Transportation Research Part E 210 (2026) 104761

Contents lists available at ScienceDirect

Transportation Research Part E

journal homepage: www.elsevier.com/locate/tre

Linear regression parallel block coordinate descent method with Barzilai–Borwein steps for large-scale traffic assignment problems

Kai Zhang [a], Zhiyuan Liu [b], Yuan Zhang [c], Honggang Zhang [d], Xiaowen Fu [a,*]

[a] Department of Industrial and System Engineering, Faculty of Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong 999077, China
[b] Jiangsu Key Laboratory of Urban ITS, Jiangsu Province Collaborative Innovation Center of Modern Urban Traffic Technologies, School of Transportation, Southeast University, Nanjing 211189, China
[c] Network and Information Center, Southeast University, Nanjing 210096, China
[d] Department of Logistics and Maritime Studies, Faculty of Business, The Hong Kong Polytechnic University, Kowloon, Hong Kong 999077, China

ARTICLE INFO

Keywords:
Traffic assignment
Linear regression
Parallel block coordinate descent
Gradient projection
Barzilai–Borwein step size

ABSTRACT

Traffic assignment is the cornerstone of the conventional four-step transportation planning framework. As a fundamental technique for predicting network flow distribution, it is pivotal in optimizing transportation planning and infrastructure design. However, traditional traffic assignment algorithms have a high computational requirement when addressing increasingly large-scale problems driven by ever-growing travel demand and expanding network sizes in real-world applications, making the trade-off between computational efficiency and solution accuracy increasingly critical. This study proposes a novel linear regression parallel block descent (LR-PBCD) method to address this challenge. First, we comprehensively analyze origin–destination (OD) pair characteristics and path travel time distributions. We then apply a linear regression model that identifies hard-to-converge OD pairs, followed by a hierarchical decomposition strategy using parallel block coordinate descent. A gradient projection algorithm is implemented within each block that uses fixed-step updates for normal OD pairs and the Barzilai–Borwein steps algorithm for hard-to-converge OD pairs. Experimental validation on real-world networks demonstrates that the LR-PBCD method improves solution efficiency over conventional methods while maintaining solution precision, providing a computationally efficient paradigm for large-scale transportation network analysis.

1. Introduction

Traffic assignment algorithms have been used for almost seven decades and have been developed into several forms, such as link-based, path-based, and origin-based. However, faced with the challenges of continuously expanding traffic network scales and increasing travel demand in the real world, the computation required by traditional algorithms to solve large-scale traffic assignment problems (TAPs) is rising, making it increasingly challenging to balance computational efficiency and accuracy (Perederieieva et al. 2015; Xie et al. 2018; Raadsen et al. 2020).

In the era of big data and smart mobility, statistical tools such as linear regression are widely used to analyze transportation

* Corresponding author.
E-mail addresses: kai-frank.zhang@polyu.edu.hk (K. Zhang), zhiyuanl@seu.edu.cn (Z. Liu), zhangyuan@seu.edu.cn (Y. Zhang), honggang.zhang@polyu.edu.hk (H. Zhang), xiaowen.fu@polyu.edu.hk (X. Fu).

https://doi.org/10.1016/j.tre.2026.104761
Received 11 July 2025; Received in revised form 6 January 2026; Accepted 10 February 2026
Available online 17 March 2026

Table 1
The development of traffic assignment algorithms.

Algorithm type	Convergence speed	Memory space	Representative serial algorithms	Representative parallel algorithms	Representative data-driven machine learning algorithms
Link-based	Slow	Low	FW (LeBlanc et al. 1975), PARTAN (Florian et al. 1987), Conjugate FW (Mitradjieva and Lindberg 2013)	Parallel FW (Allen 2023), ADMM (Liu et al., 2023a)	Implicit neural network (Liu and Yin, 2025), Graph convolution (Liu et al. 2025), Graph attention (Hu and Xie, 2025)
Path-based	Fast	High	DSD (Larsson and Patriksson, 1992), GP(Jayakrishnan et al. 1994; Chen and Jayakrishnan 1998; Xie et al. 2018), PG (Florian et al. 2009) RG (Babazadeh et al. 2020)	PBCD (Chen et al., 2020b), iPBCD (Wang et al., 2021b)	Computational graph (Wu et al. 2018), Graph attention (Hu and Xie, 2025), Implicit neural network (Liu et al. 2023b; Liu and Yin, 2025)
Bush-based	Fast	Moderate	OBA (Bar-Gera, 2002), B (Dial 2006), QBA (Nie 2016), LUCE (Gentile 2014), TAPAS (Bar-Gera, 2010), iTAPAS (Xie and Xie, 2016)	Parallel B (Potuzak and Kolovsky, 2022)	–

Notes: In the serial algorithms, FW denotes the Frank–Wolfe algorithm; GP denotes Gradient Projection; PG denotes Projection Gradient; DSD denotes the Disaggregate Simplicial Decomposition method; OBA denotes the Origin-Based Algorithm; B denotes Bush; QBA is a "quasi-bush" algorithm that avoids line search; LUCE denotes the Local User Cost Equilibrium; TAPAS denotes Traffic Assignment by Paired Alternative Segments. In the parallel algorithms, ADMM denotes the Alternating Direction Method of Multipliers; PBCD denotes the Parallel Block Coordinate Descent method.

patterns through the examination of features such as demand and path overlap. While most optimization methods focus on precise solutions, we combine linear regression with block coordinate descent (BCD) to tackle large-scale traffic assignment problems. Our framework uses the linear regression model to identify key convergence patterns from traffic data and optimizes resource allocation through parallel computing. This proposed approach efficiently balances accuracy and speed, providing a practical solution for analysing large modern transportation networks.

1.1. Literature review

In this section, we review the development of traffic assignment research methods, which have evolved into three major methodological frameworks: link-based, path-based, and bush-based approaches. We summarize the representative papers of these three categories in Table 1, including serial, parallel and data-driven machine learning algorithms.

1.1.1. User equilibrium TAP

The traditional "four-step" transportation planning process includes an independent analysis of trip generation, trip distribution, mode split, and traffic assignment. A distinct methodology is employed at each step, with the output of the previous step serving as the input for the next step (Sheffi 1984; Wu et al. 2018). User equilibrium (UE), also known as Wardrop equilibrium, represents an assignment pattern in which drivers choose their routes based on self-interest, and no individual driver can decrease their travel cost by unilaterally changing their route (Wardrop 1952). UE serves as the foundational principle governing the final step of traffic assignment in the "four-step" transportation planning process.

TAPs play an essential role in transport planning and management and have a long history dating back to the early 1950 s (Sheffi 1984; Florian and Hearn 1995). Perederieieva et al. (2015) outlined an overview of practical algorithms for TAPs, and Buchhold et al. (2019) placed the algorithms for solving TAPs into three categories depending on how their solutions are presented (i.e., link-based, origin-based, and path-based). Origin-based algorithms have high computational efficiency, but their algorithmic complexity is high. A representative algorithm is the improved traffic assignment by paired alternative segments (iTAPAS) (Xie and Xie, 2016). However, path-based algorithms are well recognized as being the most effective category and are easy to apply using parallel computing (Sheffi 1984; Jayakrishnan et al. 1994; Florian and Hearn 1995; Xie et al. 2018; Babazadeh et al. 2020).

Step size in traffic assignment algorithms is determined as either fixed or dynamic. Fixed step sizes were initially employed by Xie et al. (2018) within improved gradient projection (iGP) algorithms for solving UE problems, and were later extended by Chen et al. (2020a) to BCD frameworks. For dynamic step sizes, Liu et al. (2009) pioneered the self-regulated averaging (SRA) method for stochastic user equilibrium (SUE), while Zhang et al. (2023a) integrated the Armijo dynamic step size into successive over-relaxation (SOR) algorithms for the UE, albeit with increased computational overheads due to its parameter-intensive calculations. Du et al. (2021) made a significant advancement by demonstrating the effectiveness and computational efficiency of the parameter-free Barzilai–Borwein (BB) step size for SUE problems. Further refining this approach, Tan et al. (2024) combined the Armijo and BB step sizes for path-based SUE solutions, and identified the practicality of the BB method for large-scale networks due to its elimination of manual parameter tuning.

1.1.2. Data-driven processing techniques for TAPs

Data-driven processing techniques have been widely adopted to enhance the accuracy and efficiency of TAP models. Data-driven processing strategies employ various techniques to improve model performance, particularly during preprocessing steps such as data cleaning, feature selection, and model training. Integrating data-driven processing techniques to amplify the efficiency of traffic allocation algorithms marks a significant advancement in traffic management research. Pioneering this effort, Lu et al. (2015) enhanced the TAP by applying a normalization method to regularize the weight matrix, paving the way for more precise traffic predictions and allocations. Building on this foundation, Chen et al. (2021) further refined the approach by regularizing trajectory dimensionless features, such as free flow travel time (FFTT), capacity, and distance, thereby improving the algorithm's ability to handle dynamic traffic conditions. In a similar vein, Wang et al. (2021a) introduced a novel strategy that incorporates a random dropout technique into the flow update module of a TAP, significantly increasing the robustness and generalizability of traffic allocation predictions. Graf et al. (2022) used linear regression to analyze historical queue length data for the linear prediction of future queue lengths, constructing a linear predictor to assist in the optimization of dynamic traffic assignment models. Normalizing data generally improves a model's convergence speed (Ioffe and Szegedy 2015; Salimans and Kingma 2016; Huang et al. 2019). Moreover, when the population parameters are known, normalizing errors using the standard score proves effective for normally distributed populations (Freedman et al. 2020). Furthermore, Liu et al. (2023b) and Liu and Yin (2025) developed end-to-end traffic assignment models based on implicit neural networks. More recently, graph convolutional and graph attention techniques have been applied to TAP (Hu and Xie, 2025; Liu et al. 2025).

1.1.3. Parallel methods for TAPs

Due to the continuous expansion of traffic network sizes and the persistent increase in travel demand, traditional serial algorithms face challenges in computational efficiency. For instance, it has been demonstrated that computing the shortest path in TAPs is highly time consuming, particularly for large-scale networks, where calculating the shortest routes can take up more than 90% of the computation time (Gendreau et al. 1999; Zhang et al. 2023b). However, with continued improvements in computer hardware performance and the rapid development of parallel computing technologies, strong hardware support is now available for quickly solving large-scale network equilibrium problems (Feijoo and Meyer 1988; Jafari et al. 2017; Chen et al. 2020a) using parallel algorithms. The design frameworks of parallel algorithms fall into three categories: parallel algorithms based on model decomposition, parallel algorithms based on flow parallel loading, and parallel algorithms based on network partitioning (Liu et al., 2023a).

1) Parallel algorithms based on network partitioning.

Jafari et al. (2017) introduced a network decomposition method known as the decomposition approach to the static TAP that splits the traffic network into two subnetworks and creates many interface links. They then employed shared memory to parallelize the TAP across these two subnetworks. Yahia et al. (2018) subsequently introduced the flow-weighted spectral partitioning algorithm using the normalized graph Laplacian to divide the network, demonstrating its superiority in minimizing the interflow between subnetworks. Chen et al. (2020b) implemented parallel data exchange between partitioned regional subnetworks to parallelize existing simulation programs. Zhang et al. (2022) proposed a dual spatiotemporal partitioning method for effective road network segmentation. Recently, Liu et al. (2024) established a microscopic network partitioning framework based on spectral graph theory to accelerate large-scale traffic simulations.

2) Parallel algorithms based on model decomposition

Feijoo and Meyer (1988) proposed a piecewise linear approximation method for the commodity flow equilibrium problem that made the objective function approximately separable, thereby facilitating the application of parallel computing techniques. Chen and Jayakrishnan (1998) noted that the path flows updated between OD pairs can be performed with fixed link costs, thereby providing a theoretical basis for the use of parallel computing for OD subproblems. Larsson and Patriksson (1992) introduced the disaggregate simplicial decomposition (DSD) method to realize the parallel computation of subproblems in UE traffic assignments.

Recently, Liu et al. (2023a) introduced a novel algorithmic framework based on the model decomposition method, effectively leveraging the fundamental principles of the alternating direction method of multipliers (ADMM) to efficiently decompose a traffic assignment model, thereby establishing a computing framework with a high level of parallelism. Based on the fundamental framework of the ADMM algorithm, Zhang et al. (2023c) later developed a new parallel algorithmic process and applied it to solve network equilibrium problems that considered elastic demand. Liu and Meng (2013) proposed a distributed computing method for SUE problems based on a probabilistic approach.

3) Parallel algorithms based on parallel flow loading.

Parallelizing existing algorithms is more straightforward than developing complex model decomposition methods, and such algorithms are primarily designed based on the parallel loading of traffic flows. Recently, Chen et al. (2020a) introduced the parallel block coordinate descent (PBCD) framework to solve large-scale traffic assignment problems in parallel. They partitioned the original problem based on OD sets and computed multiple OD subproblems in parallel within each block. Subsequently, Wang et al. (2021a) pointed out that the original PBCD algorithm framework did not consider the rules for block partitioning and the sequence for updating block indexes, and proposed several strategies to improve the PBCD algorithm, which they called improved PBCD (iPBCD). Liu et al. (2021) further applied PBCD to solve suboptimal network loading capacity issues and enhance the efficiency of network design problems by parallelizing the solution of lower-level network equilibrium problems. Zhang et al. (2023b) used the PBCD method to solve the combined mode split and traffic assignment problem. Recently, Liu et al. (2025) proposed a dynamic block scaling technique, and experiments have shown that it enhances the efficiency of the PBCD algorithm in utilizing parallel computing resources to solve the TAP problem. Zhang et al. (2025) adopted the idea of vertex coloring (VC) in graph theory to divide the problem into blocks, which addressed the path overlap issue among OD pairs.

Table 2
Notation used in this study.

Symbol	Content
$G = (N, A)$	the transportation network, where N denotes the set of nodes and A denotes the set of links
W	the set of OD pairs,$od \in W$
s	block in an OD pair set W,$S = \{1, 2, ..., s\}$
q_{od}	the travel demand between OD pair od
P^{od}	the set of paths between OD pair od,$p \in P$
$\delta_{p,a}^{od}$	the link-path incidence, where $\delta_{p,a}^{od}$ equals 1 when path p uses link a, and 0 otherwise
x_a	the traffic volume on link a
f_p^{od}	the flow on path $p \in P^{od}$ between OD pair od, where $\overline{p}$ is the shortest path
$t_a(x_a)$	the travel time of link $a \in A$
L_i	the Lipschitz constant for i_{th} block

BCD methods are widely adopted in large-scale optimization due to their low per-iteration cost, modest memory requirements, parallelization potential, and ability to exploit problem structure (Wright 2015). The efficiency of BCD algorithms hinges on three aspects: variable block partitioning, block selection strategy, and the update rule for each block. Richtárik and Takáč (2014) established linear convergence for randomized BCD in a composite function minimization. Necoara et al. (2017) extended this approach to parallel randomized BCD for large-scale, linearly constrained convex network problems, and achieved linear convergence. Lopes et al. (2019) introduced accelerated BCD leveraging active set identification for bound-constrained composite minimization, achieving globally convergent and efficient performance on large-scale regularized logistic regression. Mutny et al. (2020) analyzed randomized Newton-like BCD with determinantal sampling and showed that the convergence rate depends on the Hessian approximation's eigenvalue distribution. They also offered interpretable guarantees and guidance on the optimal block size to minimize the computational cost. Nutini et al. (2022) further advanced BCD by proposing faster greedy block-selection strategies that surpass the Gauss–Southwell rule, efficient message-passing for large sparse blocks, and superlinear convergence for sparse problems. Recently, Tran-Dinh and Luo (2025) proposed two randomized block-coordinate optimistic gradient algorithms for large-scale root-finding: a non-accelerated method with constant step sizes that achieves $O(1/k)$ best-iterate convergence under Lipschitz continuity and weak Minty assumptions, and an accelerated variant that reaches $O(1/k^2)$ last-iterate rates under co-coercivity, both with almost sure convergence guarantees.

1.2. Objectives and contributions

This study presents an integrated computational framework that synergizes data-driven analytics with parallel optimization to address the scalability challenges in TAPs. Our research objectives and contributions can be summarized as follows:

(1) Linear regression method identification of hard-to-converge OD pairs via feature extraction and normalization.
(2) An LR-PBCD algorithm is developed to compute the multiple OD-based traffic assignment problem in parallel, with a sublinear convergence guarantee.
(3) Comprehensive evaluation on five transportation networks and sensitivity analyses confirming the proposed algorithm can improve solution efficiency over conventional methods while maintaining solution precision.

The remainder of this paper is organised as follows. Section 2 introduces the traffic assignment problem and method. Section 3 describes the propsed LR-PBCD method for solving TAPs. Section 4 reports the numerical experiments, and Section 5 summarises the main results and discusses directions for future research.

2. Traffic assignment problem and method

This section introduces the notation first, and then presents the TAP model used in this study. The BCD method and equidistant sampling are then presented. For ease of presentation, the notation is introduced in Table 2.

2.1. Traffic assignment problem

UE, also known as Wardrop equilibrium, is an assignment pattern where drivers select the most efficient route for themselves, and no driver can reduce their travel cost by unilaterally changing their route independently (Wardrop 1952). As the foundational principle governing the traffic assignment step in the traditional four-step transportation planning process, UE provides a solid mathematical theoretical framework for large-scale traffic assignment problems.

$$\min \sum_{a \in A} \int_0^{x_a} t(x)dx \tag{1}$$

subject to

$$\sum_{od\in W}\sum_{p\in P^{od}} f_p^{od}\delta_{p,a}^{od} = x_a, a \in A \tag{2}$$

$$\sum_{p\in P^{od}} f_p^{od} = q^{od}, od \in W \tag{3}$$

$$f_p^{od} \geq 0, od \in W, p \in P^{od} \tag{4}$$

The UE-TAP model (Eqs. (1)–(4)) satisfies the Kuhn–Tucker conditions for the convex program, which has an optimal point (Wardrop 1952). Let $\boldsymbol{f}^{od} = \left(f_p^{od}, od \in W, p \in P^{od}\right)$ represents the vector of path flows for each OD pair, consisting of the shortest path flows $f_{\bar{p}}^{od}$ and the non-shortest path flows $f_p^{od}, p \in P^{od}\backslash\bar{p}$, where $\bar{p} \in P^{od}$ denotes the shortest path. Hence, we can represent the shortest path flows $f_{\bar{p}}^{od}$ in terms of other paths,

$$f_{\bar{p}}^{od} = q_{od} - \sum_{p\in P^{od}\backslash\bar{p}} f_p^{od}, \forall od \in W \tag{5}$$

The link flow x_a can then be expressed as follows:

$$\begin{aligned} x_a &= \sum_{od\in W}\left(f_{\bar{p}}^{od}\delta_{a,\bar{p}}^{od} + \sum_{p\in P\backslash\bar{p}} f_p^{od}\delta_{a,p}^{od}\right), \forall a \in A \\ &= \sum_{od\in W}\left(\left(q^{od} - \sum_{p\in P\backslash\bar{p}} f_p^{od}\right)\delta_{a,\bar{p}}^{od} + \sum_{p\in P\backslash\bar{p}} f_p^{od}\delta_{a,p}^{od}\right) \\ &= \sum_{od\in W}\left(q^{od}\delta_{a,\bar{p}}^{od} + \sum_{p\in P\backslash\bar{p}} f_p^{od}\left(\delta_{a,p}^{od} - \delta_{a,\bar{p}}^{od}\right)\right), \forall a \in A \end{aligned} \tag{6}$$

where the non-shortest path flow can replace the shortest path flow.

