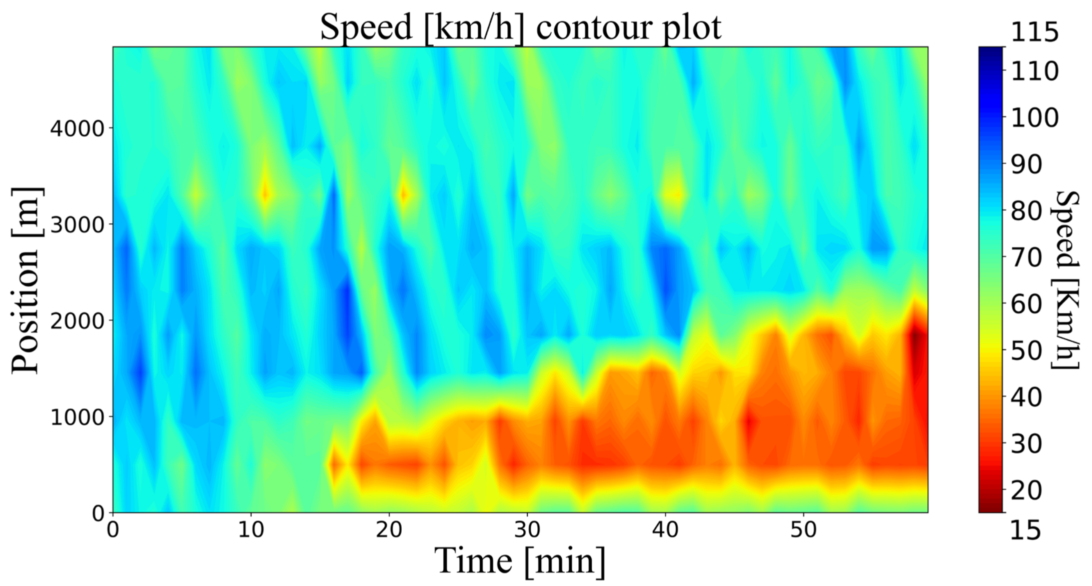
(a)Field



(b)Default



(c)Result of parallel GA



(d)Result of parallel PSO

Figure 11 Comparison of Speed Heatmap

Table 3 shows the accuracy comparison between the parallel GA and the parallel PSO. **Figure 11** shows the speed heatmaps of the two calibration algorithms' results. When using the default values in the SUMO simulation platform, the simulation model cannot simulate the bottleneck state in the actual traffic flow at all. In terms of speed and flow matching, the calibration result of parallel GA is better than that of parallel PSO. The two index values of $RMSE_{speed}$ and $RMSE_{flow}$ have been improved by 34.43% and 18.58% respectively. In terms of the matching degree of the bottleneck range, the parallel GA algorithm seems to perform better. After calibration, the simulation accuracy of the model is greatly improved, and the experimental results verify the effectiveness of the parallel calibration algorithm proposed in this paper.

Efficiency evaluation indicators of PCT

Based on the accuracy comparison indexes, the accuracy of the parallel calibration algorithms was verified. Regarding the evaluation of the efficiency improvement brought by parallel computing, aside from the commonly used computational time, we considered the ability of parallel algorithms to utilize available computing resources, which was called scalability. Thus, the efficiency evaluation indicator system was established based on both the computational time reduction rate and scalability, which included four indicators of calibration computational time, acceleration ratio, parallel efficiency, and algorithm scalability.

(1) Calibration computational time. The calibration time in this project is defined as the time of the entire calibration algorithm from encoding to outputting the best results. Calibration time is the most intuitive and simple evaluation indicator for the improvement of calibration efficiency by using parallel calculation. More importantly, the calibration time is also the basic data necessary to calculate other efficiency evaluation indicators such as acceleration ratio.

(2) Acceleration ratio. The acceleration ratio refers to the ratio of the optimal single-thread calculation time of the program to the parallel calculation time using multiple processors. It is the simplest and most widely used metric to detect the performance of parallel algorithms. The parallel acceleration ratio is defined as follows:

$$S_N = \frac{t_{seq}}{t_N} \tag{14}$$

Where

N: Number of processors used by parallel computing programs

S_N: The acceleration ratio of the parallel algorithm when using N processors.

t_{seq}: The time taken for a serial program to solve the problem on a single processor.

t_N: The time taken to solve the problem with *N* processors in parallel computing.

