*Journal of Industrial and Management Optimization*

https://www.aimspress.com/journal/jimo

JIMO, 22(6): 2726–2754.
DOI:10.3934/jimo.2026100
Received: 11 November 2025
Revised: 01 May 2026
Accepted: 11 May 2026
Published: 20 May 2026

*Research article*

# Pickup scheduling algorithms with spatio-temporal coordination for urban educational facilities

**Yuan-Ko Huang***

Department of Computer Science and Information Engineering, National Kaohsiung University of Science and Technology, Kaohsiung City, Taiwan

* **Correspondence:** Email: huangyk@nkust.edu.tw.

**Abstract:** In urban areas where multiple urban educational facilities (UEFs) are located in close proximity and dismiss students simultaneously, uncoordinated vehicle pickups often lead to localized congestion and prolonged idling. This study addresses this problem by proposing a four-phase pickup scheduling algorithm that enables spatio-temporal coordination across neighboring UEFs. The algorithm consists of four phases: (1) identifying related UEFs based on spatial proximity, (2) triggering batch scheduling when incoming requests reach a calculated threshold, (3) prioritizing requests using different strategies, and (4) assigning pickup intervals under travel-time feasibility and street capacity constraints. The approach was evaluated under various spatial and operational settings and compared with a non-scheduling baseline. Experimental results demonstrated that the proposed scheduling method reduces the variability of parent waiting times, the maximum parent waiting time, and overall makespan. These results highlight the effectiveness of coordinated scheduling across neighboring UEFs in improving pickup efficiency and alleviating localized congestion in urban school neighborhoods, thereby contributing to sustainable cities and communities.

**Keywords:** urban educational facilities; pickup scheduling; travel-time feasibility; capacity constraints; sustainable cities and communities
**Mathematics Subject Classification:** 90B35, 90C59

## 1. Introduction

In many East Asian metropolitan regions facing declining birth rates, parents highly value their children's education and seek to enhance their competitiveness through various means. One common approach is enrolling children in urban educational facilities (UEFs), such as after-school programs, talent centers, and private tutoring institutions, during non-school hours. Parents' motivations include providing academic support, fostering talent development, preparing for important exams, encouraging

social interaction, and ensuring after-school supervision. UEFs thus offer structured curricula, professional instruction, diverse enrichment activities, and safe environments, meeting families' multifaceted needs. Consequently, urban areas around schools are densely populated with various types of UEFs catering to these demands. Here, the school serves as the reference point defining the surrounding neighborhood, while UEFs denote the nearby educational facilities distributed around it. As shown in Figure 1(a), numerous different types of UEFs are densely located around schools, highlighted with red elliptical outlines; Figure 1(b) further shows multiple UEFs (marked similarly) operating along the same street.


(a) UEFs surrounding a school (b) Multiple UEFs located along the same street

**Figure 1.** Spatial distribution of urban educational facilities (UEFs).

Since most UEF courses are scheduled after regular school hours, students often finish classes in the evening. For safety reasons, many parents prefer to personally pick up their children using private vehicles. During pickup times, particularly when multiple UEFs dismiss students simultaneously, parents often temporarily park their vehicles in front of or near the facilities. However, UEFs are frequently located along busy roads, where parking spaces are scarce. As a result, double parking or illegal stopping is common, contributing to traffic congestion, inconvenience for other road users and residents, and increased risk of traffic accidents. Moreover, prolonged idling of vehicles leads to higher emissions of air pollutants, posing additional health and environmental concerns.

While several real-world solutions, such as mobile apps like PikMyKid [1], CurbSmart [2], PickMeUp [3], and others, have been developed to improve pickup safety and efficiency, they primarily support communication between individual parents and a single UEF. These systems allow parents to notify the facility of their expected arrival time, but they operate independently—each parent acts without knowledge of others' plans, and each UEF handles pickups without coordination with nearby facilities. As a result, when many parents arrive at the same UEF around the same time, or when several neighboring UEFs dismiss students within similar timeframes, traffic congestion near the facilities becomes unavoidable. This reveals a key challenge in densely populated urban areas: the lack of coordination across multiple, spatially clustered UEFs can significantly amplify pickup-related traffic and safety issues.