Consequently, Eqs. (1)–(4) can be reformulated the path-based UE-TAP model as follows:

$$\min \widetilde{Z}\left(\widetilde{\boldsymbol{f}}\right) = \sum_{a\in A}\int_0^{\sum_{od\in W}\left(q^{od}\delta_{a,\bar{p}}^{od} + \sum_{p\in P\backslash\bar{p}} f_p^{od}\left(\delta_{a,p}^{od} - \delta_{a,\bar{p}}^{od}\right)\right)} t_a(w)dw \tag{7}$$

subject to

$$f_p^{od} \geq 0, od \in W, p \in P^{od}\backslash\bar{p} \tag{8}$$

where $\widetilde{Z}\left(\widetilde{\boldsymbol{f}}\right)$ is a reformulation of Eq. (1), and $\widetilde{\boldsymbol{f}} = \left(f_p^{od}, \forall od \in W, p \in P^{od}\backslash\bar{p}\right)$.

We use the relative gap (RG) as the stop criterion because it ensures that both the path-based and link-based solutions are equally converged in terms of their respective proximity to the optimal value of the UE objective function (Boyce et al. 2004). The RG is non-negative and equal to 0 only at equilibrium solutions, making it a valid gap function.

$$RG = \left|1 - \frac{\sum_{od\in W} c_{\bar{p}}^{od} \times q^{od}}{\sum_{a\in A} x_a \times t_a(x_a)}\right| \tag{9}$$

where $c_{\bar{p}}^{od}$ is the shortest path cost between OD pairs *od*.

In the UE state, the *RG* of each OD is approximately equal to the difference $c_{\max-\min}^{od}$ between the cost of the most expensive path $c_{p_{\max}}^{od}$ and the cost of the cheapest path $c_{\bar{p}}^{od}$ (Rose et al. 1988). Therefore, for each OD pair,

$$RG_{od} = \left|1 - \frac{c_{\bar{p}}^{od} \times q^{od}}{\sum_{p\in P^{od}} c_p^{od} \times f_p^{od}}\right| \tag{10}$$

$$c_{\max-\min}^{od} = c_{p_{\max}}^{od} - c_{\bar{p}}^{od}, p \in P \tag{11}$$

$$\left|c_{\max-\min}^{od} - RG_{od}\right| \leq \varepsilon_2, \varepsilon_2 \geq 0 \tag{12}$$

2.2. Parallel block coordinate descent method for the TAP

The coordinate descent method is effective for smooth problems or those with separable non-smooth terms and separable constraints (Xu 2018).

The path-based UE-TAP model (Eqs. (7)–(8)) can be decomposed into a series of block-based subproblems. Each block-based subproblem consists of a series of independent OD subproblems that can be solved in parallel using the gradient projective (GP) algorithm. When calculating the current block, the values of the other blocks must be fixed. For the i th block,

$$x_a^{(i)} = \sum_{od \in W_i} \left(q^{od} \delta_{a\bar{p}}^{od} + \sum_{p \in P \setminus \bar{p}} f_p^{od} \left(\delta_{ap}^{od} - \delta_{a\bar{p}}^{od} \right) \right), \forall a \in A \tag{13}$$

$$x_a = x_a^{(i)} + \sum_{j \in S, j \neq i} \bar{x}_a^{(j)}, \forall a \in A \tag{14}$$

where $x_a^{(i)}$ represents part of the traffic flow of section a in the i th block and $\bar{x}_a^{(j)}$ indicates that the partial link flow of section a in the j th block is fixed. $x_a = x_a^{(i)} + \sum_{j \in S, j \neq i} \bar{x}_a^{(j)}, \forall a \in A$ indicates that when calculating the link flow $x_a^{(i)}$ in i th block, the partial flows of link a in the other blocks are all fixed, and their sum is also fixed, that is, $\sum_{j \in S, j \neq i} \bar{x}_a^{(j)}$.

Framework of the PBCD method

1 initialization: choose a feasible $\tilde{f}^{(0)} = \left(\tilde{f}_1^{(0)}, \tilde{f}_2^{(0)}, \ldots, \tilde{f}_s^{(0)} \right)$
2 **for** iteration $k = 1, 2, \ldots,$ **do**
3 **for** each block $i = 1, 2, \ldots, s$ **parallel do**
4 Calculate Eqs. (13)–(14)
5 Calculate $\tilde{f}_i^{(k)} = \underset{\tilde{f}_i}{\operatorname{argmin}} \tilde{Z}_i^{(k-1)} \left(\tilde{f}_i \right)$
5 **end for**
Update $\tilde{f}^{(k)} = \left(\tilde{f}_1^{(k)}, \tilde{f}_2^{(k)}, \ldots, \tilde{f}_s^{(k)} \right)$
6 **if** the stopping criterion is satisfied then
7 return
8 **end if**
9 **end for**

By focusing on one feature or block of features at a time during each iteration, BCD allows efficient optimization without dependence on the total number of features, coordinates or variables during an iteration, which helps solve problems with a large number of features (Chauhan et al. 2017). BCD can be adapted to parallel computing to solve the TAP (PBCD) (Chen et al. 2020a; Wang et al. 2021a).

The PBCD method combines the properties of both sequential and parallel computing. Rather than updating a chosen coordinate at each iteration, the PBCD method renews a selected block of coordinates while the other blocks are fixed. When the number of blocks s is equal to the size of the OD pair set $|W|$, we can use the cyclic rule to iterate, or Gauss–Seidel method. When the number of blocks s is equal to 1, we use the Jacobi method.

In **Algorithm 1**, we alternatively fix the paths' cost (used links cost) variables in $\tilde{Z}\left(\tilde{f}_1^{(k)}\right), \tilde{Z}\left(\tilde{f}_2^{(k)}\right), \ldots, \tilde{Z}\left(\tilde{f}_i^{(k)}\right), \ldots, \tilde{Z}\left(\tilde{f}_s^{(k)}\right)$ and solve the block-based subproblems. In each block, the GP algorithm is used to compute multiple OD-based traffic assignment problems in parallel. In other words,

$$\tilde{f}_1^{(k)} = \operatorname{argmin} \tilde{Z}\left(\tilde{f}_1, \tilde{f}_2^{(k-1)}, \ldots, \tilde{f}_i^{(k-1)}, \ldots, \tilde{f}_s^{(k-1)} \right) \tag{15}$$

$$\tilde{f}_2^{(k)} = \operatorname{argmin} \tilde{Z}\left(\tilde{f}_1^{(k)}, \tilde{f}_2, \ldots, \tilde{f}_i^{(k-1)}, \ldots, \tilde{f}_s^{(k-1)} \right) \tag{16}$$

…

$$\tilde{f}_i^{(k)} = \operatorname{argmin} \tilde{Z}\left(\tilde{f}_1^{(k)}, \tilde{f}_2^{(k)}, \ldots, \tilde{f}_{i-1}^{(k)}, \tilde{f}_i, \tilde{f}_{i+1}^{(k-1)}, \ldots, \tilde{f}_s^{(k-1)} \right) \tag{17}$$

…

$$\tilde{f}_s^{(k)} = \operatorname{argmin} \tilde{Z}\left(\tilde{f}_1^{(k)}, \tilde{f}_2^{(k)}, \ldots, \tilde{f}_i^{(k)}, \ldots, \tilde{f}_s \right) \tag{18}$$

Chen et al. (2020a) discovered that the paths between various OD pairs may overlap, leading to accumulated errors from the Jacobi method that could cause significant bias in the descent direction. The main benefit of the BCD method is that it incorporates up-to-date information from each newly updated block into subsequent calculations. However, if the OD pairs are very similar (with heavily overlapping shortest paths), this method may struggle to offer valuable information for the subsequent blocks.

Therefore, the objective function in Eqs. (7)–(8) can be rewritten as a block-based function,

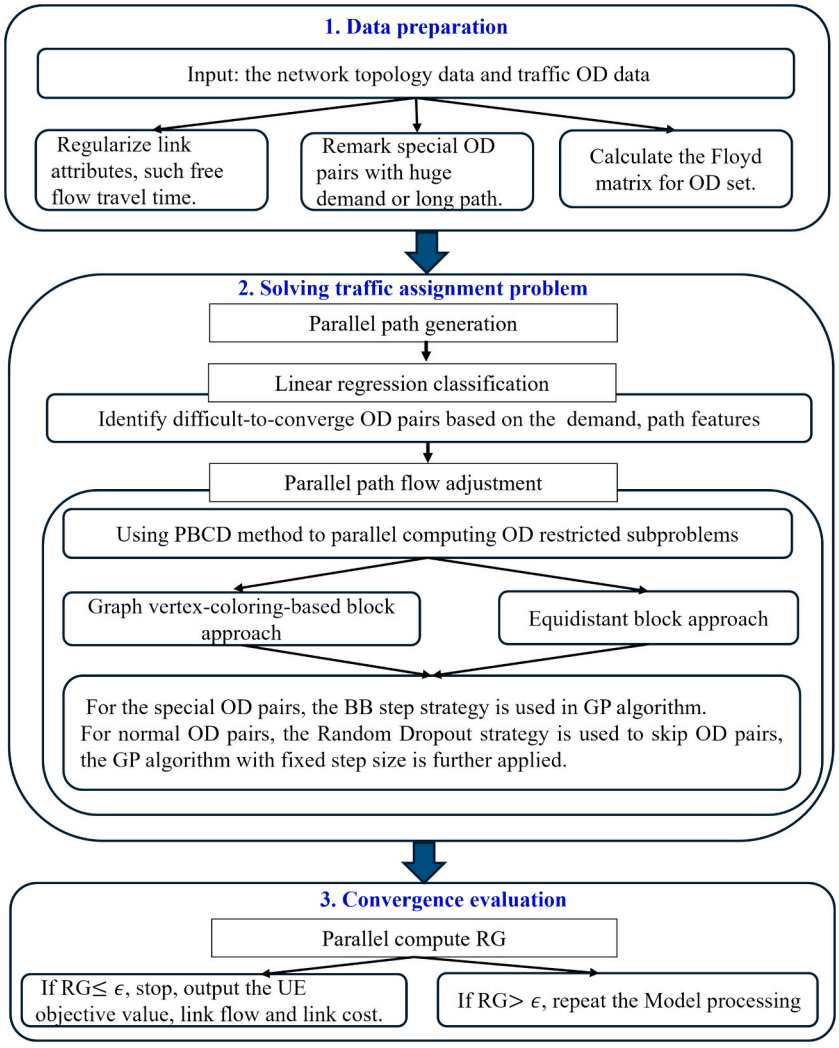


Fig. 1. Proposed LR-PBCD framework for solving TAPs.

$$\min \tilde{Z}\left(\tilde{\boldsymbol{f}}\right) = \sum_{a\in A} \int_0^{\sum_{w_i\in W}\sum_{od\in W_i}\left(q^{od}\delta_{a,\bar{p}}^{od}+\sum_{p\in P^{od}\setminus\bar{p}} f_p^{od}\left(\delta_{a,p}^{od}-\delta_{a,\bar{p}}^{od}\right)\right),\forall a\in A} t_a(w)dw \tag{19}$$

subject to

$$f_p^{od} \geq 0, od \in W, p \in P^{od}\setminus\bar{p} \tag{20}$$

2.3. Block strategies

2.3.1. Equidistant block approach

The equidistant rule is employed to partition the set of OD pairs W to address the issue of overlapping paths between different OD pairs within the same block. In detail, the OD pair set $W(\overline{W})$ is divided into $\lfloor |W|/\overline{m} \rfloor + 1$ blocks, where $\overline{m}$ denotes the block size or the parallel level within blocks. Let $ParaL = \lfloor |W|/\overline{m} \rfloor$ and $W_{i,od}$ denote the index of an OD pair within block i, which starts from 1. An equidistant rule is used to group OD pairs into blocks Chen et al. (2020a). For block $i \in \{1, \ldots, |W|/\overline{m}\}$,

$$W_{i,od} = i + k \times ParaL, k \in \{1, 2, \cdots\} \tag{21}$$

The remaining OD pairs ($|W|$ *mod* m) are grouped into the last block. The OD pair set $W(\overline{W})$ can then be divided into $\lfloor |W|/\overline{m} \rfloor + 1$ blocks. Therefore, the OD pairs in the same block are independent, and these OD subproblems can be computed in parallel in the same block.

2.3.2. Graph vertex-coloring-based block approach

Zhang et al. (2025) introduced a graph vertex-coloring (VC) based approximately uniform OD grouping/block approach to tackle the path overlap issue among OD pairs. They formulated the OD grouping problem as a VC problem and transformed it into an integer linear programming model. The resulting VC-based partitioning model can be formulated as follows:

$$\min \sum_{c=1}^{|C|} y_c \tag{22}$$

subject to

$$\sum_{c=1}^{|C|} \varphi_{od_m}^{c} = 1, \forall od_m \in W \tag{23}$$

$$\varphi_{od_m}^{c} + \varphi_{od_n}^{c} \le 1, \forall (od_m, od_n) \in A', \forall c \in C \tag{24}$$

$$\varphi_{od_m}^{c} \le y_c, \forall od_m \in W, \forall c \in C \tag{25}$$

$$\varphi_{od_m}^{c} \in \{0,1\}, \forall od_m \in W, \forall c \in C \tag{26}$$

$$y_c \in \{0,1\}, \forall c \in C \tag{27}$$

$$\left| |B_{c_i}| - |B_{c_j}| \right| \le \kappa, i \ne j, c_i \in C, c_j \in C \tag{28}$$

where the objective function (22) minimizes the total block overlap rather than the minimum number of colors used, and y_c denotes the usage of color c. Constraint (23) requires that each vertex is assigned a unique color, while constraint (24) ensures that adjacent vertices do not share the same color. $\varphi_{od_m}^{c} = 1$ if OD belongs to the c th block, otherwise $\varphi_{od_m}^{c} = 0$. Constraint (25) indicates that if color c is used, at least one vertex must be colored. Constraints (26) and (27) denote that $\varphi_{od_m}^{c}$ and y_c are binary variables. The number of OD-based vertices in each color is defined as $|B_c| = \sum_{od_m \in W} \varphi_{od_m}^{c}, \forall c \in C$. Eq. (28) indicates that the difference in the number of elements between any two color/block sets, $|B_{c_i}|$ and $|B_{c_j}|$, is less than κ.

3. Using the LR-PBCD method to solve TAPs

This section presents novel insights from the application of the LR-PBCD method to enhance the efficacy of TAP models. The proposed framework comprises data preparation, traffic assignment problem-solving, and convergence evaluation phases, as illustrated in Fig. 1, with a focus on the data preprocessing and iterative calculation components. The data preprocessing phase analyzes the FFTT, path number, average path length, and spatial location between OD pairs.

3.1. Feature distribution and classification

Traffic assignment models rely on two key inputs: road network information (including topology and link attributes) and travel demand (of OD pairs). These generate the primary outputs of expected traffic flow and travel time for vehicles across each road or link for traffic assignment patterns (Roocroft et al. 2023).

3.1.1. Feature extraction of link attributes

As traffic data have multiple features spanning varying degrees of magnitude, range, and units, it is necessary to carry out feature scaling before applying data processing techniques (Chen et al. 2021). The UE assignment is a Nash equilibrium, where paths are selected to ensure that each user's impedance (e.g., vehicle hours of travel) between any OD pair is at equilibrium. The issue is that the impedance of each.

link is uncertain until after equilibrium is reached. Because the impedance function uses equilibrium volumes, it can be solved iteratively by estimating the volume delay function (VDF), which is then used to update the link impedances between iterations. A precisely specified VDF reaches equilibrium in fewer iterations, thereby improving algorithm efficiency.

The FFTT, t_a^0, of the links set should follow the 68%/95%/99.7% rule for normal distribution,

$$f(t_a^0) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(t_a^0-\mu)^2}{2\sigma^2}} \tag{29}$$

and the normal distribution for the FFTT is

$$t_a^0 \sim N(\mu, \sigma^2) \tag{30}$$

where μ is the mean or expected distribution (and its media), while the parameter σ is its standard deviation. The variance of the distribution is σ^2.

The current form of the VDF is a form of the Bureau of Public Roads (BPR) function,

$$t_a(x_a) = t_a^0\left(1 + \alpha(x_a/c_a)^\beta\right) \tag{31}$$

where the FFTT is typically represented as the time when the traffic flow is at 0, t_a^0.

To facilitate description and application, Eq. (30) is used to convert the general normal distribution to the standard normal distribution (Arezoumandi 2011; Guessous et al. 2014), which is expressed as

$$\bar{t}_a^0 = \left|\left(t_a^0 - \mu\right)/\sigma\right| \tag{32}$$

$$\bar{t}_a^0 \sim N(0,1) \tag{33}$$

Feature scaling, such as normalization, can expedite the solution speed of gradient descent and enhance the convergence speed of the model when used to solve optimization problems (Salimans and Kingma 2016).

The scaling parameter ε_1 of the FFTT t_0 in the VDF is normalized to match the magnitude of the link capacity c_a. Therefore, the BPR function is written as follows:

$$t_a'(x_a) = \varepsilon_1 \cdot \bar{t}_a^0\left(1 + \alpha(x_a/c_a)^\beta\right), \varepsilon_1 > 0 \tag{34}$$

Then, the objective function Eq. (19) is rewritten as

$$\min\widetilde{Z}\left(\widetilde{\boldsymbol{f}}\right) = \sum_{a\in A}\int_0^{\sum_{w_i\in W}\sum_{od\in w_i}\left(q^{od}\delta_{a,\bar{p}}^{od}+\sum_{p\in P,\bar{p}}f_{\bar{p}}^{od}\left(\delta_{a,p}^{od}-\delta_{a,\bar{p}}^{od}\right)\right),\forall a\in A}\frac{1}{\varepsilon_1}t_a'(w)dw \tag{35}$$

3.1.2. Feature extraction for OD pair attributes

Applying data processing techniques enables the effective use of critical information by allocating more iterations to subproblems with essential features (Dhal and Azad, 2022). The path features of an OD pair significantly influence the UE traffic flow (Xie et al. 2018). In general, assuming fixed travel demand and volume delay function, calculating an OD pair with longer paths requires more time than calculating one with shorter road sections. Hence, during the initialization phase, the lengths of the first path of each OD pair are computed, and OD pairs with path lengths exceeding the average length of the path set are identified. The formulas are as follows:

$$L_p^{od} > \beta_2 * median\left(LE_{\bar{p}}^{od_1}, LE_{\bar{p}}^{od_2}, \ldots\right), od \in W, \bar{p} \in P^{od}, \beta_2 \geq 1 \tag{36}$$

$$Remark_{od} = \begin{cases} 1, & case \\ 0, & elsewise \end{cases}, od \in W \tag{37}$$

where LE_p^{od} denotes the length (total number of used links) of path p between OD pair od, $p \in P$.

To streamline the iteration process and minimize redundant calculation, we calculate and store only the length of the initial path during the initialization phase. This approach ensures that subsequent iterations rely on precomputed path lengths, thus enhancing computational efficiency.

3.1.3. Using a linear regression approach to classify OD pairs

We employ a linear regression model to learn the mapping between features of known not-yet-converged OD pairs, $RG_{od} > 0$, and their convergence gap, enabling the prediction of potentially difficult-to-converge OD pairs.