(3) Parallel efficiency. Parallel efficiency is an evaluation index calculated based on the acceleration ratio, which is defined as follows:

$$E_N = \frac{S_N}{N} \tag{15}$$

The efficiency of parallel computing is affected by the theoretical maximum acceleration ratio S_N, which can also be interpreted as the program's parallelizable ratio. Additionally, hardware components such as network communication quality and speed, application algorithms, communication overhead, and the specific characteristics of the problems being addressed also play significant roles.

(4) Scalability. Scalability refers to the ability of parallel algorithms to increase parallel speed proportionally by adding more computing resources. It is a measure of the extent to which parallel algorithms can effectively utilize the increased capacity of multiprocessors. With the increase of the number of available processors, if the parallel marginal efficiency curve remains basically unchanged or slightly decreases, the scalability of the parallel algorithm is considered to be good. Conversely, if the efficiency curve drops quickly, it is considered that the parallel algorithm has poor scalability.

Results of calibration efficiency experiment

The experiment controlled the evaluation times in parallel GA and parallel PSO calibration algorithms to be the same. When using different numbers of processors in parallel, the calibration computational time, reduction ratio and parallel acceleration ratio results of the two are shown in **Table 4**. **Figure 12** shows the comparison of the time reduction ratio of parallel GA and parallel PSO calibration algorithms under different numbers of processors. **Figure 13** shows the efficiency curves of the two parallel algorithms.

TABLE 4 Time reduction and acceleration ratio

Calibration algorithm	Parallel GA			Parallel PSO		
Number of processors	Calibration time（h）	Time reduction ratio (%)	Acceleration ratio	Calibration time (h)	Time reduction ratio (%)	Acceleration ratio
Serial	5.65	-*	-*	4.84	-*	-*
2	2.31	59.1%	2.45	2.26	53.3%	2.14
3	1.43	74.6%	3.94	1.76	63.6%	2.75
4	1.28	77.3%	4.41	1.32	72.8%	3.68
5	1.16	79.4%	4.85	1.07	77.9%	4.52
6	1.01	82.1%	5.57	0.85	82.4%	5.69
7	0.98	82.6%	5.75	0.80	84.6%	6.09
8	0.97	82.9%	5.85	0.76	85.4%	6.42

*: *We take the calibration time in serial calculation mode without applying PCT as the baseline for comparison. Therefore, in the first row of the table, the time reduction rate and acceleration ratio do not exist, which have no actual physical meaning.*