From the parents' perspective, arriving at the pickup location precisely when their children are dismissed, without waiting along nearby streets, would significantly improve convenience and reduce

unnecessary delays. From the UEFs' perspective, if parental arrivals could be distributed more evenly over time, it would help alleviate localized congestion and improve traffic safety around school neighborhoods. However, in urban areas where multiple UEFs are located within a relatively small geographic region, dismissal times often overlap, causing many vehicles to converge on the same streets within a short period. As a result, pickup activities that are independently managed by individual UEFs can collectively generate severe congestion in surrounding neighborhoods. These observations suggest that effective coordination of pickup activities should consider not only individual UEFs but also neighboring facilities within the same school-centered area.

Designing such coordinated pickup scheduling, however, involves several challenges. First, multiple UEFs may be spatially clustered around the same school area, making it necessary to identify which facilities should be considered together in the coordination process. Second, pickup requests from parents typically arrive over time rather than all at once. Therefore, the system must initiate scheduling based on currently available requests, rather than requiring that all requests be known in advance. Third, once requests are accumulated, an appropriate ordering mechanism is needed to determine the sequence of pickups in a fair and efficient manner. Fourth, the final schedule must assign feasible pickup intervals while satisfying travel-time constraints and street capacity limitations at the pickup locations.

To address these challenges, we propose a four-phase pickup scheduling approach designed to improve coordination among nearby UEFs. In the first phase, *distance range queries* [4–7] are used to identify UEFs located within a specified distance from a given school and having overlapping dismissal times. These facilities are grouped to form a coordination set for scheduling. In the second phase, as parents submit pickup requests over time, the system monitors the accumulation of requests and triggers scheduling when the number of requests reaches a predefined threshold. In the third phase, the requests within each batch are prioritized using one of four scheduling strategies: (1) *request-time-based prioritization*, favoring earlier requests; (2) *travel-time-based prioritization*, favoring shorter travel durations; (3) *request-distribution-based prioritization*, which distributes pickups across spatial zones around the school; and (4) *UEF-distribution-based prioritization*, which balances pickups among facilities located in different zones. In the fourth phase, feasible pickup intervals are assigned to each request based on the determined order while ensuring compliance with travel-time feasibility and vehicle capacity constraints.

This four-phase pickup scheduling approach enables proactive coordination across neighboring UEFs with concurrent dismissal times. It allows parents, especially those residing farther away, to submit requests in advance and receive pickup time slots that align with their travel constraints. We conduct extensive experiments to evaluate multiple scheduling strategies in terms of waiting time fairness and efficiency, specifically using the *standard deviation of waiting times* and the *maximum individual waiting time* as evaluation metrics. The former captures the equity of waiting time distribution across parents, while the latter reflects the worst-case experience in the schedule. In addition, we compare the proposed four-phase pickup scheduling approach with a baseline scenario without any scheduling mechanism, using the *makespan*—the time difference between the earliest parent arrival and the latest parent departure at the UEFs—as an additional metric. The experimental results demonstrate that our method supports the generation of temporally compact and balanced pickup schedules, thereby effectively staggering vehicle arrivals and mitigating traffic congestion around schools in dense urban environments.

The algorithmic innovation of this work lies in a coordinated scheduling framework for managing pickup requests across multiple nearby UEFs. It handles asynchronous request arrivals and generates feasible pickup schedules under travel-time feasibility and street capacity constraints. Based on this framework, the main contributions of this paper are summarized as follows:

- We propose a four-phase pickup scheduling approach that leverages a distance range query to identify clusters of neighboring UEFs with overlapping dismissal periods, providing a foundation for coordinated scheduling across densely distributed educational facilities.
- We design a dynamic batch scheduling mechanism that activates once the number of incoming pickup requests reaches a threshold, enabling responsive coordination without requiring complete knowledge of all requests in advance.
- We develop four prioritization strategies, request-time-based, travel-time-based, request-distribution-based, and UEF-distribution-based, that enable balanced and fair scheduling across spatial and temporal dimensions.
- We conduct experiments to evaluate the proposed four-phase pickup scheduling approach in terms of waiting time fairness, worst-case delay, and schedule duration. Metrics include the standard deviation and maximum waiting time, along with the makespan compared against a baseline without scheduling. Results show that our method produces compact, balanced schedules that help reduce traffic congestion.