First, all not-yet-converged OD pairs are selected as training samples. The relationship between their features—including demand D^{od}, number of paths $\left|P_{od}^{(k)}\right|$, average path length $\frac{\sum_{p\in P_{od}^{(k)}} Len_p^{od}}{\left|P_{od}^{(k)}\right|}$, and the Hessian indicator $Hessian_{od}$—and the actual relative gap RG_{od} is modeled via linear regression as:

$$\begin{aligned}\widehat{y}_{od}^{(k)} &= \beta_0 + \beta_1 \cdot D^{od} + \beta_2\left|P_{od}^{(k)}\right| + \beta_3 \cdot \frac{\sum_{p\in P_{od}^{(k)}} Len_p^{od}}{\left|P_{od}^{(k)}\right|} \\ &\quad + \beta_4 \cdot Hessian_{od} + \varepsilon_{od}^{(k)}, RG_{od} > 0, od \in W,\end{aligned} \tag{38}$$

where k represents the number of iterations of the main loop; β_0 represents the intercept term, that is, the constant term in the linear regression model; $\beta_1, \beta_2, \beta_3, \beta_4$ represent the feature weight coefficients, that is, the weights of each feature in the linear regression model obtained through the least squares method; and ε_{od} represents the error term, that is, the difference between the actual value and

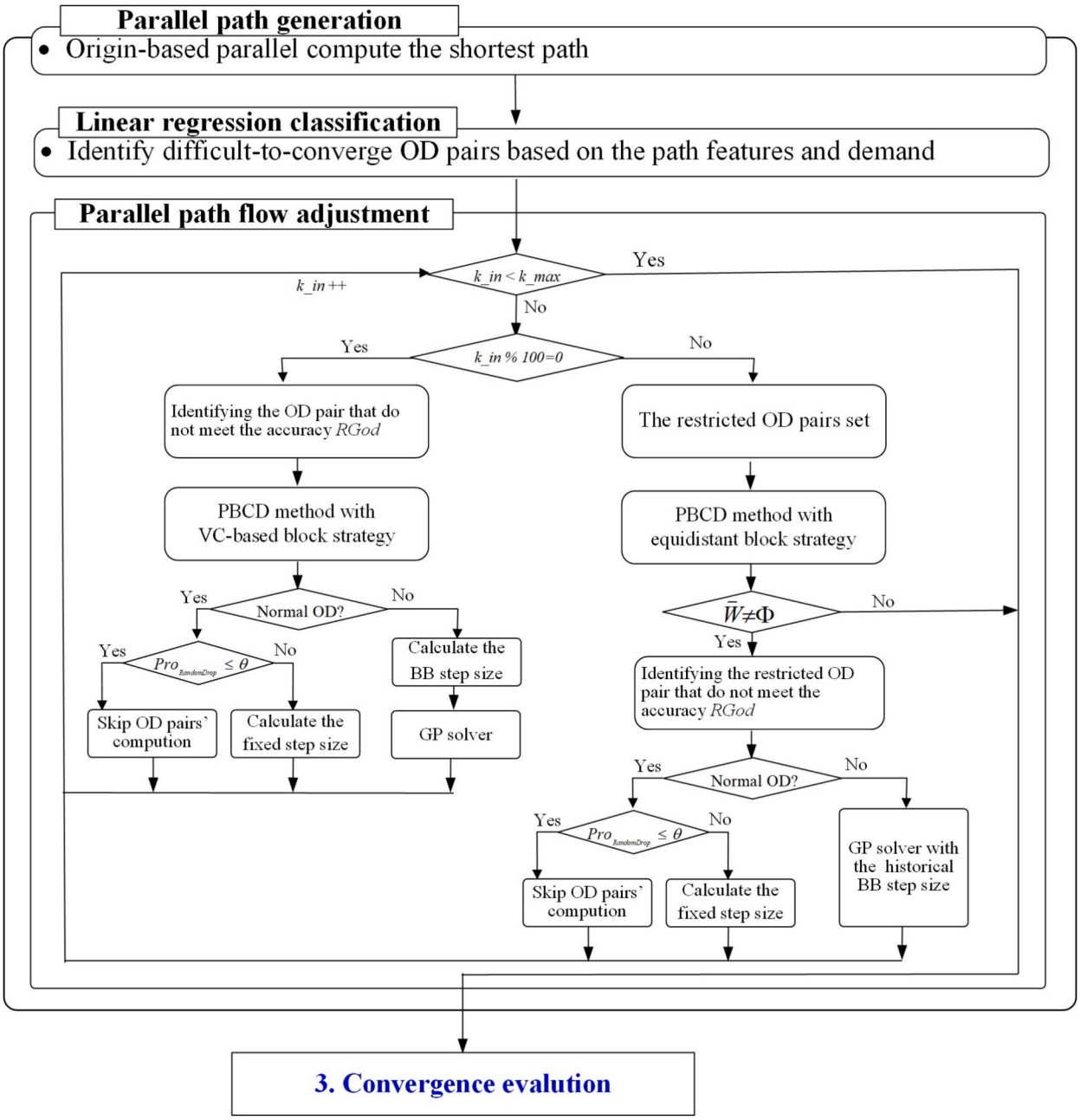


Fig. 2. The parallel path generation and adjustment of LR-PBCD framework.

the predicted value. $Hessian_{od}$ represents the second-order derivative of the OD-based traffic assignment model, $Hessian_{od} = \frac{\partial^2 \widetilde{Z(f)}}{\left(\partial \widetilde{f}_p^{od}\right)^2} = \sum_{a \in A} \widetilde{t}_a'(v_a)\left(\delta_{a,p}^{od} - \delta_{a,\bar{p}}^{od}\right)^2$. For the detailed derivation of the Hessian matrix, please refer to Zhang et al. (2023a).

To enhance the model fitting performance, all features are normalized to mitigate scale and magnitude differences.

$$\overline{x} = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{39}$$

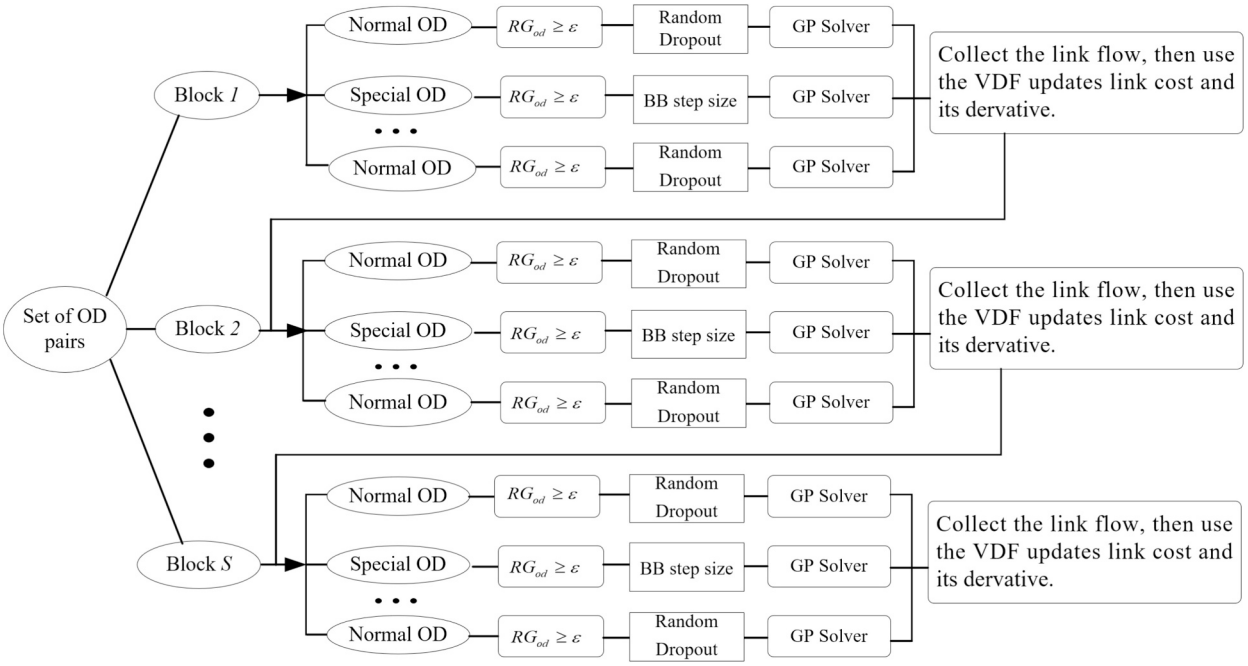


Fig. 3. Parallel computation of the OD path flow in each block.

$$\widehat{y}_{od}^{(k)} = \beta_0 + \beta_1 \cdot \overline{(D^{od})} + \beta_2 \overline{(|P_{od}^{(k)}|)} + \beta_3 \cdot \overline{\left(\frac{\sum_{p \in P_{od}^{(k)}} Len_p^{od}}{|P_{od}^{(k)}|} \right)} + \beta_4 \cdot \overline{Hessian_{od}} + \varepsilon_{od}^{(k)}, RG_{od} > 0, od \in W. \tag{40}$$

$$\overline{x} \in [0,1]$$

Next, using the trained regression coefficients $\beta_i, i = \{0,1,2,3,4\}$, the model predicts the convergence difficulty for OD pairs with $RG_{od} = 0$ (those already converged), yielding estimated values:

$$\widehat{y}_{od}^{(k)} = \beta_0 + \beta_1 \cdot \overline{(D^{od})} + \beta_2 \overline{(|P_{od}^{(k)}|)} + \beta_3 \cdot \overline{\left(\frac{\sum_{p \in P_{od}^{(k)}} Len_p^{od}}{|P_{od}^{(k)}|} \right)} + \beta_4 \cdot \overline{Hessian_{od}}, RG_{od} = 0, od \in W. \tag{41}$$

These predicted values $\widehat{y}_{od}^{(k)}$ reflect the potential convergence risk of OD pairs in the model feature space.

Furthermore, combining the observed non-converged OD pairs ($RG_{od} > 0$) with a selected subset of ordinary OD pairs having high predicted convergence difficulty, these latter pairs are also labeled as "special OD", which are defined as hard-to-converge OD pairs. This selection is based on the rationale that OD pairs with higher predicted values tend to exhibit larger path cost deviations or more complicated traffic states, making them more prone to convergence difficulties during computation. The labeling rule is formalized as:

$$Label_{od} = \begin{cases} 1, RG_{od} > 0 or \widehat{y}_i > \theta & (SpecialODs) \\ 0, \quad otherwish & (normalODs) \end{cases}, \tag{42}$$

where θ is a predefined threshold used to identify ordinary OD pairs with sufficiently high predicted convergence risk to be classified as special.

The R^2 score (also known as the coefficient of determination) is an important indicator used in statistics to evaluate the goodness of fit of regression models, reflecting the proportion of the variation in the dependent variable that the independent variable can explain. We use R^2 to measure the performance of the model:

$$R^2 = 1 - \frac{\sum_{od\in W}\left(RG_{od} - \widehat{y}_{od}^{(k)}\right)^2}{\sum_{od\in W}\left(RG_{od} - \overline{y}_{od}^{(k)}\right)^2} \tag{43}$$

where $\widehat{y}_{od}^{(k)}$ denotes the predicted value of an OD pair; $\overline{y}_{od}^{(k)}$ denotes the average value of all pre-calculated RG_{od}, $\overline{y}_{od}^{(k)} = \frac{1}{|W|}\sum_{od\in W} RG_{od}$. The value range of the R^2 score is between 0 and 1. The closer it is to 1, the better the model fit; the closer it is to 0, the worse the model fits.

3.2. LR-PBCD algorithm for solving TAP

The process of solving traffic assignment problem consists of two phases: parallel path generation and parallel path flow adjustment phases, as illustrated in Fig. 2.

First, the parallel path generation phase follows the approach of Zhang et al. (2023b), enabling tasks to be executed concurrently using the Dijkstra algorithm with OpenMP libraries. Second, a linear regression method is applied to classify OD pairs as either normal or hard-to-converge. Third, we set the maximum number of inner iterations, *K_max*, to 1,000; that is, for each path generation, 1,000 inner iterations are performed. Specifically, in each 100 inner iterations, a VC-based block strategy (Eqs. –) is applied to all OD pairs $\overline{W}$ (when $K\_in\ \%\ 100 = 0$), while in the remaining 99 inner iterations, an equidistant block strategy (Eq.) is used for the restricted subset of OD pairs $\overline{W}$. For each block-based subproblem, parallel libraries are employed to solve multiple OD subproblems simultaneously. This section highlights the application of data processing techniques within the parallel path flow adjustment phase.

3.2.1. Parallel path flow adjustment

Zhao et al. (2014) pointed out that the randomized BCD method can accelerate block processing. As shown in Fig. 3, for special OD pairs, the BB step strategy is used to improve the step size for the GP algorithm, which adjusts the path flow between an OD pair as follows: for normal OD pairs, the random dropout strategy is used to skip OD pairs, and the GP algorithm with a fixed step size is applied to adjust the path flow.

The data processing strategies filter multiple OD pairs first, then apply parallel computing using the GP solver. The link flow is mapped from the calculated OD pairs' path flow by the link-path incidence matrix, and the link cost is then updated by the BPR function. The updated formation is then transferred to the next block.

3.2.2. GP algorithm with BB step size

The GP algorithm is a popular optimization method to determine optimal model parameters (Bottou 2010; Wang et al. 2021b). For each block-based problem, the parallel libraries (OpenMP) are used to implement multiple OD subproblems, and the GP algorithm is then applied to parallel adjust the traffic flow on the path between an OD pair (Zhang et al. 2023b), as follows:

$$f_p^{od}(k+1) = \max\left\{0, \left[f_p^{od}(k) - \alpha^{od(k)} \times \frac{\partial}{\partial f_p^{od}}\widetilde{Z}\left(\widetilde{\boldsymbol{f}}\right) \Big/ \frac{\partial^2 \widetilde{Z}\left(\widetilde{\boldsymbol{f}}\right)}{\left(\partial f_p^{od}\right)^2}\right]\right\}, p \in P^{od}\backslash\overline{p},\ od \in W_i \tag{44}$$

$$f_{\overline{p}}^{od}(k+1) = q_{od} - \sum_{p\in P^{od}, p\neq\overline{p}} f_p^{od}(k+1),\ \forall od \in W_i \tag{45}$$

For normal OD pairs, we use the fixed step size $\alpha^{od(k)} = 0.3$. We adopt the random dropout strategy proposed by Wang et al. (2021a) to discard a certain proportion of OD pairs. The random dropout approach reduces the likelihood of path overlapping by selectively updating specific OD pairs within a block. When the probability X exceeds $\overline{X}$, we stop adjusting the flow between this OD pair.

$$Pro_{RandomDrop} = Pro(X < \overline{X}) \tag{46}$$

where X is a random value in the range [0, 1], and $\overline{X}$ is the fixed threshold for random dropout.

For special (hard-to-converge) OD pairs, we use the BB step size algorithm to obtain $\alpha^{od(k)} = \alpha_{BB}^{od(k)}$. The BB step size algorithm determines the step size using information from the last two iterations, and requires no additional function evaluations. This approach avoids parameter setting, making it more practical for large-scale transportation networks. For each OD pair, the BB step size can be calculated as follows:

$$\alpha_{BB}^{od(k)} = \frac{\sum_{p\in P}\left(f_p^{od(k)} - f_p^{od(k-1)}\right)^2}{\sum_{p\in P}\left(f_p^{od(k)} - f_p^{od(k-1)}\right)\left(c_p^{od(k)} - c_p^{od(k-1)}\right)} \tag{47}$$

where $f_p^{od(k)} - f_p^{od(k-1)}$ represents the path flow vector and $c_p^{od(k)} - c_p^{od(k-1)}$ represents the gradient vector. The numerator is the sum of

Table 3
Open-source transportation networks.

Network	Node	Link	OD	Demand	Block size
Anaheim (Ana)	416	914	1,406	104,695	[124, 143]
Chicago-Sketch (CS)	933	2950	93,135	113,750	[4852, 4953]
Birmingham (Birm)	14,639	33,937	470,805	565,201	[2890, 3780]
Philadelphia (Phi)	13,389	40,003	1,149,795	13,037,834	[1784, 1889]
Chicago Regional (CR)	12,982	39,018	2,296,227	1,135,990	[2013, 2545]

Notes: To avoid the double meaning of the comma, the commas within the numbers have been omitted here. "OD pairs [min, max]" indicates the range of ODs in each block; the number of OD pairs in each block represents the maximum number of threads that can be used for parallel computation within each block. The data can be downloaded from https://github.com/bstabler/TransportationNetworks.

flow vectors across all paths for the OD pair; the denominator is the inner product of the flow vector, and the gradient vector is summed across all paths for the OD pair. The detailed process and parameter setting for the BB step algorithm are provided in Appendix A.

3.2.3. Solving algorithm

For special OD pairs, the BB step strategy is used to improve the step size in the GP algorithm, which adjusts the path flow between OD pairs. For normal OD pairs, the random dropout strategy is employed to skip these pairs, and the GP algorithm with a fixed step size is then applied to adjust the path flow. Furthermore, the proofs of convergence for the proposed method are provided in Appendices B and C.

The LR-PBCD method for solving TAPs can be defined as follows.

Algorithm 1: LR-PBCD method for solving TAPs

Initialization: choose a feasible $\left(\tilde{\boldsymbol{f}}_1^{(0)}, \tilde{\boldsymbol{f}}_2^{(0)}, \ldots, \tilde{\boldsymbol{f}}_s^{(0)}\right)$
Reform the BPR function by Eqs. (32)-(34)
for iteration $k = 1, 2, \ldots,$ **do**
  Parallel path generation
  Remark about the OD pair with long path by Eqs. (36)-(37)
  Use linear regression model (Eq. (38)-(43))to classify OD set to special ODs and normal ODs
  for each block $i = 1, 2, \ldots, s$ **do**
    Parallel compute the OD pairs using the PBCD framework
    If the OD pair is special, **do**
      Calculate the BB step size
      Calculate the GP algorithm (Eqs. (44)–(45)) with the BB step size
    If the OD pair is normal, **do**
      If $Pro_{RandomDrop} \leq \overline{Sim}$ **do**
        Skip the computation of OD pairs
      Otherwise
        Run the GP algorithm with a fixed step size
    Gather the link flow from the updated OD path flow using Eqs. (13)–(14)
    Calculate the link cost and link cost derivative by the BPR function Eq. (31)
    Update the block information by Eqs. (15)-(18)
  end for
  Thus, $\tilde{\boldsymbol{f}}^{(k+1)}$ equals the updated values of the last block, $\tilde{\boldsymbol{f}}^{(k+1)} = \tilde{\boldsymbol{f}}_s^{(k)}$
  if the stopping criterion Eq. (9) is satisfied **do**
    Return $\left(\tilde{\boldsymbol{f}}_1^{(k)}, \tilde{\boldsymbol{f}}_2^{(k)}, \ldots, \tilde{\boldsymbol{f}}_s^{(k)}\right)$

Note that $\overline{L}_j (j = 1, 2, \cdots, s)$ is the upper bound on the Lipschitz constant $L_j \left(\overline{L}_j \geq L_j\right)$. A GP step with a constant step size is performed at each iteration for a different block of variables, applied in a cyclic order.

4. Numerical experiments

In this section, we report on the performance of the proposed method as evaluated on five open-source traffic networks, as shown in Table 3. We used the vertex-coloring based OD pairs block result generated in our earlier work (Zhang et al. 2025). The largest degree first algorithm was used to solve the path overlapping problem, thereby enabling the identification of OD pairs within each block with minimal path overlap. The experiments were conducted on a Linux computer with 128 GB DDR4 memory and an Intel Xeon E5-2660 2.20 GHz CPU.

The proposed algorithm and several traditional comparison algorithms, including the path-based methods such as iGP (Xie et al. 2018), PBCD (Chen et al. 2020a), iPBCD (Wang et al. 2021a), Dy-PBCD (Liu et al. 2025), and the origin-based iTAPAS (Xie and Xie 2016; Xie and Nie, 2024), were implemented using C++. The OpenMP library was used in parallel computing. Multiple experiments were performed for each case to mitigate errors arising from CPU computation instability. The *RG* (Eq.(9)) was set to be less than or

Table 4
Transportation network link features.