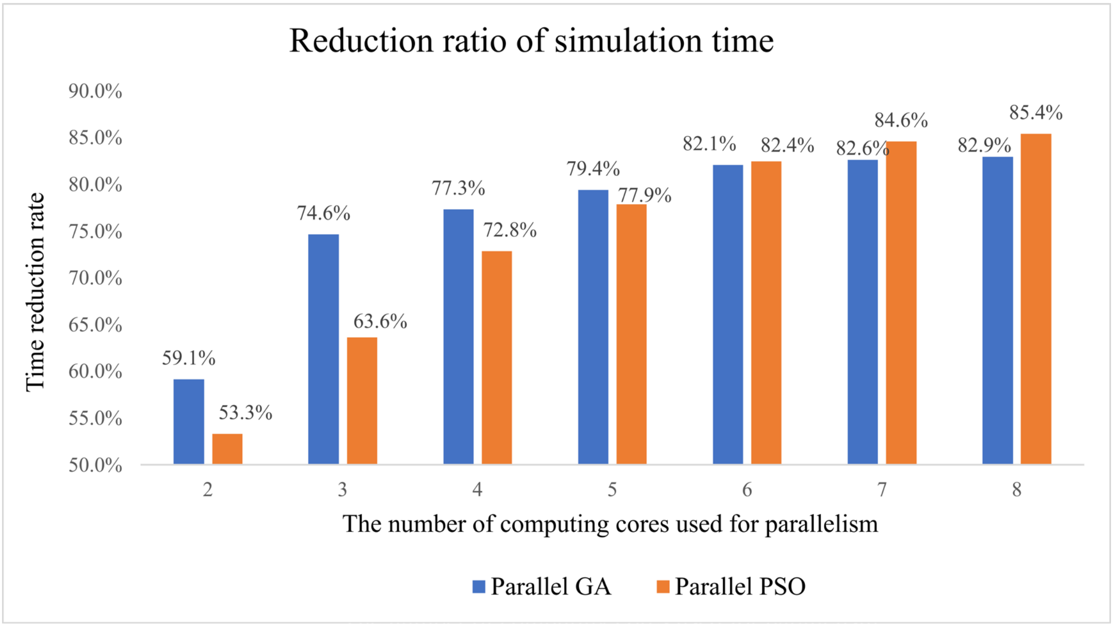


Figure 12 Calibration time reduction ratio

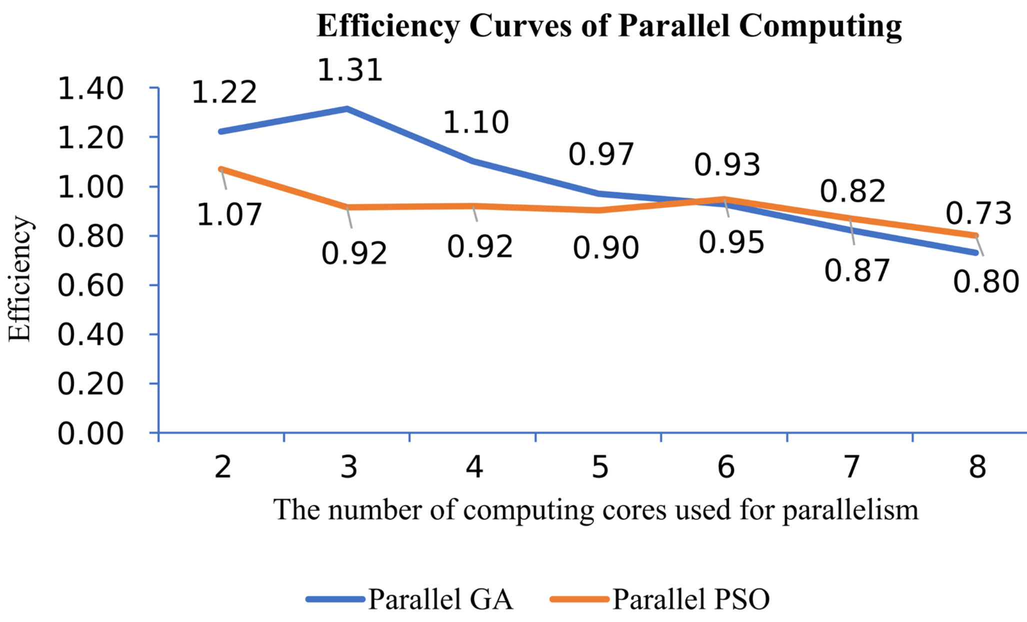


Figure 13 Parallel efficiency curve

It can be seen from the results that the application of parallel computing greatly shortens the calibration time. In the serial calibration algorithm, the calibration takes 5.65 hours. By using parallel computing technology, the calibration time is shortened to less than 1 hour, and a reduction of 80% is achieved. The experimental results verify the great significance of parallel computing in the calibration problem.

As can be seen from the efficiency curves of the two, the curve of the parallel PSO calibration algorithm is gentler. This shows that its scalability is better, which means its ability to use increased computing resources is stronger. When the number of parallel processors is small, parallel computing has a better acceleration effect on the GA calibration algorithm. For example, when using 3 processors, the calibration time of parallel GA is reduced by 74.6% compared to serial calibration, while the time reduction rate of parallel PSO is only 63.6%. However, as the number of processors increases, the acceleration effect of parallel computing in the PSO calibration algorithm becomes better. When 8 processors are used for parallel computing, the calibration time of the parallel PSO calibration algorithm has been reduced by 85.4%, while the reduction rate of the calibration time in the parallel GA calibration algorithm has stagnated at about 80%. In this case, if the effectiveness of parallel algorithms is compared only from the single dimension of the simulation time reduction ratio, once the number of processors used in parallel alters, the comparison result would be invalid. The result of the case study verifies the scientific and necessity of comparing the application effects of parallel computing from the two dimensions of simulation time reduction and scalability.

CONCLUSION AND DISCUSSION

At present, the heuristic algorithm has been widely used in the parameter calibration problem of microscopic traffic simulation, but the calculation bottleneck of the trial-and-error simulation makes it unable to meet the requirements of rapid calibration. PCT can be applied to optimize this computing bottleneck. This paper designs and implements parallel GA and parallel PSO calibration algorithms in accordance with the three steps of parallel framework selection, algorithm bottleneck identification and subtask load balancing design. When evaluating the application effect of parallel computing, the evaluation indicator system was established from the two dimensions of calibration computational time and scalability.