To facilitate a comprehensive understanding of our proposed approach, the remainder of this paper is organized as follows. Section 2 reviews prior work on spatial proximity queries, student pickup scheduling approaches, and current applications designed to assist with school transportation. Section 3 introduces the data models for UEFs and road networks that support subsequent spatial analysis and scheduling tasks. Section 4 formally defines the pickup scheduling problem and outlines the objectives and constraints. Section 5 presents our four-phase pickup scheduling approach, including the clustering process, batch scheduling mechanism, request prioritization strategies, and the assignment of pickup intervals. Section 6 reports the results of extensive experiments that evaluate the effectiveness and efficiency of the proposed methods. Finally, Section 7 concludes the paper and outlines directions for future research.

## 2. Related works

This section reviews prior studies related to the pickup coordination problem addressed in this paper from three complementary perspectives: (1) spatial proximity query processing techniques, (2) scheduling and routing methods in educational or urban service contexts, and (3) existing applications designed to assist student pickup operations. These research streams represent the main areas relevant to the proposed problem, which combines spatial facility identification, transportation scheduling, and operational coordination in urban school neighborhoods. Spatial query techniques provide mechanisms for identifying nearby facilities based on spatial relationships, routing and scheduling studies address transportation planning under temporal constraints, and practical pickup systems support operational communication between parents and educational facilities. Reviewing these areas together helps clarify how existing studies address different aspects of the pickup process and where additional coordination mechanisms may be required when multiple nearby urban educational facilities (UEFs) operate simultaneously.

### 2.1. Spatial proximity query techniques

Since the proposed approach utilizes a distance range query to identify nearby UEFs with overlapping dismissal times, we first review spatial proximity queries that support such operations over road networks. These include range queries, nearest neighbor (NN) queries, and spatial keyword queries, which address common tasks in spatial analysis such as retrieving objects within a specified distance from a query location, identifying the closest object to a query point, or locating entities that satisfy both spatial and textual constraints.

Range queries retrieve objects located within a user-defined spatial region [8, 9]. When the query region is centered at the query object, a specialized form known as the distance range query can be applied to return all objects located within a specified distance from the query location [5, 6]. Such queries are widely used in location-based services to identify nearby facilities or points of interest. NN queries identify the object or objects closest to a given query point. Numerous variants have been proposed to support different analytical needs. For example, reverse nearest neighbor (RNN) queries retrieve objects that consider the query point as their nearest neighbor [10, 11], while aggregate nearest neighbor (ANN) queries identify the object that minimizes the aggregate distance to a group of query points [12, 13]. Location-based aggregate (LBA) queries retrieve groups of nearby objects of different types that are close to both the user and one another, based on aggregate distance measures such as average, minimum, maximum, or total travel distance [14]. Spatial keyword queries extend spatial queries by incorporating textual constraints. These queries retrieve spatial objects that are both geographically close to a query location and relevant to specified keywords [15, 16]. Variants include group top-k spatial keyword queries [17], collective spatial keyword queries [18, 19], and group-based collective spatial keyword queries designed for road networks [20].

Over the years, extensive research has been devoted to improving the processing of spatial proximity queries across various application domains, often by extending their core functionality to support more complex and realistic requirements. Some studies [21–24] focused on supporting continuous spatial proximity queries over moving objects, ensuring that results remained valid as either the query location or the data objects changed over time. Uncertain spatial proximity queries [25–28] have also been proposed to handle imprecise or noisy spatial data, aiming to estimate the probability that an object lies within the query region. Furthermore, with the increasing volume of spatial data, distributed processing techniques [29–32] have been developed to support scalable and efficient evaluation of spatial proximity queries across cloud and edge computing environments.