	Feature	Ana	CS	Birm	Phi
FFTT(min)	links count	912	2,950	33,938	40,004
	Link's FFTT = 0	0	774	24	9,802
	Average value	0.88	3.38	1.12	0.89
	Minimum value	0.05	0.00	0.00	0.00
	Maximum value	3.58	24.92	122.39	18.13
Norm distribution	μ	0.88	3.88	1.12	0.89
	σ	0.64	2.97	3.67	1.16

Note that the FFTT of these networks is measured in minutes. μ, σ denote the mean and standard deviation, respectively, of the normal distribution of FFTTs.

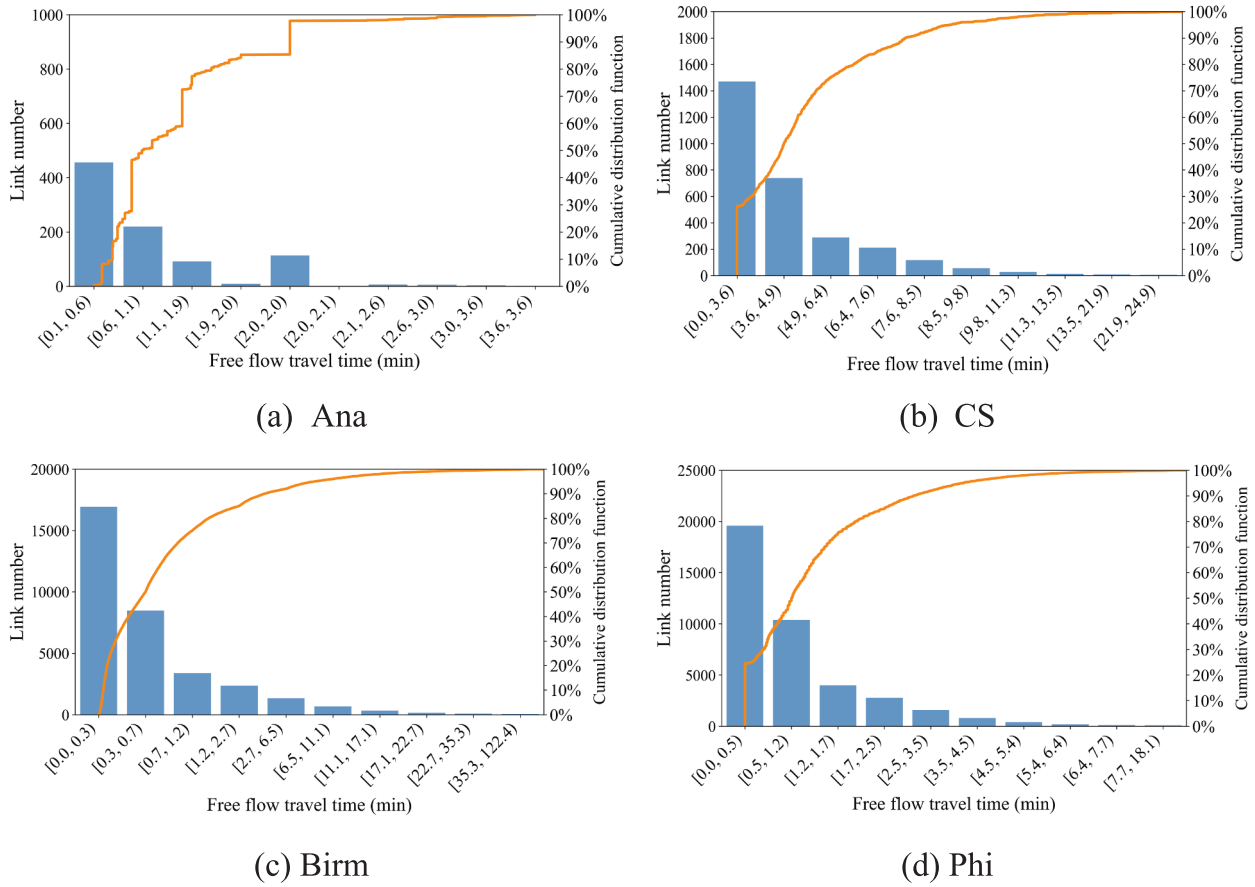


Fig. 4. Cumulative distribution result for the FFTT.

equal to 1.0E-12 as the convergence stop indicator of the algorithm.

The speedup metric is critical for assessing the efficacy of parallel computing implementations. Speedup, denoted as $SP(k)$, is defined as the ratio of the execution time with a single thread (T_1) to the execution time with k threads (T_k).

$$SP(k) = \frac{T_1}{T_k} \tag{48}$$

As the number of threads increases, the computational overhead associated with data communication also increases. Consequently, the speedup function is expected to exhibit sublinear behavior for a number of threads.

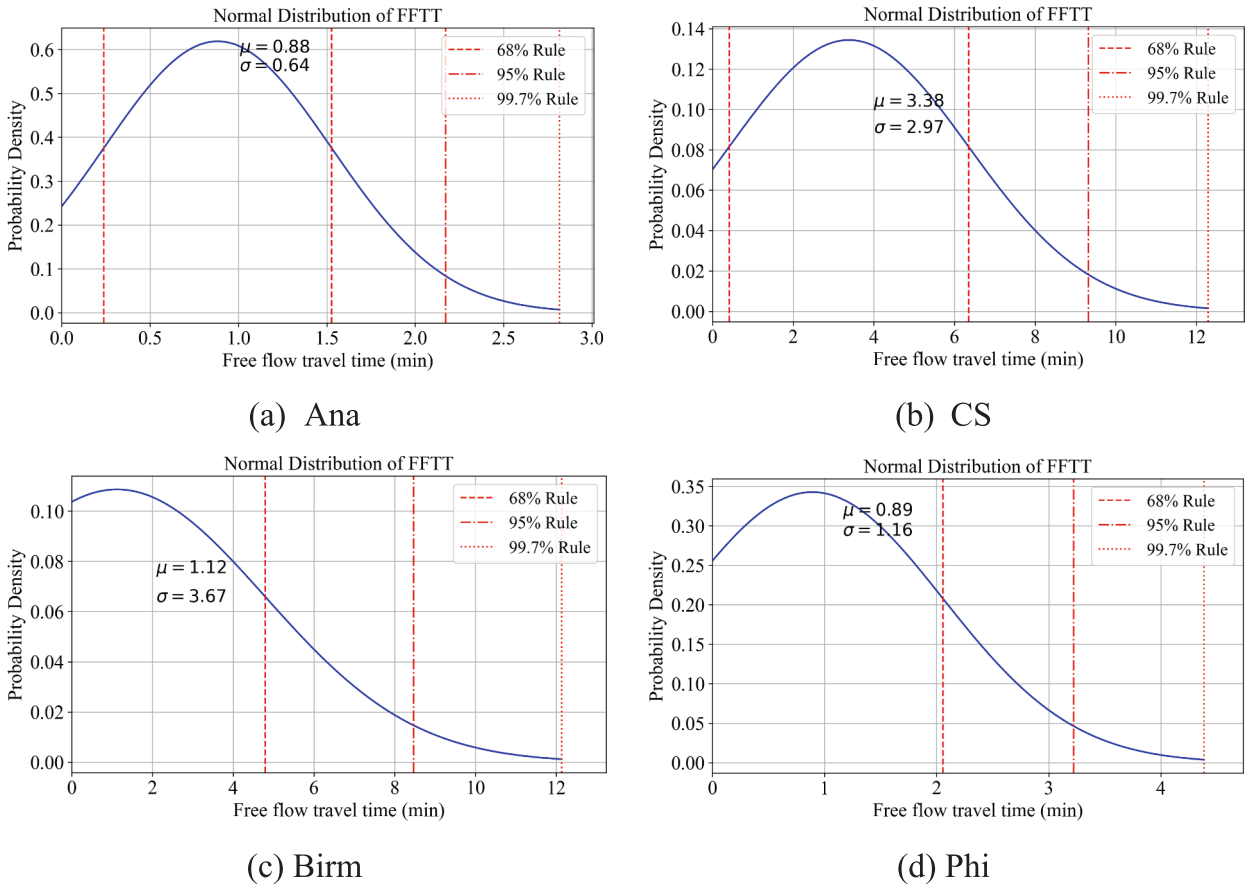


Fig. 5. Normal distribution curve of the FFTT.

4.1. Analysis of data-driven optimization strategies based on data features

We analyzed the features of the input data to identify OD subproblems that necessitate more iterative computation to achieve high-precision convergence from a data-driven standpoint. Studies conducted with the Ana, Birm, Phi, and CR networks revealed that OD subproblems with longer paths that start and end at city boundaries or downtown areas necessitate increased computation to achieve high-precision outcomes.

4.1.1. Regularizing the FFTT

Table 4 displays a statistical analysis of the FFTT characteristics of all of the traffic network links, showing the quantity, mean, maximum, and minimum values and instances where FFTT is 0.

Fig. 4 presents the cumulative distribution function (CDF) analysis employed to evaluate the FFTT on each link of the traffic network. Fig. 5 depicts the normal distribution of the FFTT across all links, allowing μ and σ to be determined.

The highest point of a normal distribution curve corresponds to the maximum value of the probability density function. For the standard normal distribution (with $\mu = 0$ and $\sigma^2 = 1$), the probability density function value at the highest point is approximately 0.40. This mathematical expectation corresponds to the peak of the normal distribution curve, indicating where the probability density is highest in the vicinity of that point. As observed from Fig. 5, the specific numerical values of the mathematical expectation (ME) in the standard distribution curves of the FFTT for the three traffic networks are approximately 1, except for the CS network, which has an ME of 3.38.

In Fig. 6, the computational efficiency of PBCD with normalization is compared with the original PBCD for four transportation networks. The standard normal distribution is applied to the Ana and Phi networks to scale the values of the FFTT by Eq., as shown in Fig. 6 (a) and (d). The FFTT over the mean is applied to the CS and Birm networks, as shown in Fig. 6 (b) and (c). The algorithm's computation for the Ana network used a single thread, while the computations for the other three networks employed 20 threads.

When the result of the TAP for a transportation network reaches a high level of precision, the computation time for the small-sized Ana network accelerates by 0.43 s (45.5%), that of the medium-sized CS network by 0.2 s (8.7%), and those of the large-sized Birm and

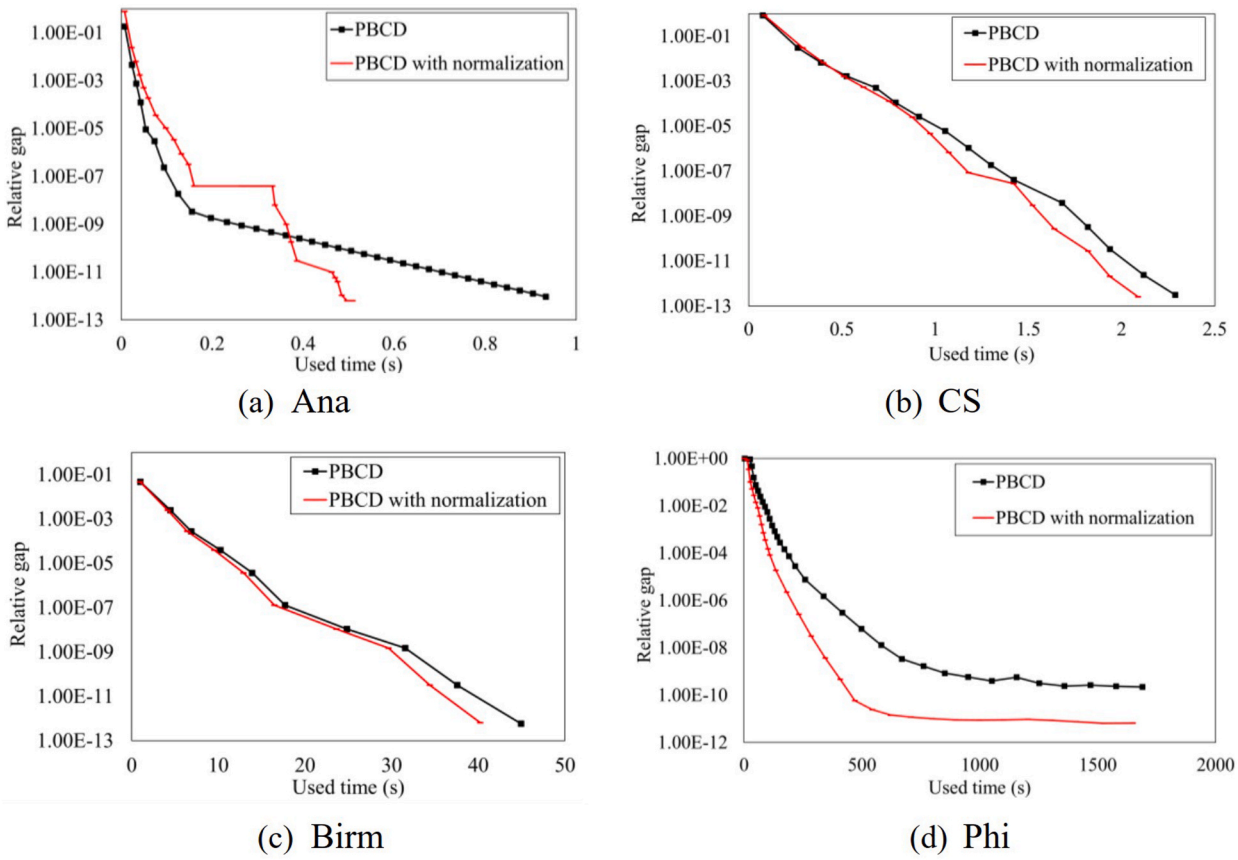


Fig. 6. Convergence of the TAP.

Table 5
OD pairs with the top 20 most calculated counts.

	Ana	CS	Birm	Phi
Top 20 OD pairs' average count	20,340	498	1,358	23,736
Top 20 OD pairs' average path length	18	10	51	36
Top 20 OD pairs' average path number	2.35	2.25	2.20	4.55
Top 20 OD pairs' Location	Countryside	Downtown	Downtown	Countryside and Downtown
Congestion areas (The green area in Fig. 7)	Several main roads	Several main roads	The city center	Almost the entire city

Phi networks by 0.64 s (14.1%) and 642.84 s (61.17%), respectively. Fig. 6 illustrates that normalizing the FFTT numerical values can improve the algorithm's computational efficiency.

4.1.2. Analysis of OD subproblem features

(1) Analyzing the characteristics of the top 20 OD pairs based on calculated count.

We examined the top 20 OD pairs in Table 5 based on computation frequency (calculated count), emphasizing travel demand, computation count/frequency, number of paths, average path length, and OD locations. Fig. 7 displays the service level of the road as measured by link flow over link capacity. It was found that OD pairs with longer paths necessitate more iterations. Details of the feature extraction for the top 20 OD pairs are provided in Appendix D.

Fig. 8 shows that the *RG* of each OD (RG_{od}) has a lower convergence accuracy by an order of magnitude than the difference $c^{od}_{\max-\min}$ between the cost of the most expensive path and the cost of the cheapest path in the UE state.

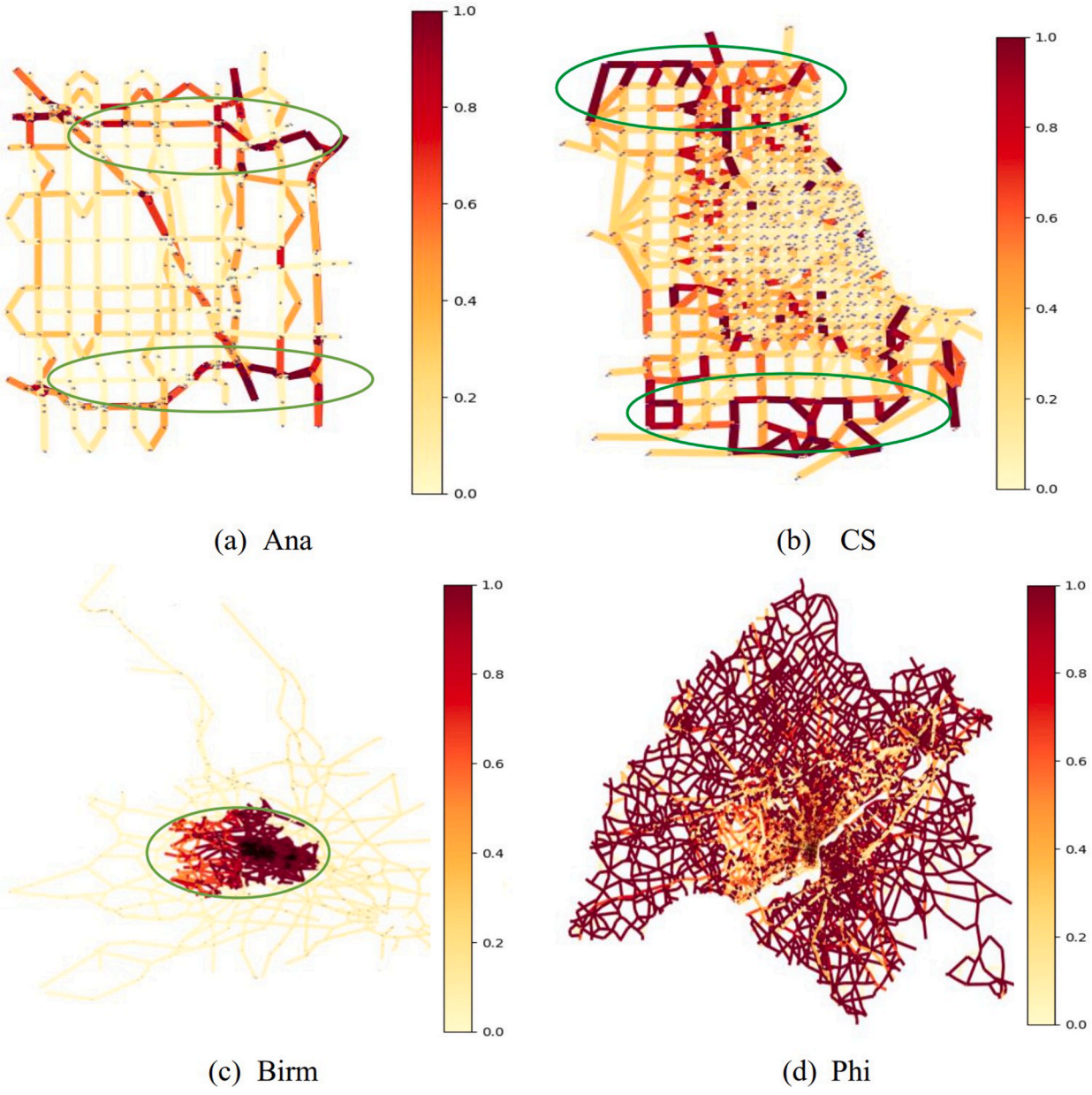


Fig. 7. Service level of a road as measured by link flow over link capacity.

(2) Subproblems of OD pairs with longer paths exhibit an increase in inner iterations.

In Fig. 9, augmenting inner iterations for OD pairs with lengthy paths enhances their algorithmic efficiency by 2.2% on the Ana network. Similarly, on larger-scale networks like CS, Birm, and Phi, increasing the number of additional inner iterations for long-distance OD pairs yields efficiency gains of 6.71%, 21.69%, and 8.21%, respectively.