This research verifies that the application of parallel computing to parameter calibration can greatly speed up the calibration speed, and the acceleration effect reached above 80% in the case study. At the same time, the research results also prove that the ability of parallel algorithms for utilizing increased resources, which is called scalability, is an important and necessary indicator for evaluating parallel algorithms. The evaluation indicator system proposed in this paper from calibration computational time and scalability is reasonable and necessary. As one can be seen from the results, the scalability of parallel PSO is better than that of parallel GA.

The findings of this paper have contributed to the rapid calibration of microscopic traffic simulation parameters. It also guides researchers who apply PCT to the calibration problem in the future.

This study also holds limitations. The GA and PSO algorithms used in this paper are the most basic forms. At present, both algorithms have been improved a lot, producing many variants with better optimization effects. In future work, we will consider the latest variants of PSO and GA, and consider complex parameter adjustment mechanisms such as time-varying acceleration coefficients in PSO.

On the other hand, although the PCT used in this paper greatly improved the efficiency of the parameter calibration, the current calibration computational time within one hour is still far from the requirement of online calibration. In the future, the calibration method based on the proxy model can be combined with PCT to further speed up the calibration process.

Furthermore, verifying the applicability of this scheme to road networks of different scales and comparing the application effects is another worthy research direction. However, due to the lack of field traffic flow data on large-scale road networks, currently we are unable to do further research for the time being. As the network size increases, completely different parallel computing schemes can be applied. For example, we can divide the large road network into discrete road segments, and simulation model of the traffic flow on each segment can be run by different computing cores The effect of such a parallel scheme will definitely be different from our current scheme. We will take these into account in order to make a comprehensive, systematic and scientific research on the application of PCT in the calibration of microscopic traffic simulation models. This paper will be an essential foundation for our future research.

ACKNOWLEDGEMENT

This work is sponsored by National Nature and Science Foundation of China [52002279].

AUTHOR CONTRIBUTION STATEMENT

The authors confirm contribution to the paper as follows: study conception and design: Lanyue Tang, Yu Han, Jian Sun; analysis and interpretation of results: Lanyue Tang, Yu Han, Duo Zhang, Ye Tian, Jian Sun; draft manuscript preparation: Lanyue Tang, Yu Han, Duo Zhang, Ye Tian, Aohui Fu, Lishengsa Yue, He Zhang. All authors reviewed the results and approved the final version of the manuscript.

REFERENCES

[1] Gardes, Y., A. D. May, J. Dahlgren, and A. Skabardonis. Freeway calibration and application of the PARAMICS model. 2002.

[2] Gomes, G., A. May, and R. Horowitz. Congested Freeway Microsimulation Model Using VISSIM. Transportation Research Record, Vol. 1876, No. 1, 2004, pp. 71-81.

[3] Osorio, C., and V. Punzo. Efficient calibration of microscopic car-following models for large-scale stochastic network simulators. Transportation Research Part B: Methodological, Vol. 119, 2019, pp. 156-173 %U https://linkinghub.elsevier.com/retrieve/pii/S019126151731055X.

[4] Ištoka Otković, I., T. Tollazzi, and M. Šraml. Calibration of microsimulation traffic model using neural network approach. Expert Systems with Applications, Vol. 40, No. 15, 2013, pp. 5965-5974 %U https://linkinghub.elsevier.com/retrieve/pii/S0957417413002893.

[5] Dorothy, P. W., R. P. Ambadipudi, and R. M. Kill. Development and Validation of Large-Scale Microscopic Models. 2006.

[6] Li, X., and G. O. Yeh. Calibration of cellular automata by using neural networks for the simulation of complex urban systems. Environment & Planning A, Vol. 33, No. 8, 2001, pp. 1445-1462.

[7] Siddharth, S. M. P., and G. Ramadurai. Calibration of VISSIM for Indian Heterogeneous Traffic Conditions. Procedia - Social and Behavioral Sciences, Vol. 104, 2013, pp. 380-389 %U https://linkinghub.elsevier.com/retrieve/pii/S1877042813045229.

[8] Abdalhaq, B. K., and M. I. A. Baker. Using Meta Heuristic Algorithms to Improve Traffic Simulation. Journal of Algorithms and Optimization, Vol. 2, 2014, p. 19.