While these spatial proximity query techniques are effective for identifying nearby facilities based on spatial relationships, they are primarily designed for retrieval and spatial analysis tasks. When such queries are applied to student pickup scenarios, additional mechanisms are required to incorporate temporal coordination, request management, and access sequencing among multiple nearby facilities.

### 2.2. Scheduling and routing for student pickup

Prior research on school bus routing and scheduling has evolved along several directions, focusing on improving transportation efficiency while considering operational and educational constraints. One line of work extends classical vehicle routing models to student transportation scenarios, addressing pickup and delivery operations under time window constraints.

Many studies build upon the vehicle routing problem with time windows (VRPTW) introduced by Solomon [33]. These models have been adapted to educational transportation contexts by incorporating time-constrained pickups and coordinated transport resources [34, 35]. Within this framework, researchers have further considered school-specific operational factors, such as aligning pickup schedules with school bell times, supporting student transfers across schools, and balancing route workloads across multiple vehicles [36–39].

Other studies extend school transportation models by incorporating additional planning considerations. For example, affinity relationships among students have been incorporated into school bus routing to improve social compatibility within buses [40]. Multi-objective optimization models have also been proposed to jointly consider school start times and bus routing decisions in order to balance transportation costs and students' travel times [41]. Integrated planning models have further been developed to coordinate extracurricular activity assignments together with transportation deployment across collaborating schools [42].

Uncertainty and dynamic operational conditions have also been investigated. Some works model feeder-bus operations under uncertain weather conditions using chance-constrained frameworks that account for stochastic demand and travel times [43]. Other studies consider dynamic environments by modeling stochastic travel times and variable demand patterns [44, 45]. More recent research explores responsive and real-time coordination approaches that integrate mobile applications for live student feedback or coordinate multiple vehicles under shared constraints, particularly in dense or microtransit urban settings [46–49]. Collectively, these studies demonstrate increasing interest in flexible and context-aware transportation coordination strategies.

Such coordination problems inherently involve determining the order in which service requests are handled under limited operational resources. From this perspective, they can also be interpreted as scheduling problems. Prior studies have widely investigated rule-based strategies for determining job processing order through dispatching or priority rules. Some studies focus on the design of effective dispatching rules for dynamic scheduling environments [50, 51], and other studies investigate comparative evaluations or extensions of rule-based scheduling frameworks [52–54]. However, these studies primarily address job scheduling problems in operational or production environments rather than transportation coordination contexts.

Although these studies provide valuable insights into student transportation planning and coordination, many of them assume that pickup requests are known in advance and focus primarily on routing or scheduling within a single school system. Such assumptions may not fully capture urban pickup environments where multiple educational facilities are located close to one another and dismissal times may overlap. In addition, the asynchronous arrival of parental pickup requests introduces coordination challenges across nearby facilities under real-time operational conditions.

### *2.3. Existing student pickup applications*

In addition to academic research, several real-world applications have been developed to assist schools and parents in managing student pickups. These systems typically provide functions such as digital check-in, real-time tracking, and pickup notifications to support daily operations and enhance communication during the pickup process.

Among the many applications available, we focus on four representative systems that illustrate a range of deployment contexts and design strategies. In the United States, PikMyKid [1] and CurbSmart [2] are

widely adopted for managing campus pickup logistics. PikMyKid offers mobile check-in with dynamic dismissal queues, enabling staff to coordinate vehicle arrivals in real time, while CurbSmart emphasizes barcode-based student matching and real-time carline dashboards. In Taiwan, KIDSDAY [55] provides attendance-based push notifications and supports multiple guardians through individualized pickup logs, whereas PickMeUp [3] allows parents to authorize pickup delegates via the app and integrates with bell schedules to manage staggered dismissals. These systems reflect a growing interest in digital tools that improve safety and efficiency during student release periods.