4.2. Sensitivity analysis of LR-PBCD

4.2.1. Path information

This subsection examines how the number of selected routes during path generation affects the OD classification accuracy of the LR-PBCD method, thereby evaluating the sensitivity of this critical parameter. Note that "Loop1″ in "LR-PBCD_Loop1″ indicates that the linear regression model performs grouping after the first path generation. Similarly, "Loop2″ indicates that the linear regression model performs grouping after the second path generation. The resulting classification is then used throughout the subsequent computations,

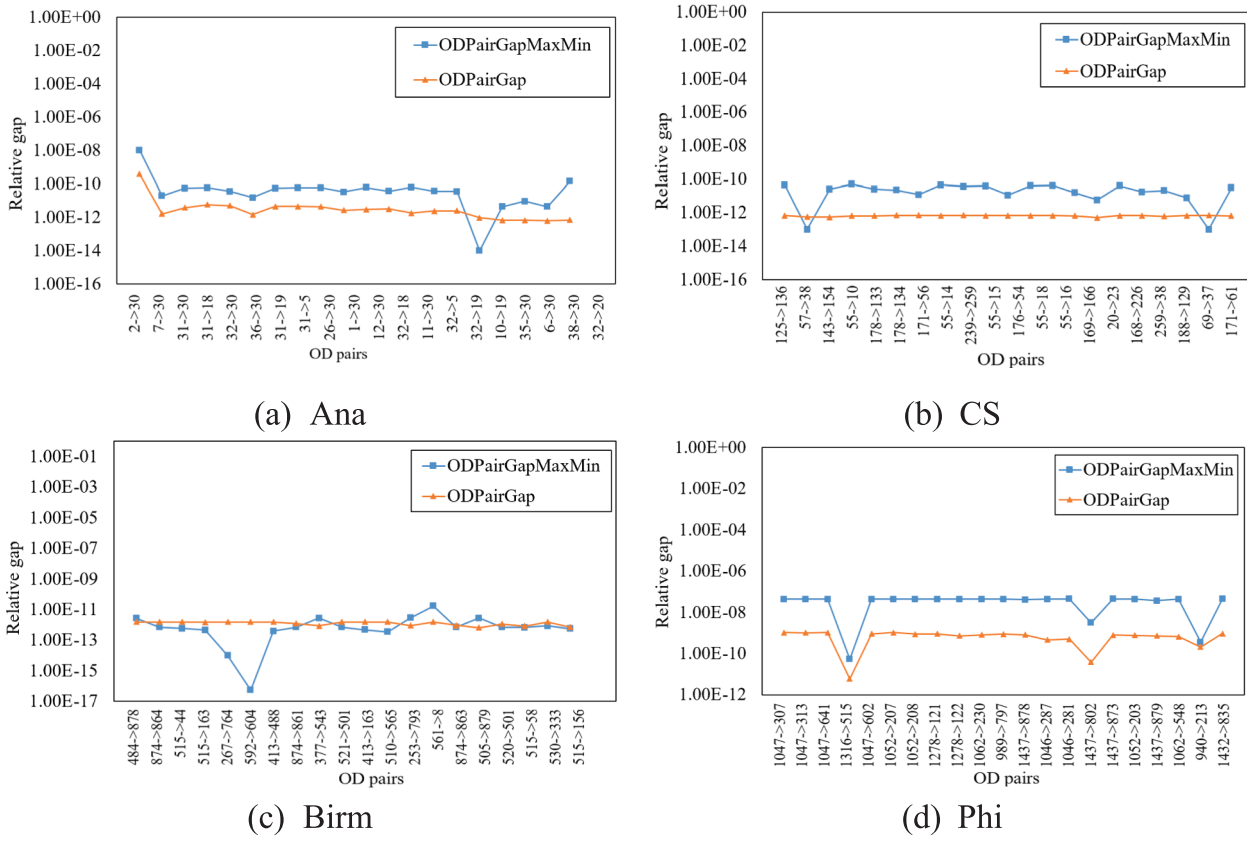


Fig. 8. Relative gap of the OD pairs with the top 20 most calculated counts. (Note that ODPairGap represents Eq. (10) and ODpairGapMaxMin denotes Eq. (11)).

with the BB step size algorithm explicitly applied only to the identified hard-to-converge OD pairs.

As shown in Table 6 (a), the R^2 scores reveal significant variations in linear regression classification performance across different transportation networks, with most networks achieving optimal performance at Loop 2. Ana demonstrates the strongest classification performance with its highest R^2 score of 0.28 at Loop 2, significantly outperforming other loop configurations. This superior performance can be attributed to Ana's relatively simple network structure (416 nodes, 914 links) and moderate complexity (1,406 OD pairs), which creates clearer patterns for distinguishing hard-to-converge pairs after moderate exploration. Phi and CR follow similar trends, both achieving their peak R^2 scores at Loop 2 (0.17 and 0.18 respectively), indicating that these large-scale networks with over 1 million OD pairs benefit from the optimal balance between exploration depth and classification accuracy provided by two path generation iterations. Birm network presents an exception, achieving its highest R2 score of 0.18 at Loop 1, suggesting that this network's convergence patterns become established early in the optimization process and additional exploration introduces noise rather than useful classification information. CS consistently shows poor performance across all loops (maximum R^2 of 0.05), likely due to its sparse demand matrix with extremely high OD-to-demand ratio.

Furthermore, Table 6(b) and 6(c) present the computing times of the LR algorithm for different numbers of loops and different quantities of special ODs, respectively. Our analysis reveals that as the network scale increases, the time required by LR to identify special ODs grows only slightly. Notably, for large-scale networks such as Phi and CS, the proportion of LR computing time remains below 3%.

Meanwhile, we present the computing times of the LR-PBCD algorithm in Figs. 10 and 11 for different numbers of loops and different quantities of special ODs, respectively. It should be noted that the computing times shown in Figs. 10 and 11 do not include the time taken by LR to identify special ODs. The results in Fig. 10 show that, on the Ana network, the computation time of LR-PBCD with two loops (LR-PBCD_Loop2) takes 47% less time than LR-PBCD_Loop1. Similar trends are observed for the Birm, and Phi networks, where LR-PBCD_Loop2 consistently achieves the lowest runtime. However, the CS and CR networks are exceptions: LR-PBCD_Loop6 and LR-PBCD_Loop4 achieve the lowest runtimes. These findings indicate that OD classification based on information from the first six generated paths effectively distinguishes hard-to-converge OD pairs from normal OD pairs.

Furthermore, we analyzed the impact of different numbers of special ODs on the computational efficiency of the LR algorithm, as

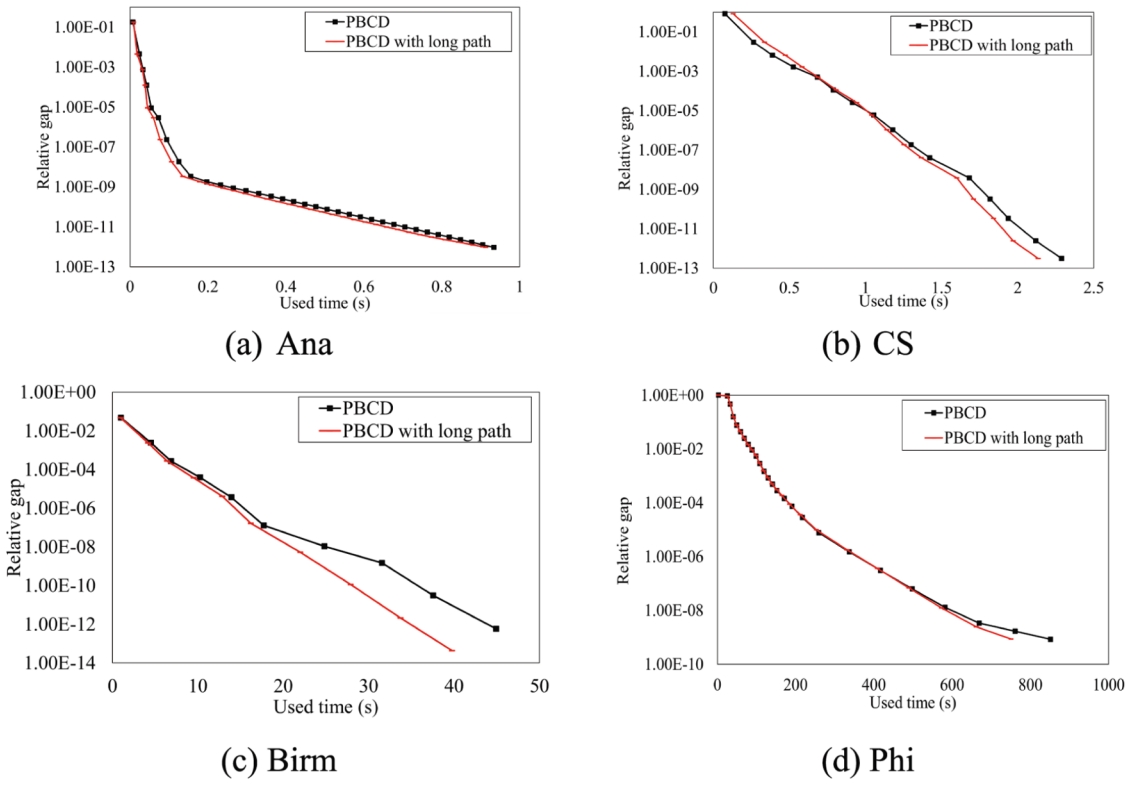


Fig. 9. RG of the OD pairs with the top 20 most calculated counts.

shown in Fig. 11. The sensitivity analysis is conducted across the set {ξ%, (ξ+2)%, (ξ+4)%, (ξ+6)%, (ξ+8)%}. The black curve, which uses only ODs with $RG_{od} > 0$, serves as the baseline. The (ξ+6)% setting yields the shortest runtimes on Ana and CS, (ξ+8)% on Birm and CR, and ξ% on Phi. These results demonstrate that linear regression–guided special ODs selection significantly improves the efficiency of the traffic assignment algorithm without compromising solution accuracy, and that the optimal proportion of special ODs is mildly problem-dependent.

4.2.2. *BB and dropout strategies*

We further explored optimization strategies for the LR-PBCD method by leveraging the best-performing settings identified in Figs. 12 and 13 (the proportion of special ODs and the loop information). Note that the computational times reported in these figures

Table 6a
The R^2 score of LR algorithm for different loops.

Loop	Ana	CS	Birm	Phi	CR
1	0.22	0.02	0.18	0.10	0.07
2	0.28	0.04	0.10	0.17	0.18
4	0.18	0.03	0.06	0.15	0.16
6	0.10	0.05	0.06	0.12	0.12
8	0.02	0.04	0.06	0.09	0.11

Table 6b
Computing time (s) of LR algorithm for different loops.

Loop	Ana	CS	Birm	Phi	CR
1	0.013	0.18	0.85	1.76	3.58
2	0.013	0.18	0.85	1.78	3.62
4	0.013	0.18	0.88	1.81	3.63
6	0.014	0.19	0.95	1.84	3.69
8	0.014	0.20	0.89	1.82	3.64
Average time	0.013	0.19	0.88	1.80	3.63
ProLR	9.97%	22.87%	9.02%	2.51%	2.02%

Table 6c
Computing time (s) of LR algorithm for different numbers of special ODs.

Loop	Ana	CS	Birm	Phi	CR
ξ%	0.013	0.17	0.84	1.66	3.04
(ξ+ 2) %	0.013	0.18	0.85	1.76	3.69
(ξ+ 4) %	0.014	0.18	0.83	1.75	3.72
(ξ+ 6) %	0.015	0.17	0.86	1.77	3.71
(ξ+ 8) %	0.015	0.18	0.85	1.85	3.85
Average time	0.014	0.18	0.85	1.76	3.60
ProLR	10.37%	22.0%	8.67%	2.45%	2.0%

Note that ProLR represents the proportion of LR computation time to the total runtime required by the LR-PBCD algorithm to solve the TAP with RG < 1.0E-12.

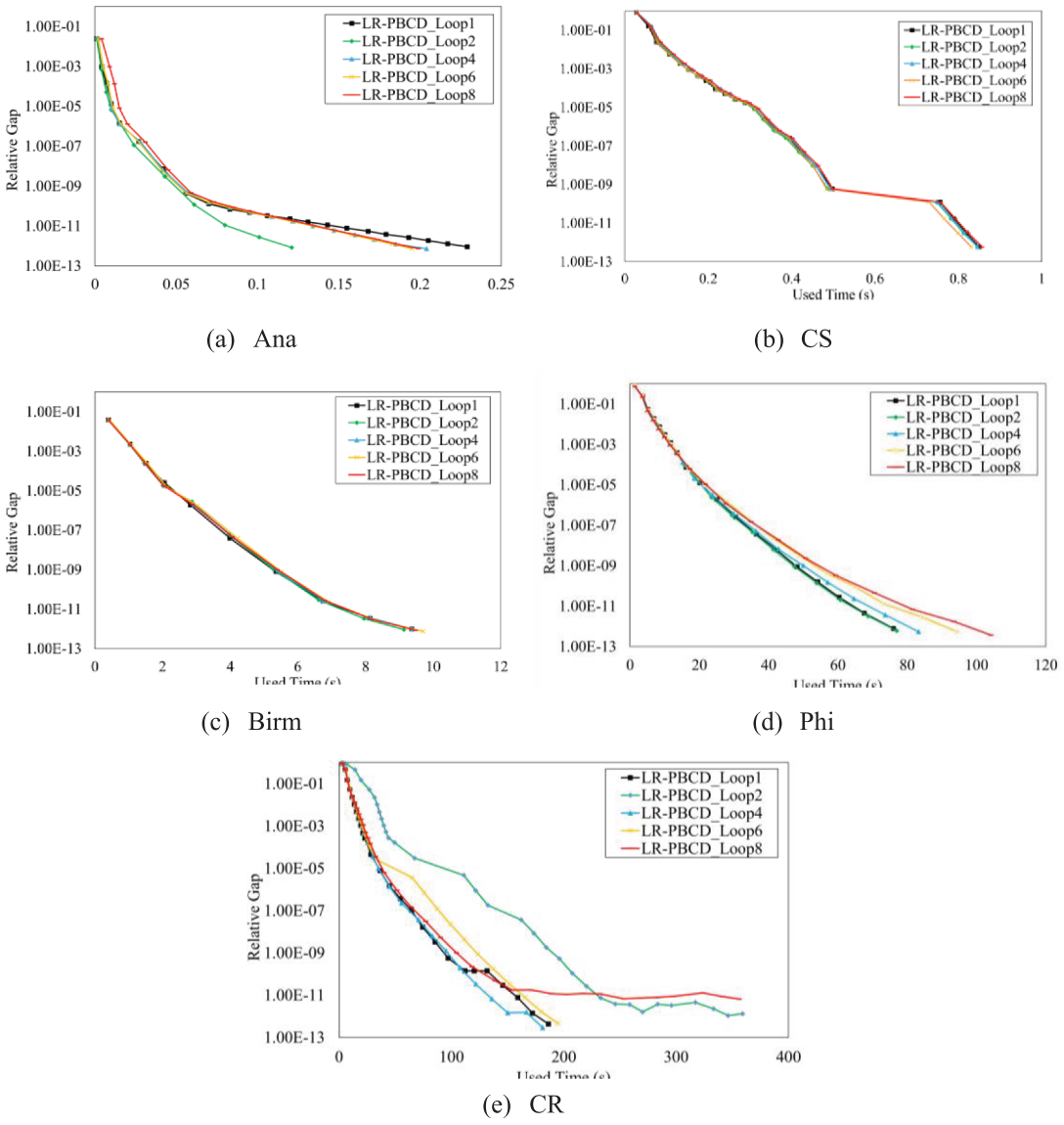
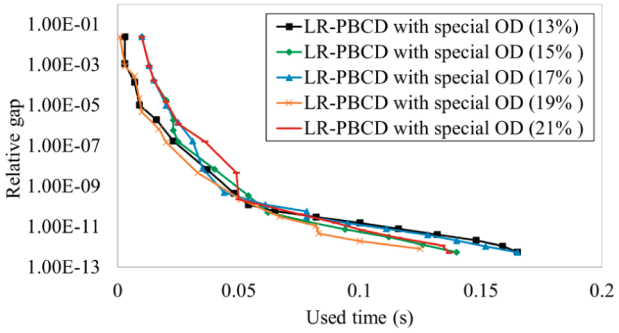
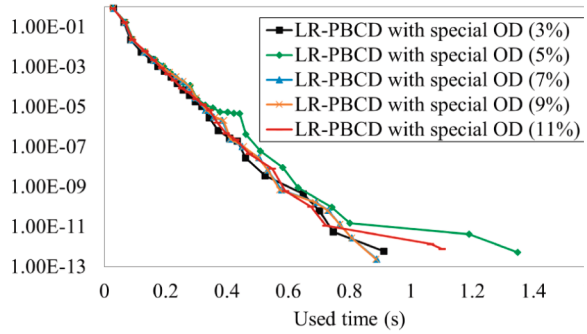


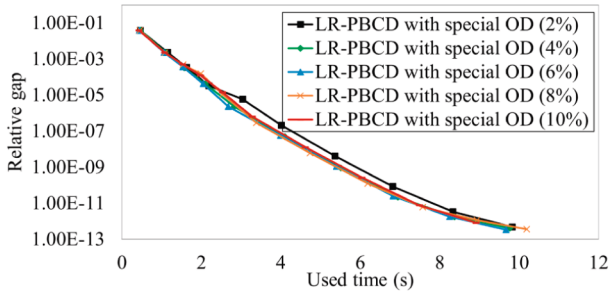
Fig. 10. Sensitivity analysis of loop information.



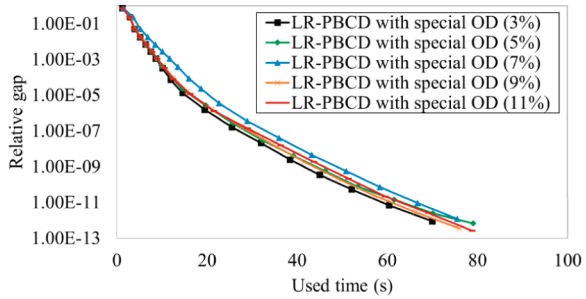
(a) Ana (loop=2)



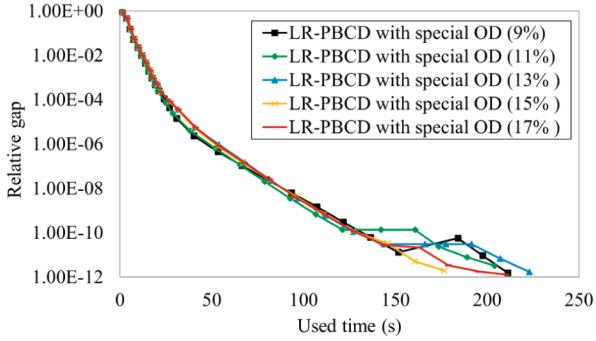
(b) CS (loop=6)



(c) Birm (loop=2)



(d) Phi (loop=2)



(e) CR (loop=4)

Fig. 11. Sensitivity analysis of special ODs.

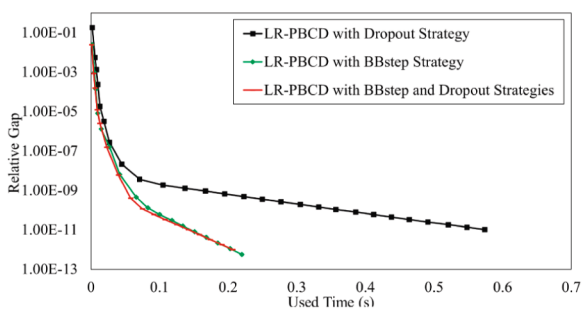
exclude the time required by the LR approach to identify special OD pairs. We also introduced a dropout strategy that randomly skipped a subset of general OD computations: 10% on the Ana network, 50% on the CS network, and 30% on the Birm, Phi, and CR networks.