[9] Hou, Z., and J. Lee. Multi-Thread Optimization for the Calibration of Microscopic Traffic Simulation Model. Transportation Research Record: Journal of the Transportation Research Board, Vol. 2672, No. 20, 2018, pp. 98-109 %U http://journals.sagepub.com/doi/110.1177/0361198118796395.

[10] Verkaik, J., J. V. Engelen, S. Huizer, M. Bierkens, and G. Essink. Distributed memory parallel computing of three-dimensional variable-density groundwater flow and salt transport. Advances in Water Resources, 2021, p. 103976.

[11] Song, W., Y. Zheng, C. Fu, and P. Shan. A novel batch image encryption algorithm using parallel computing. Information Sciences, Vol. 518, 2020, pp. 211-224.

[12] Fu, Z., X. Sun, Q. Liu, L. Zhou, and J. Shu. Achieving Efficient Cloud Search Services: Multi-Keyword Ranked Search over Encrypted Cloud Data Supporting Parallel Computing. Ieice Transactions on Communications, Vol. 98, No. 1, 2015, pp. 190-200.

[13] Shao, W., L. Yao, Z. Ge, and Z. Song. Parallel Computing and SGD-Based DPMM For Soft Sensor Development With Large-Scale Semisupervised Data. IEEE Transactions on Industrial Electronics, Vol. 66, No. 8, 2019, pp. 6362-6373.

[14] Dadashzadeh, N., M. Ergun, A. S. Kesten, and M. Zura. Improving the calibration time of traffic simulation models using parallel computing technique.In 2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS), IEEE, Cracow, Poland, 2019. pp. 1-7 %@ 978-971-5386-9484-5388 %U https://ieeexplore.ieee.org/document/8883322/.

[15] Song, R., and J. Sun. Calibration of a micro-traffic simulation model with respect to the spatial-temporal evolution of expressway on-ramp bottlenecks. SIMULATION, Vol. 92, No. 6, 2016, pp. 535-546.

[16] Treiber, M., A. Hennecke, and D. Helbing. Congested Traffic States in Empirical Observations and Microscopic Simulations. Physical Review E, Vol. 62, 2000, pp. 1805-1824.

[17] Erdmann, J. SUMO's Lane-changing model.In 2nd SUMO User Conference, 2015.

[18] Tursun, M., and M. Geni. SUMO Simulation of Oversaturated Weaving Section and Lane-changing Model Optimization. Journal of Xinjiang University(Natural Science Edition), 2018.

[19] Saltelli, A., P. Annoni, I. Azzini, F. Campolongo, M. Ratto, and S. Tarantola. Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. COMPUTER PHYSICS COMMUNICATIONS, 2010.

[20] Joe, S., and F. Y. Kuo. Constructing Sobol sequences with better two-dimensional projections. SIAM Journal on Scientific Computing, Vol. 30, No. 5, 2008, pp. 2635-2654.

[21] McKay, D., R. Beckman, and W. Conovcr. „A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code "Technometrics 21. 1979.

[22] Punzo, V., M. Montanino, and B. Ciuffo. Do We Really Need to Calibrate All the Parameters? Variance-Based Sensitivity Analysis to Simplify Microscopic Traffic Flow Models. IEEE Transactions on Intelligent Transportation Systems, Vol. 16, No. 1, 2015, pp. 184-193 %U https://ieeexplore.ieee.org/document/6849526.

[23] Holland, J. Adaptation in natural and artificial systems : an introductory analysis with application to biology. Control & Artificial Intelligence, 1975.

[24] Kennedy, J., and R. Eberhart. Particle swarm optimization[C]. 1995.

[25] Liu, Y., B. Zou, A. Ni, L. Gao, and C. Zhang. Calibrating microscopic traffic simulators using machine learning and particle swarm optimization. Transportation Letters, Vol. 13, No. 4, 2021, pp. 295-307.

[26] Rezaee Jordehi, A., and J. Jasni. Parameter selection in particle swarm optimisation: a survey. Journal of Experimental & Theoretical Artificial Intelligence, Vol. 25, No. 4, 2013, pp. 527-542.

[27] Shi, Y. A Modified Particle Swarm Optimizer.In Proc of IEEE Icec Conference, 1998.

[28] Krajzewicz, D. Traffic Simulation with SUMO – Simulation of Urban Mobility. Fundamentals of Traffic Simulation, 2011.