While these applications effectively assist with check-in and communication, they do not incorporate scheduling mechanisms that coordinate pickups across nearby UEFs or dynamically prioritize requests based on spatial and temporal factors. Most operate as tools for managing on-site flow within a single school, without considering dismissal overlaps, asynchronous request timing, or proximity between neighboring facilities. These limitations highlight the need for scheduling models that can support coordinated decision-making across multiple UEFs in complex urban settings.

## 3. Data modeling

This section defines the spatial and temporal entities involved in the scheduling framework. It begins by introducing five core data tables that describe the road network, urban educational facilities (UEFs), and time-specific dismissal events. These tables together support distance computation and pickup scheduling. An illustrative example is then provided to demonstrate how the data is structured and used in practice.

### Data tables

- **Nodes Table**
  Each record in this table represents a road network node identified by its geographic coordinates.

  `Nodes(NodeID, Latitude, Longitude)`

  - `NodeID` - unique identifier of the node.
  - `Latitude, Longitude` - geographic coordinates of the node.

- **Edges Table**
  This table represents directed road segments connecting pairs of nodes.

  `Edges(EdgeID, FromNodeID, ToNodeID, Length, StreetID)`

  - `EdgeID` - unique identifier of the edge.
  - `FromNodeID, ToNodeID` - node IDs forming a directed edge.
  - `Length` - travel length of the edge.
  - `StreetID` - the street to which the edge belongs.

- **Streets Table**
  This table captures logical streets, each composed of multiple edges.

  `Streets(StreetID, Capacity, PickupDuration)`

- `StreetID` - unique identifier of the street.
- `Capacity` - maximum number of vehicles that can be served simultaneously on a street.
- `PickupDuration` - service time required for each pickup.

- **UEFs Table**
  UEFs are positioned along specific edges.

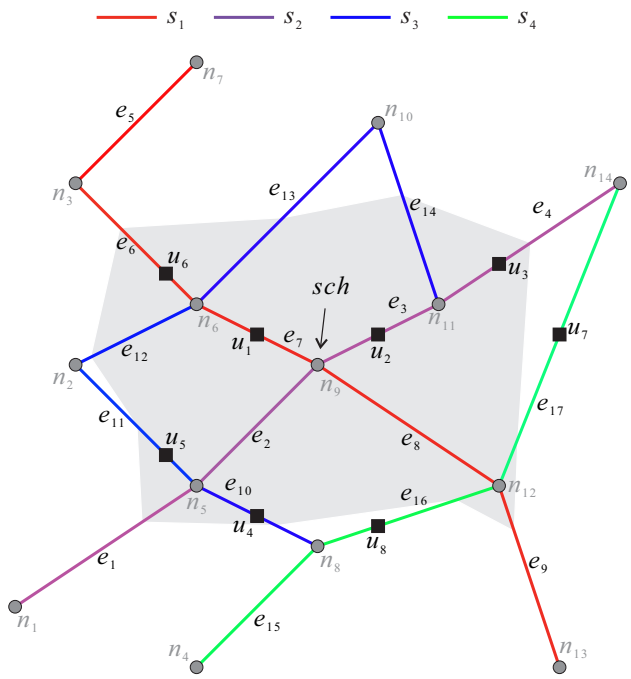
`UEFs(UEFID, EdgeID, PositionRatio, Latitude, Longitude)`

  - `UEFID` - unique identifier of the facility.
  - `EdgeID` - edge on which the UEF is located.
  - `PositionRatio` - relative position along the edge, measured from the FromNodeID (0) toward the ToNodeID (1).
  - `Latitude, Longitude` - geographic coordinates for mapping.

- **DismissalEvents Table**
  Each record defines a time-specific dismissal event at a UEF.

`DismissalEvents(EventID, UEFID, DismissalTime, StudentCount)`

  - `EventID` - unique identifier of the event.
  - `UEFID` - the facility associated with this event.
  - `DismissalTime` - time of student dismissal.
  - `StudentCount` - number of students requiring pickup.