As shown in Fig. 12, we found that LR-PBCD using both the BB step and dropout strategies performed better across all four test networks, especially on larger networks, for the larger networks, including CS, Birm, Phi, and CR, LR-PBCD with both strategies achieved runtimes that were 25.9%, 12.9%, 9.4% and 31.2% faster than LR-PBCD using only the BB strategy, respectively.

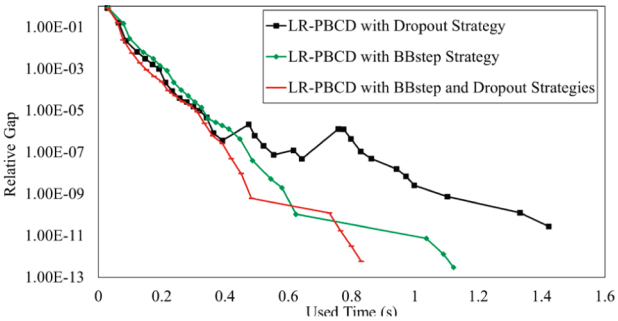
4.3. Comparison of LR-PBCD with existing algorithms

This section will verify the accuracy and efficiency of the proposed algorithm.

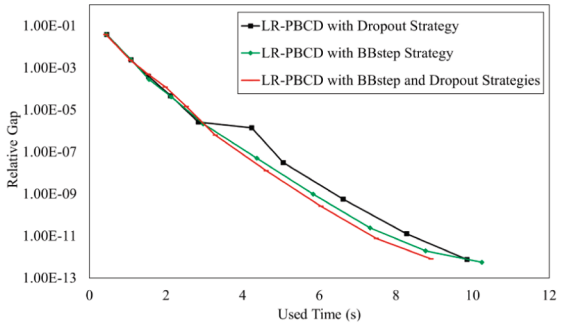
Firstly, we benchmark our approach against the iTAPAS algorithm proposed by Xie and Xie (2016), utilising the publicly available code implementation released by Xie and Nie (2024). We report the Beckmann objective value at convergence and link-level accuracy



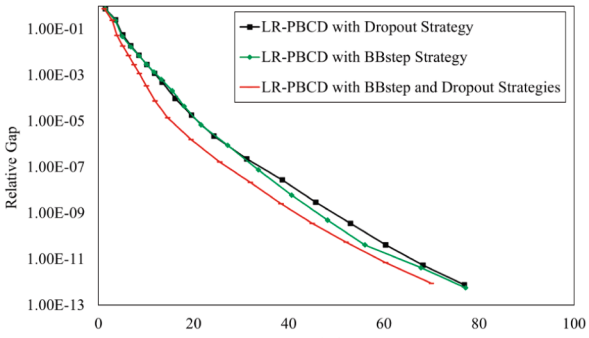
(a) Ana (11% special ODs)



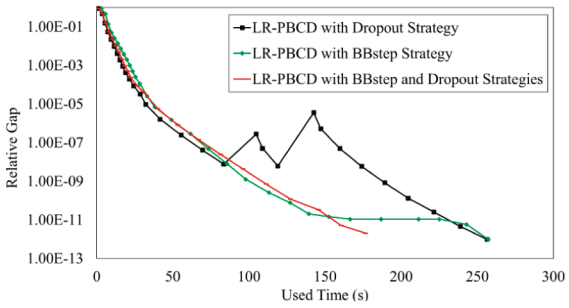
(b) CS (19% special ODs)



(c) Birm (10% special ODs)



(d) Phi (3% special ODs)



(e) CR (15% special ODs)

Fig. 12. LR-PBCD with strategies.

metrics at user equilibrium (UE link flows and UE link costs) for tested networks, thereby quantifying the trade-off between solution quality and runtime across methods. As shown in Table 7, LR-PBCD achieves virtually identical Beckmann objective (obj) values to iTAPAS across all test networks, with differences bounded at approximately 5.2E-5. Moreover, the UE link flows and link costs generated by the iTAPAS and LR-PBCD solutions are virtually identical, and the mean absolute error (MAE) for both is tiny.

Furthermore, Fig. 13 illustrates the comparative performance of the proposed algorithm against existing algorithms, including the path-based iGP, PBCD, iPBCD, Dy-PBCD, and iTAPAS. Notably, iGP and iTAPAS are implemented as serial algorithms, while PBCD, iPBCD, Dy-PBCD, and the proposed LR-PBCD algorithms are parallel algorithms. In this comparison, the runtime of Dy-PBCD to reach RG $<$ 1.0E-12 is used as the benchmark. The computation time reported for LR-PBCD in this section includes the time taken by the LR method to identify special OD pairs.

As shown in Fig. 13(a), the LR algorithm demonstrates superior efficiency compared to existing algorithms on the medium-sized Ana network. Only the iTAPAS and LR-PBCD algorithms exhibit comparable efficiencies when achieving a high-precision RG of 1.0E-12. This performance difference arises because, although LR-PBCD can identify hard-to-converge OD pairs more quickly, it requires additional computational effort and storage for path data, whereas iTAPAS does not necessitate path storage. For the CS network (Fig. 13(b)), Dy-PBCD achieves the shortest computation time of 0.63 s, while LR-PBCD requires 1.06 s. The additional time in LR-PBCD is mainly attributed to the LR classification and BB algorithm components.

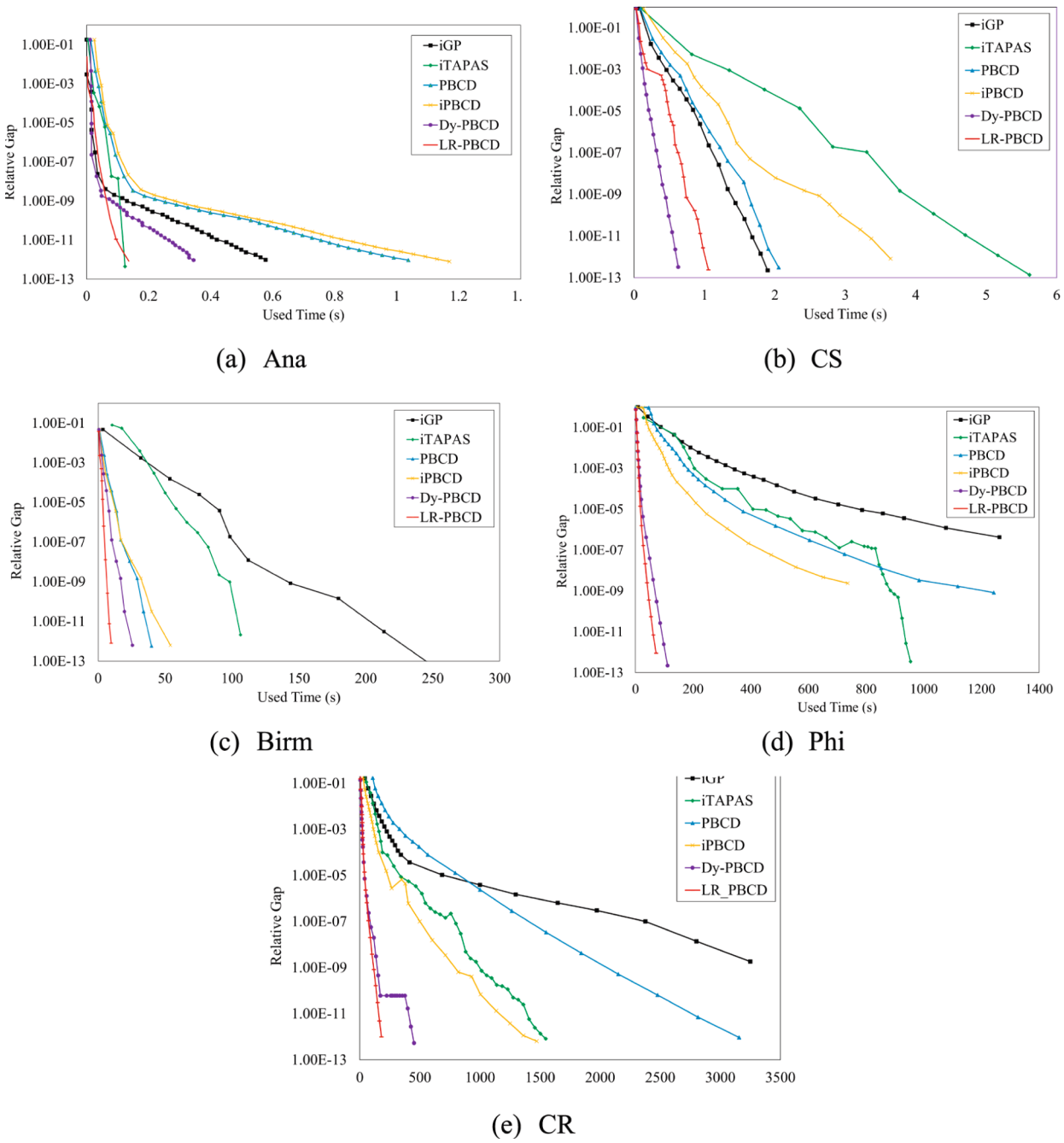


Fig. 13. Computation time of the proposed algorithm compared with existing algorithms.

Furthermore, on larger networks such as Birm, Phi, and CR (Fig. 13 (c)-(e)), the LR-PBCD algorithm consistently outperforms traditional methods, achieving computational efficiencies that are 2.63, 1.55, and 2.51 times greater than Dy-PBCD, respectively. This result demonstrates that as network scale increases, the computational overhead of the BB algorithm decreases proportionally. Consequently, the proposed LR-PBCD algorithm demonstrates exceptional effectiveness, particularly in large-scale scenarios.

5. Conclusion and discussion

This study presents a comprehensive computational framework that leverages linear regression and PBCD to address scalability and efficiency challenges in TAPs. The proposed LR-PBCD method integrates data-driven analysis of OD pair characteristics with advanced optimization techniques. This enables targeted and adaptive treatment of subproblems, especially for OD pairs that converge slowly.

Table 7
The objective value of traffic assignment algorithms with RG ≤ 1.0E-12.

Network	iTAPAS obj	LR-PBCD obj	Obj difference	LF-MAE	LC-MAE
Ana	1286032.171	1286032.171	<5.0E-04	1.40E-04	2.15E-05
CS	16748438.600	16748438.600	<5.0E-04	8.62E-09	1.31E-06
Birm	193514.990	193514.990	<5.0E-04	1.60E-05	2.10E-05
Phi	271499904.642	271499904.642	<5.2E-05	9.68E-01	1.87E-05
CR	25845689.333	25845689.333	<5.0E-04	8.62E-09	1.31E-06

Note that Obj difference denotes the difference in the Beckmann objective value between the user-equilibrium (UE) solutions obtained by iTAPAS and LR-PBCD. LF-MAE represents the mean absolute error (MAE) of link flow differences between the iTAPAS and LR-PBCD UE solutions, while LC-MAE represents the MAE of link cost differences between these two solutions. In other words, LF - MAE $= \frac{1}{|A|}\sum_A \left|x_a^{UE,iTAPAS} - x_a^{UE,LR-PBCD}\right|$ and LC - MAE $= \frac{1}{|A|}\sum_A \left|t_a^{UE,iTAPAS} - t_a^{UE,LR-PBCD}\right|$, where $x_a^{UE,iTAPAS}$ and $t_a^{UE,iTAPAS}$ are the link flow of iTAPAS UE solutions, $x_a^{UE,LR-PBCD}$ and $t_a^{UE,LR-PBCD}$ are the link flow of LR-PBCD UE solutions.

The gradient projection strategy combines fixed-step updates for normal OD pairs and adaptive BB step sizes for challenging cases, and demonstrates robust performance in terms of both solution accuracy and computational speed. Extensive experimental evaluation on five transportation networks confirms that the LR-PBCD method delivers significant efficiency improvements compared with conventional algorithms without compromising precision.

For future research, several technically grounded extensions are particularly promising. On the one hand, the choice of learning model and parallelisation strategy can be refined as richer data become available. In this work, a simple linear regression model is used to distinguish special OD pairs from normal ones. This choice is mainly due to the limited size and dimensionality of available traffic data. Nevertheless, this lightweight statistical model shows that convergence difficulty can be predicted effectively. It also allows computational effort to be allocated adaptively across blocks. With larger and more detailed input data, more powerful machine learning methods could be employed. For example, computational graph formulations can encode the entire path-generation and flow-update process as a differentiable computation graph (Wu et al. 2018). Implicit neural network models can represent equilibrium conditions as differentiable fixed-point or variational operators (Liu et al. 2023a; Liu and Yin 2025). Graph neural networks can process the road network as a graph with node- and link-level features. They can learn OD-level representations that capture path overlap, congestion patterns, and spatial correlations (Liu et al. 2025). Such models are well-suited to capturing nonlinear and high-order interactions between network topology, demand and flow, and could substantially improve the accuracy and robustness of special-OD identification, thereby further enhancing the overall efficiency of LR-PBCD. In addition, the inherently parallel structure of LR-PBCD makes it a natural candidate for GPU-accelerated implementations, in which path generation, block-wise gradient projection, and OD-level updates are mapped to massively parallel kernels, further increasing effectiveness on large networks.

On the other hand, the proposed framework can be extended beyond static UE models to more general assignment models. First, LR-PBCD can be generalised to SUE by replacing the deterministic UE objective with an SUE formulation defined on a fixed path set. The linear regression module can then be trained on OD features together with information on perceived-cost distributions and probabilistic route choice, enabling it to identify special OD pairs under stochastic behaviour (Zhang et al. 2025). Second, the proposed method can also be applied to dynamic traffic assignment. The linear regression module can be trained on OD feature vectors that incorporate time-dependent attributes, such as time-varying travel times, temporal congestion indicators, and spatiotemporal path lengths, to identify special OD pairs that require more intensive updates; these OD pairs can then be treated with more advanced optimisation strategies or allocated additional iterations to improve convergence (Graf et al. 2022). Therefore, these extensions would make the proposed approach a deployable tool for large-scale transportation network analysis and planning across multiple traffic assignment models.

Author Statement

The authors confirm contribution to the paper as follows; study conception and design: K. Zhang, Z. Liu, X. Fu; data preparation: K. Zhang, H. Zhang; analysis and interpretation of results: K. Zhang, Y. Zhang; draft manuscript preparation: K. Zhang, X. Fu. All authors reviewed the results and approved the final version of the manuscript.

CRediT authorship contribution statement

Kai Zhang: Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Zhiyuan Liu:** Writing – original draft, Validation, Project administration, Methodology, Investigation, Conceptualization. **Yuan Zhang:** Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation. **Honggang Zhang:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Formal analysis. **Xiaowen Fu:** Writing – original draft, Project administration, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to

influence the work reported in this paper.

Acknowledgement

Research Institute for Advanced Manufacturing (RIAM, Project # 1-CDLG).

Appendix A:. Process and parameter settings of the BB step size algorithm

We use the BB step size algorithm for special "hard-to-converge" OD pairs. The algorithm is as follows:

Algorithm2: BB step size algorithm	
1	**Input:** Current path flow ($f_p^{od(k)}$), current path cost ($c_p^{od(k)}$), historical path flow ($f_p^{od(k-1)}$), historical path gradient ($c_p^{od(k-1)}$)
2	**If** OD pairs set $\|W\| < 5{,}000$ **do**
3	Return the default step size ($\alpha_{BB}^{od(k)} = 0.1$)
4	**Otherwise, if** the number of paths < 2 **do**
5	Return the default step size ($\alpha_{BB}^{od(k)} = 0.3$)
6	**Otherwise, if** the number of paths ≥ 2 **do**
7	**If** historical data $f_p^{od(k-1)}$ exists **do**
8	Calculate numer $=\sum_{p\in P}\left(f_p^{od(k)} - f_p^{od(k-1)}\right)^2$ Calculate denom $=\sum_{p\in P}\left(f_p^{od(k)} - f_p^{od(k-1)}\right)\left(c_p^{od(k)} - c_p^{od(k-1)}\right)$
9	If $\|$denom$\|$ > 1E-10 and numer > 1E-10 **do**
10	BB step size ($\alpha_{BB}^{od(k)}$) = numer/denomRestrict $\alpha_{BB}^{od(k)}$ to range [0.01, 0.5]
11	Otherwise:Use previous $\alpha_{BB}^{od(k)}$
12	**Otherwise, do**Use default step size ($\alpha_{BB}^{od(k)} = 0.3$)
13	Update historical data $f_p^{od(k)} = f_p^{od(k-1)}$ and $c_p^{od(k)} = c_p^{od(k-1)}$
14	**Output:** Optimized $\alpha_{BB}^{od(k)}$

Note that the path flow vector and gradient vector are maintained for each OD pair. In the inner loop (1,000 iterations), the BB step size is updated once at the first iteration, and the current step size is directly used for the remaining 999 inner loop iterations. In addition, when the number of OD pairs in the network is less than 5,000, the proportion of time consumed by using the BB step size is relatively large. Therefore, a smaller step size is adopted for this type of network.

Appendix B:. Proof of convergence of the proposed method

This subsection gives the proposed method's feasibility conditions, lemma, proposition, and convergence proof.

Feasibility conditions

Consider the block-based traffic assignment model in Eqs. (19)–(20), which satisfies the following basic assumptions:

(a) The objective function $\widetilde{Z}\left(\widetilde{\boldsymbol{f}}\right)$ is continuously differentiable over $\mathbb{R}^n$, and this model satisfies the Kuhn–Tucker conditions for the convex program. Therefore, its gradient is Lipschitz continuous over $\mathbb{R}^n$.

(b) The optimal set for Eq. (52) is nonempty and is represented by $\widetilde{\boldsymbol{f}}^*$, with the corresponding optimal value denoted as $\widetilde{Z}^*$.

The vector $\widetilde{\boldsymbol{f}}$ of the decision variables is partitioned as follows:

$$\widetilde{\boldsymbol{f}} = \left(\widetilde{\boldsymbol{f}}_1, \widetilde{\boldsymbol{f}}_2, \cdots, \widetilde{\boldsymbol{f}}_{i-1}, \widetilde{\boldsymbol{f}}_i, \widetilde{\boldsymbol{f}}_{i+1}, \cdots, \widetilde{\boldsymbol{f}}_s\right) \tag{49}$$

where $\widetilde{\boldsymbol{f}}_i \in \mathbb{R}^{n_i}$ with $n_1, n_2, \cdots, n_s$ are s positive integers that satisfy n. Following the notation from (Nesterov 2012; Cai et al. 2023), we define the matrices $\boldsymbol{U}_i$, for which

$$(\boldsymbol{U}_1, \boldsymbol{U}_2, \cdots, \boldsymbol{U}_s) = \boldsymbol{I}_n \tag{50}$$

Then, $\widetilde{\boldsymbol{f}}_i = \boldsymbol{U}_i\widetilde{\boldsymbol{f}}$ is satisfied for every $\widetilde{\boldsymbol{f}}_i$, $i = 1, 2, \cdots, s$. Moreover, if $\widetilde{\boldsymbol{f}}$ is specified as in Eq. (50), then $\widetilde{\boldsymbol{f}} = \sum_{i=1}^{s}\boldsymbol{U}_i\widetilde{\boldsymbol{f}}_i$. Additionally, we define the vector of partial derivatives for the variables in the vector $\widetilde{\boldsymbol{f}}_i$ as follows:

$$\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}\right) \equiv \boldsymbol{U}_i\nabla\widetilde{Z}\left(\widetilde{\boldsymbol{f}}\right), i = 1, 2, \cdots, s \tag{51}$$

We assume that the gradient of $\widetilde{Z}$ is block coordinate-wise Lipschitz continuous, with the Lipschitz constant for i_{th} block denoted as

L_i, as follows:

$$\left\| \nabla_i \widetilde{Z}\left(\widetilde{\boldsymbol{f}} + \boldsymbol{U}_i \boldsymbol{h}_i\right) - \nabla_i \widetilde{Z}\left(\widetilde{\boldsymbol{f}}\right) \right\| \leq L_i \|\boldsymbol{h}_i\| \tag{52}$$

The constants $L_1, L_2, \cdots, L_s$ are referred to as block-based Lipschitz constants. Additionally, the gradient of $\widetilde{Z}$ is globally Lipschitz continuous, with the corresponding global Lipschitz constant, also denoted as L, specified as follows:

$$\left\| \nabla \widetilde{Z}\left(\widetilde{\boldsymbol{f}}_i\right) - \nabla \widetilde{Z}\left(\widetilde{\boldsymbol{f}}_j\right) \right\| \leq L \left\| \widetilde{\boldsymbol{f}}_i - \widetilde{\boldsymbol{f}}_j \right\| \tag{53}$$

where $\widetilde{\boldsymbol{f}}_i \in \mathbb{R}^n$ and $\widetilde{\boldsymbol{f}}_j \in \mathbb{R}^n$.

The maximum and minimum block-wise Lipschitz constants are represented by

$$L_{\max} = \max_{j=1,2,\cdots,s} L_j \tag{54}$$

$$L_{\min} = \min_{j=1,2,\cdots,s} L_j \tag{55}$$

We use similar notation for the maximal and minimal upper estimates on the block-based Lipschitz constants, as follows:

$$\overline{L}_{\max} = \max_{j=1,2,\cdots,s} \overline{L}_j \tag{56}$$

$$\overline{L}_{\min} = \min_{j=1,2,\cdots,s} \overline{L}_j \tag{57}$$

It is established that the Lipschitz constants $L_1, L_2, \cdots, L_s$, L satisfy the relationships (Nesterov 2012),

$$L \leq \sum_{i=1}^{s} L_i \tag{58}$$

which directly leads to

$$L \leq s L_{\max} \tag{59}$$

then

$$L \leq s \overline{L}_{\max} \tag{60}$$

Furthermore, the ratio is given as follows:

$$L_{rate} = \frac{\overline{L}_{\max}}{\overline{L}_{\min}} \tag{61}$$

where L_{rate} is crucial to analyzing the convergence rate: a large ratio L_{rate} indicates that the convergence of the algorithm is worse.

Given the decomposed structural features of the block-based TAP model described in (Eqs. (19)–(20)), the BCD method operates through an iterative calculation process. Each iteration proceeds as follows: first, we internally optimize each block while maintaining a fixed value for the global variable. Subsequently, we determine the value of the local variable. Finally, global optimization is carried out. The specific recursive updating iterations are outlined as follows.

$$\widetilde{\boldsymbol{f}}_i^{(k)} = \widetilde{\boldsymbol{f}}_{i-1}^{(k)} - \frac{1}{\overline{L}_i} \boldsymbol{U}_i \nabla_i \widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right), i = 1, 2, \cdots, s \tag{62}$$

Lemma and proposition for the proposed method

The effectiveness assessment of the proposed method relies on several Lipschitz constants that have been previously defined (Beck and Tetruashvili 2013; Li et al. 2018).

Our analysis frequently references the well-known descent lemma, which is included here for completeness. The proof of the descent lemma can be found in (Bertsekas 1997). The gradient method converges linearly when the objective function is assumed to be

strongly convex and continuously differentiable with a Lipschitz continuous gradient (Chong and Żak 2013).

Lemma 1. (***(descent)***. *Suppose that $f:\mathbb{R}^d\to\mathbb{R}$ is a continuously differentiable function with a gradient ∇f that is Lipschitz continuous with a constant M.*) *Then,*

$$f(y)\le f(x)+\langle\nabla f(x),y-x\rangle+\frac{M}{2}\|x-y\|^2 \tag{63}$$

Next, we derive the following "block" version of lemma 1.

Lemma 2. (***(block descent)***. *Consider $\widetilde{Z}$ as a continuously differentiable function over $\mathbb{R}^n$ that satisfies Eq. (52). Let $\widetilde{\boldsymbol{f}}_{i,\boldsymbol{u}},\widetilde{\boldsymbol{f}}_{i,\boldsymbol{v}}$ be two vectors that differ solely in the i_{th} block, $i\in(1,2,\ldots,s)$. Thus, there exists a vector $\widetilde{\boldsymbol{f}}_{i,\boldsymbol{h}}\in\mathbb{R}^{n_i}$ such that $\widetilde{\boldsymbol{f}}_{i,\boldsymbol{u}}-\widetilde{\boldsymbol{f}}_{i,\boldsymbol{v}}=\boldsymbol{U}_i\widetilde{\boldsymbol{f}}_{i,\boldsymbol{h}}$. Then*)

$$\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i,\boldsymbol{v}}\right)\le\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i,\boldsymbol{u}}\right)+\left\langle\nabla\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i,\boldsymbol{u}}\right),\widetilde{\boldsymbol{f}}_{i,\boldsymbol{v}}-\widetilde{\boldsymbol{f}}_{i,\boldsymbol{u}}\right\rangle+\frac{\overline{L}_i}{2}\left\|\widetilde{\boldsymbol{f}}_{i,\boldsymbol{v}}-\widetilde{\boldsymbol{f}}_{i,\boldsymbol{u}}\right\|^2 \tag{64}$$

Given the underlying assumptions about the objective function, we demonstrate that a sublinear convergence rate can be established for the sequence of function values.

Proposition 1. (***(Convergence of the coordinate block descent method):*** *$\widetilde{Z}$ is continuously differentiable over $\mathbb{R}^s$, and every $\widetilde{\boldsymbol{f}}_i\in\widetilde{\boldsymbol{f}},i=1,2,\cdots,s$ satisfies*)

$$\widetilde{\boldsymbol{f}}_i:=\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_1^{(k)},\widetilde{\boldsymbol{f}}_2^{(k)},\ldots,\widetilde{\boldsymbol{f}}_{i-1}^{(k)},\widetilde{\boldsymbol{f}}_i,\widetilde{\boldsymbol{f}}_{i+1}^{(k-1)},\ldots,\widetilde{\boldsymbol{f}}_s^{(k-1)}\right) \tag{65}$$

There is a unique minimum point $\widetilde{\boldsymbol{f}}_i^{\min}$ on $\widetilde{\boldsymbol{f}}_i$, and the objective function $\widetilde{Z}$ is monotonically non-increasing over the interval from $\widetilde{\boldsymbol{f}}_i$ to $\widetilde{\boldsymbol{f}}_i^{\min}$. Let $\left\{\widetilde{\boldsymbol{f}}^{(k)}\right\}$ be the sequence generated by the iterative calculation of the BCD method, then every limit point of $\left\{\widetilde{\boldsymbol{f}}^{(k)}\right\}$ is a coordinate-wise minimum point.

Sublinear rate of convergence of the proposed method

Lemma 3. (*Consider the sequence $\left\{\widetilde{\boldsymbol{f}}^{(k)}\right\}$ generated by the proposed method. Thus, for each iteration $k=1,2,\cdots$, the sublinear rate of convergence is as follows:*)

$$\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)-\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k+1)}\right)\ge\frac{\overline{L}_{\min}^2}{4\overline{L}_{\max}\left(sL^2+\overline{L}_{\min}^2\right)}\sum_{i=1}^{s}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)\right\|^2 \tag{66}$$

where $\overline{L}_{\max}$ and $\overline{L}_{\min}$ are defined in Eqs. (56) and (57), respectively. Detailed proof is provided in **Appendix C.**

Appendix C:. Proof of Lemma 3

Proof. Using the block descent lemma (Lemma 3) and considering that $L_i\le\overline{L}_i$, we have that for all $i=1,\ldots,s$,

$$\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_i^{(k)}\right)\le\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_i^{(k-1)}\right)+\left\langle\nabla\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_i^{(k-1)}\right),\widetilde{\boldsymbol{f}}_i^{(k)}-\widetilde{\boldsymbol{f}}_i^{(k-1)}\right\rangle+\frac{\overline{L}_i}{2}\left\|\widetilde{\boldsymbol{f}}_i^{(k)}-\widetilde{\boldsymbol{f}}_i^{(k-1)}\right\|^2 \tag{67}$$

Inserting the recursion relation from Eq. (62) into Eq. (67), we get

$$\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)-\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_i^{(k)}\right)\ge\frac{1}{2\overline{L}_i}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2 \tag{68}$$

where $\frac{1}{\overline{L}_i}$ is the step size.

According to the $\overline{L}_i\le\overline{L}_{\max}$, we sum all of the inequalities and obtain

$$\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)-\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k+1)}\right)\ge\frac{1}{2}\sum_{i=1}^{s}\frac{1}{\overline{L}_i}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2\ge\frac{1}{2\max\limits_{j=1,\ldots,s}\{\overline{L}_j\}}\sum_{i=1}^{s}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2 \tag{69}$$

Examining Eq. (62) and noting that $\widetilde{\boldsymbol{f}}_1^{(k)}$ equals $\widetilde{\boldsymbol{f}}^{(k)}$ results in the following for each block $i=1,\ldots,s$,

$$\widetilde{\boldsymbol{f}}^{(k)}=\widetilde{\boldsymbol{f}}_i^{(k)}+\sum_{j=1}^{i}\frac{1}{\overline{L}_j}\boldsymbol{U}_j\nabla\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_j^{(k-1)}\right) \tag{70}$$

By using the Lipschitz continuity, we have

$$\begin{aligned}\left\|\nabla\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)-\nabla\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_i^{(k)}\right)\right\|^2 &\le L^2\left\|\widetilde{\boldsymbol{f}}^{(k)}-\widetilde{\boldsymbol{f}}_i^{(k)}\right\|^2\\ &= L^2\left\|\sum_{j=1}^{i}\frac{1}{\overline{L}_j}\boldsymbol{U}_j\nabla_j\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_j^{(k-1)}\right)\right\|^2\\ &\le \frac{L^2}{2\min\limits_{j=1,\ldots,s}\left\{\overline{L}_j^2\right\}}\sum_{j=1}^{i}\left\|\nabla_j\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_j^{(k-1)}\right)\right\|^2\end{aligned} \tag{71}$$

Then, we use the norm inequality

$$\begin{aligned}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)\right\|^2 &\le \left(\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)-\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|+\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|\right)^2\\ &\le 2\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)-\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2+2\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2\\ &\le 2\frac{L^2}{\min\limits_{j=1,\ldots,s}\left\{\overline{L}_j^2\right\}}\sum_{j=1}^{i-1}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2+2\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2\end{aligned} \tag{72}$$

In summary, over all blocks $i = 1, \ldots, s$, we obtain

$$\begin{aligned}\sum_{i=1}^{s}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)\right\|^2 &\le \sum_{i=1}^{s}\left(2\frac{L^2}{\min\limits_{j=1,\ldots,s}\left\{\overline{L}_j^2\right\}}\sum_{j=1}^{i-1}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2+2\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2\right)\\ &\le 2\left((s-i)\frac{L^2}{\min\limits_{j=1,\ldots,s}\left\{\overline{L}_j^2\right\}}+1\right)\sum_{i=1}^{s}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2\\ &\le 2\left(s\frac{L^2}{\min\limits_{j=1,\ldots,s}\left\{\overline{L}_j^2\right\}}+1\right)\sum_{i=1}^{s}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2\end{aligned} \tag{73}$$

Then,

$$\frac{\sum_{i=1}^{s}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)\right\|^2}{2\left(s\frac{L^2}{\min\limits_{j=1,\ldots,s}\left\{\overline{L}_j^2\right\}}+1\right)}\le\sum_{i=1}^{s}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2 \tag{74}$$

Hence, by combining Eq. and Eq., Lemma 3 can be obtained as follows.

$$\begin{aligned}\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)-\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k+1)}\right) &\ge \frac{1}{2\max\limits_{j=1,\ldots,s}\left\{\overline{L}_j\right\}}\sum_{i=1}^{s}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}_{i-1}^{(k)}\right)\right\|^2\\ &\ge \frac{1}{2\max\limits_{j=1,\ldots,s}\left\{\overline{L}_j\right\}}\frac{\sum_{i=1}^{s}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)\right\|^2}{2\left(s\frac{L^2}{\min\limits_{j=1,\ldots,s}\left\{\overline{L}_j^2\right\}}+1\right)}\\ &\ge \frac{\overline{L}_{\min}^2}{4\overline{L}_{\max}\left(sL^2+\overline{L}_{\min}^2\right)}\sum_{i=1}^{s}\left\|\nabla_i\widetilde{Z}\left(\widetilde{\boldsymbol{f}}^{(k)}\right)\right\|^2\end{aligned} \tag{75}$$

This completes the proof. □

Appendix D:. OD pair feature extraction in the data preprocessing stage

The length of OD pairs is the main factor affecting convergence, while OD demand (when relatively small) is less important. It can

be observed in Fig. a1 that a reasonable level of travel demand between OD pairs has a minimal impact on UE TAP convergence. However, when the travel demand between OD pairs is extremely high, it significantly affects the UE traffic flow distribution, thus influencing the convergence speed of a UE TAP.

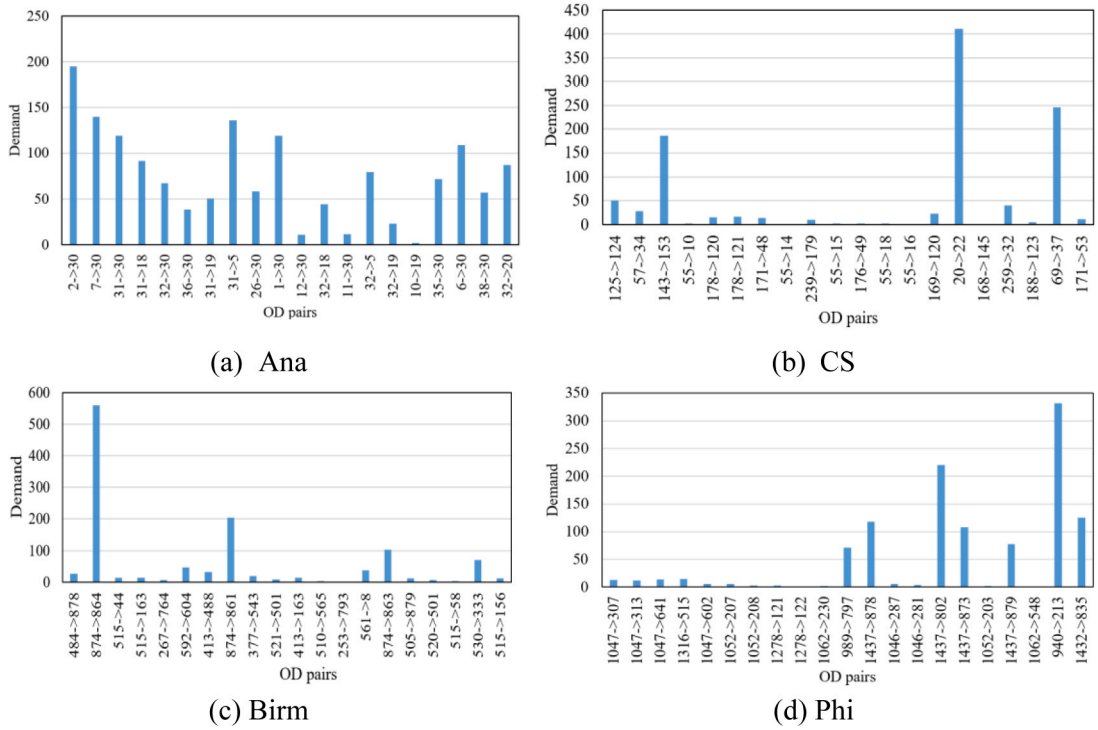


Fig. c1. Demand of the OD pairs with the top 20 most calculated counts

Figs. c2–c4 show that OD subproblems with longer paths require more iterations. Thus, it can be inferred that OD pair length is the primary factor influencing convergence, whereas OD pair demand (when relatively small) is less important.

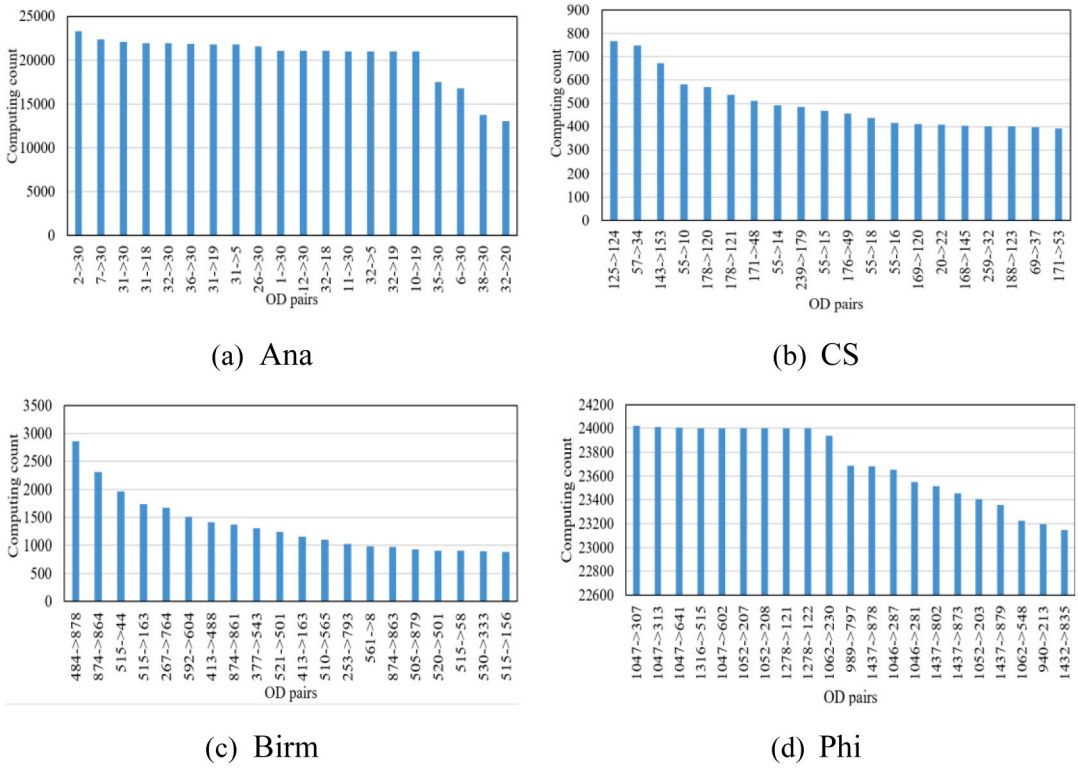


Fig. c2. OD pairs with the top 20 most calculated counts

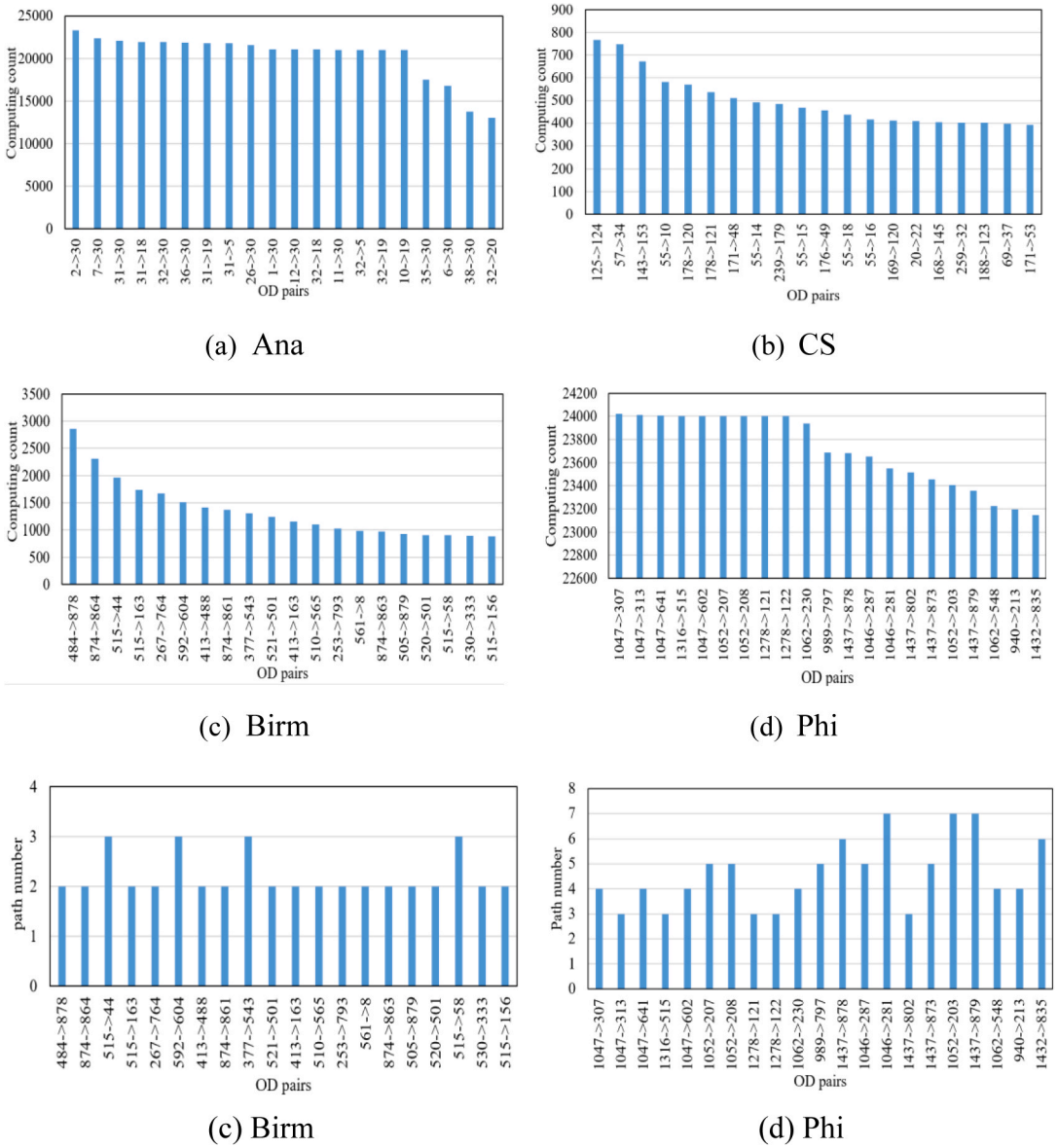


Fig. c3. Number of paths of OD pairs with the top 20 most calculated counts

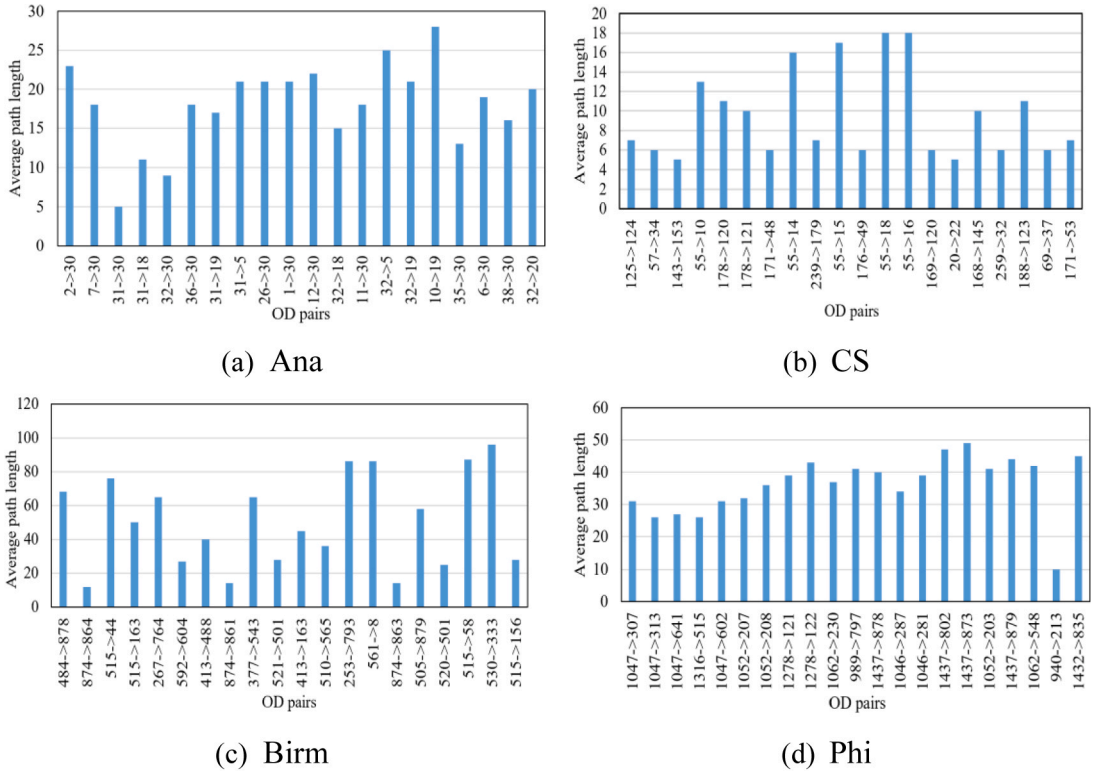


Fig. c4. Average path length of OD pairs with the top 20 most calculated counts

Data availability

The original links for downloading the datasets are provided in the manuscript.

References

Allen S. E. Parallel implementations of the Frank-Wolfe algorithms for the traffic assignment problem. 2023. Doctoral Dissertation, Old Dominion University, US.

Arezoumandi, M., 2011. Estimation of travel time reliability for freeways using mean and standard deviation of travel time. J. Transp. Syst. Eng. Inf. Technol. 11, 74–84.

Babazadeh, A., Javani, B., Gentile, G., Florian, M., 2020. Reduced gradient algorithm for user equilibrium traffic assignment problem. Transportmetrica a: Transport Science 16, 1111–1135.

Bar-Gera, H., 2002. Origin-based algorithm for the traffic assignment problem. Transp. Sci. 36, 398–417.

Bar-Gera, H., 2010. Traffic assignment by paired alternative segments. Transp. Res. B Methodol. 44, 1022–1046.

Beck, A., Tetruashvili, L., 2013. On the convergence of block coordinate descent type methods. SIAM J. Optim. 23, 2037–2060.

Bertsekas, D.P., 1997. Nonlinear programming. J. Oper. Res. Soc. 48, 334.

Bottou, L., 2010. Large-scale machine learning with stochastic gradient descent. *19th International Conference on Computational Statistics*.

Boyce, D., Ralevic-Dekic, B., Bar-Gera, H., 2004. Convergence of traffic assignments: how much is enough? J. Transp. Eng. 130, 49–55.

Buchhold, V., Sanders, P., Wagner, D., 2019. Real-time traffic assignment using engineered customizable contraction hierarchies. The ACM Journal of Experimental Algorithmics 24, 1–28.

Cai, X., Song, C., Wright, S., Diakonikolas, J., 2023. Cyclic block coordinate descent with variance reduction for composite nonconvex optimization. International Conference on Machine Learning 3469–3494.

Chauhan, V.K., Dahiya, K., Sharma, A., 2017. Mini-batch block-coordinate based stochastic average adjusted gradient methods to solve big data problems. *9th Asian Conference on Machine Learning*.

Chen, A., Jayakrishnan, R., 1998. A Path-Based Gradient Projection Algorithm: Effects of Equilibration with a Restricted Path Set under Two Flow Update Policies. Research Board, Washington, USA, January.

Chen, H., Yang, K., Rizzo, S.G., Vantini, G., Taylor, P., Ma, X., Chawla, S., 2020a. QarSUMO: a parallel, congestion-optimized traffic simulator. *28th International Conference on Advances in Geographic Information Systems*.

Chen, R., Leclercq, L., Ameli, M., 2021. Unravelling system optimums by trajectory data analysis and machine learning. Transp. Res. Part C Emerging Technol. 130, 103318.

Chen, X., Liu, Z., Zhang, K., Wang, Z., 2020b. A parallel computing approach to solve traffic assignment using path-based gradient projection algorithm. Transp. Res. Part C Emerging Technol. 120, 102809.

Chong, E.K., Żak, S.H., 2013. An introduction to optimization. John Wiley & Sons.

Dial, R.B., 2006. A path-based user-equilibrium traffic assignment algorithm that obviates path storage and enumeration. Transp. Res. B Methodol. 40, 917–936.

Dhal, P., Azad, C., 2022. A comprehensive survey on feature selection in the various fields of machine learning. Applied. Intelligence:1–39.

Du, M., Tan, H., Chen, A., 2021. A faster path-based algorithm with Barzilai-Borwein step size for solving stochastic traffic equilibrium models. Eur. J. Oper. Res. 290, 982–999.
Feijoo, B., Meyer, R.R., 1988. Piecewise-linear approximation methods for nonseparable convex optimization. Manag. Sci. 34, 411–419.
Florian, M. and D. Hearn. 1995. Chapter 6: Network equilibrium models and algorithms. Handbooks in Operations Research and Management Science. 485-550.
Florian, M., Guálat, J., Spiess, H., 1987. An efficient implementation of the PARTAN variant of the linear approximation method for the network equilibrium problem. Networks 17, 319–339.
Florian, M., Constantin, I., Florian, D., 2009. A new look at projected gradient method for equilibrium assignment. Transp. Res. Rec. 2090, 10–16.
Freedman, D., Pisani, R., Purves, R., 2020. Statistics: fourth international, student edition. Norton & Company, W. W.
Gendreau, M., Guertin, F., Potvin, J.Y., Taillard, E., 1999. Parallel tabu search for real-time vehicle routing and dispatching. Transp. Sci. 33, 381–390.
Gentile, G., 2014. Local user cost equilibrium: a bush-based algorithm for traffic assignment. Transportmetrica a: Transport Science 10, 15–54.
Graf, L., Harks, T., Kollias, K., Markl, M., 2022. Machine-learned prediction equilibrium for dynamic traffic assignment. AAAI Conference on Artificial Intelligence 5059–5067.
Guessous, Y., Aron, M., Bhouri, N., Cohen, S., 2014. Estimating travel time distribution under different traffic conditions. Transp. Res. Procedia 3, 339–348.
Huang, L., Zhou, Y., Zhu, F., Liu, L., Shao, L., 2019. Iterative normalization: beyond standardization towards efficient whitening. IEEE/CVF Conference on Computer Vision and Pattern Recognition 4874–4883.
Hu, X., Xie, C., 2025. Use of graph attention networks for traffic assignment in a large number of network scenarios. Transp. Res. Part C Emerging Technol. 171, 104997.
Ioffe, S. and C. Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift., Lille, France, 448-456.
Jafari, E., Pandey, V., Boyles, S.D., 2017. A decomposition approach to the static traffic assignment problem. Transp. Res. B Methodol. 105, 270–296.
Jayakrishnan, R., Tsai, W.T., Prashker, J.N., Rajadhyaksha, S., 1994. A faster path-based algorithm for traffic assignment. University of California Transportation Center.
Larsson, T., Patriksson, M., 1992. Simplicial decomposition with disaggregated representation for the traffic assignment problem. Transp. Sci. 26, 4–17.
LeBlanc, L.J., Morlok, E.K., Pierskalla, W.P., 1975. An efficient approach to solving the road network equilibrium traffic assignment problem. Transp. Res. 9, 309–318.
Li, X., Zhao, T., Arora, R., Liu, H., Hong, M., 2018. On faster convergence of cyclic block coordinate descent-type methods for strongly convex minimization. J. Mach. Learn. Res. 18, 1–24.
Liu, H.X., He, X., He, B., 2009. Method of successive weighted averages (MSWA) and self-regulated averaging schemes for solving stochastic user equilibrium problem. Netw. Spat. Econ. 9, 485–503.
Liu, X., Zhang, Y., Zhang, K., Cheng, Q., Xing, J., Liu, Z., 2025. A scalable learning approach for user equilibrium traffic assignment problem using graph convolutional networks. Electron. Res. Arch. 33, 3246–3270.
Liu, Z., Meng, Q., 2013. Distributed computing approaches for large-scale probit-based stochastic user equilibrium problems. J. Adv. Transp. 47, 553–571.
Liu, Z., Wang, Z., Cheng, Q., Yin, R., Wang, M., 2021. Estimation of urban network capacity with second-best constraints for multimodal transport systems. Transp. Res. B Methodol. 152, 276–294.
Liu, Z., Chen, X., Hu, J., Wang, S., Zhang, K., Zhang, H., 2023a. An alternating direction method of multipliers for solving user equilibrium problem. Eur. J. Oper. Res. 3, 1072–1084.
Liu, Z., Xie, S., Zhang, H., Zhou, D., Yang, Y., 2024. A parallel computing framework for large-scale microscopic traffic simulation based on spectral partitioning. Transportation Research Part e: Logistics and Transportation Review 181, 103368.
Liu, Z., Yin, Y., Bai, F., Grimm, D.K., 2023b. End-to-end learning of user equilibrium with implicit neural networks. Transp. Res. Part C Emerging Technol. 150, 104085.
Liu, Z., Yin, Y., 2025. End-to-end learning of user equilibrium: expressivity, generalization, and optimization. Transp. Sci. https://doi.org/10.1287/trsc.2023.0489.
Lopes, R., Santos, S.A., Silva, P.J., 2019. Accelerating block coordinate descent methods with identification strategies. Comput. Optim. Appl. 72, 609–640.
Lu, L., Xu, Y., Antoniou, C., Ben-Akiva, M., 2015. An enhanced SPSA algorithm for the calibration of dynamic traffic assignment models. Transp. Res. Part C Emerging Technol. 51, 149–166.
Mitradjieva, M., Lindberg, P.O., 2013. The stiff is Moving-Conjugate direction Frank-Wolfe methods with applications to traffic assignment. Transp. Sci. 47, 280–293.
Mutny, M., Derezinski, M., Krause, A., 2020. Convergence analysis of block coordinate algorithms with determinantal sampling. International Conference on Artificial Intelligence and Statistics 3110–3120.
Necoara, I., Nesterov, Y., Glineur, F., 2017. Random block coordinate descent methods for linearly constrained optimization over networks. J. Optim. Theory Appl. 173, 227–254.
Nesterov, Y., 2012. Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM J. Optim. 22, 341–362.
Nie, Y. M. 2016. VNET. Assignment simulation software. https://sites.northwestern.edu/marconie/.
Nutini, J., Laradji, I., Schmidt, M., 2022. Let's make block coordinate descent converge faster: faster greedy rules, message-passing, active-set complexity, and superlinear convergence. J. Mach. Learn. Res. 23, 1–74.
Perederieieva, O., Ehrgott, M., Raith, A., Wang, J.Y.T., 2015. A framework for and empirical study of algorithms for traffic assignment. Comput. Oper. Res. 54, 90–107.
Potuzak, T., Kolovsky, F., 2022. Parallelization of the B static traffic assignment algorithm. Ain Shams Eng. J. 13, 101576.
Raadsen, M.P.H., Bliemer, M.C.J., Bell, M.G.H., 2020. Aggregation, disaggregation and decomposition methods in traffic assignment: historical perspectives and new trends. Transp. Res. B Methodol. 139, 199–223.
Richtárik, P., Takáč, M., 2014. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. Math. Program. 144, 1–38.
Roocroft, A., Ramli, M.A., Punzo, G., 2023. Data-driven traffic assignment through density-based road-specific congestion function estimation. IEEE. Access:192–205.
Rose, G., Daskin, M.S., Koppelman, F.S., 1988. An examination of convergence error in equilibrium traffic assignment models. Transp. Res. B Methodol. 22, 261–274.
Salimans, T., Kingma, D.P., 2016. Weight normalization: a simple reparameterization to accelerate training of deep neural networks. In: *Advances in Neural Information Processing Systems*, p. 29.
Sheffi, Y., 1984. Urban transportation networks. Prentice-Hall, Englewood Cliffs, NJ.
Tan, H., Xu, X., Chen, A., 2024. On endogenously distinguishing inactive paths in stochastic user equilibrium: a convex programming approach with a truncated path choice model. Transp. Res. B Methodol. 183, 102940.
Tran-Dinh, Q., Luo, Y., 2025. Randomized block-coordinate optimistic gradient algorithms for root-finding problems. Math. Oper. Res.
Wang, X., L. Yan and Q. Zhang. 2021b. Research on the application of gradient descent algorithm in machine learning. *International Conference on Computer Network, Electronic and Automation, Xi'an, China, September 24-26*, 11-15.
Wang, Z., Zhang, K., Chen, X., Wang, M., Liu, R., Liu, Z., 2021b. An improved parallel block coordinate descent method for the distributed computing of traffic assignment problem. Transportmetrica a: Transport. Science:1–32.
Wardrop, J.G., 1952. Some theoretical aspects of road traffic research. Proc. Inst. Civ. Eng. 3, 325–362.
Wright, S.J., 2015. Coordinate Descent Algorithms. Mathematical Programming 151, 3–34.
Wu, X., Guo, J., Xian, K., Zhou, X., 2018. Hierarchical travel demand estimation using multiple data sources: a forward and backward propagation algorithmic framework on a layered computational graph. Transp. Res. Part C Emerging Technol. 96, 321–346.
Xie, J. and Y. Nie. 2024. The open-source code of iTAPAS. https://github.com/junxie016/Open-TNM.
Xie, J., Nie, Y., Liu, X., 2018. A greedy path-based algorithm for traffic assignment. Transportation Research Record 2672, 36–44.
Xie, J., Xie, C., 2016. New insights and improvements of using paired alternative segments for traffic assignment. Transportation Research Part b: Methodological 93, 406–424.
Xu, Y., 2018. Hybrid Jacobian and Gauss-Seidel proximal block coordinate update methods for linearly constrained convex programming. SIAM Journal on Optimization 28, 646–670.

Yahia, C.N., Pandey, V., Boyles, S.D., 2018. Network partitioning algorithms for solving the traffic assignment problem using a decomposition approach. *Transportation Research Record*. Journal of the Transportation Research Board.
Zhang, H., Liu, Z., Wang, J., Wu, Y., 2023a. A novel flow update policy in solving traffic assignment problems: Successive over relaxation iteration method. Transportation Research Part e: Logistics and Transportation Review 174, 103111.
Zhang, K., Zhang, H., Cheng, Q., Chen, X., Wang, Z., Liu, Z., 2023b. A customized two-stage parallel computing algorithm for solving the combined modal split and traffic assignment problem. Computers & Operations Research 154, 106193.
Zhang, K., Zhang, H., Dong, Y., Wu, Y., Chen, X., 2023c. An ADMM-based parallel algorithm for solving traffic assignment problem with elastic demand. Communications in Transportation Research 3, 100108.
Zhang, K., Liu, Z., Zhang, H., Zhang, Y., Tang, Y.M., Fu, X., 2025. A graph vertex-coloring-based parallel block coordinate descent method for solving the traffic assignment problem. Transportation Research Part c: Emerging Technologies 183 (2025), 105439.
Zhang, Y., Li, L., Zhang, W., Cheng, Q., 2022. GATC and DeepCut: deep spatiotemporal feature extraction and clustering for large-scale transportation network partition. Physica a: Statistical Mechanics and Its Applications 606, 128110.
Zhao, T., Yu, M., Wang, Y., Arora, R., Liu, H., 2014. Accelerated mini-batch randomized block coordinate descent method. Advances in Neural Information Processing Systems 27.