Nodes

| NodeID | Latitude | Longitude |
|---|---|---|
| $n_1$ | 0 | 1 |
| $n_2$ | 1 | 5 |
| ⋮ | ⋮ | ⋮ |
| $n_{14}$ | 10 | 8 |

Streets

| StreetID | Capacity | Pickup Duration |
|---|---|---|
| $s_1$ | 1 | 2 min |
| $s_2$ | 2 | 1 min |
| $s_3$ | 1 | 1 min |
| $s_4$ | 2 | 2 min |

Edges

| EdgeID | From NodeID | ToNodeID | Length | StreetID |
|---|---|---|---|---|
| $e_1$ | $n_1$ | $n_5$ | $\sqrt{13}$ | $s_2$ |
| $e_2$ | $n_5$ | $n_9$ | $2\sqrt{2}$ | $s_2$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $e_{17}$ | $n_{12}$ | $n_{14}$ | $\sqrt{29}$ | $s_4$ |

UEFs

| UEFID | EdgeID | Position Ratio | Latitude | Longitude |
|---|---|---|---|---|
| $u_1$ | $e_7$ | 0.5 | 4 | 5.5 |
| $u_2$ | $e_3$ | 0.5 | 6 | 5.5 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $u_8$ | $e_{16}$ | 0.33 | 6 | 2.3 |

DismissalEvents

| EventID | UEFID | Dismissal Time | Student Count |
|---|---|---|---|
| $evt_1$ | $u_1$ | 20:00 | 2 |
| $evt_2$ | $u_1$ | 20:30 | 4 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| $evt_{11}$ | $u_6$ | 20:30 | 4 |

(a) Road network with UEFs (b) Summary of data tables

**Figure 2.** Data modeling.

**Illustrative example** Figure 2 provides an example that demonstrates how the five data tables represent a road network with UEFs. Panel (a) shows the spatial layout, and panel (b) lists corresponding tuples from each table. The network includes 14 nodes, labeled $n_1$ through $n_{14}$, and 17 directed edges forming the road structure. For example, the tuple $(e_1, n_1, n_5, \sqrt{13}, s_3)$ indicates that edge $e_1$ connects nodes $n_1$ and $n_5$, spans a geometric length of $\sqrt{13}$, and belongs to street $s_3$. Edges are grouped into four streets: $s_1$ (red), $s_2$ (purple), $s_3$ (blue), and $s_4$ (green), each with specific vehicle constraints. For instance, $(s_2, 2, 1 \text{ min})$ indicates that up to two vehicles may operate concurrently on $s_2$, with each pickup taking one minute. UEFs are shown as black squares and positioned along edges using a relative ratio. For example, the tuple $(u_1, e_7, 0.5, 4, 5.5)$ indicates that UEF $u_1$ is located at the midpoint of edge $e_7$, at coordinates (4, 5.5). Dismissal events are tied to UEFs and specify the time and number of students requiring pickup. For instance, the event $evt_1$ represented by the tuple $(evt_1, u_1, 20{:}00, 2)$ indicates a dismissal event at UEF $u_1$ at 8:00 PM for two students. This example illustrates how the proposed data schema represents spatial structure, facility distribution, and time-based demand, forming the basis for scheduling operations.

## 4. Problem definition

In urban school neighborhoods, multiple UEFs dismiss students concurrently at a specific time point $t_d$. Around this time, parents submit pickup requests to retrieve their children. Due to limited road capacity and spatially concentrated pickup activities, a scheduling mechanism is required to assign pickup intervals to parents while considering spatial proximity, time constraints, and congestion control. From a scheduling perspective, the pickup coordination problem can be viewed as a service scheduling task. Each pickup request is interpreted as a job requiring service, while street segments hosting UEFs act as service resources with limited vehicle capacity.

**Spatial scope and UEFs:** The spatial region of interest is defined by a distance range query centered at a reference school, with radius $d$, resulting in a set of relevant UEFs identified at time $t_d$:

$$\text{UEFs}(t_d) = \{u_1, u_2, \ldots, u_n\}$$

Each UEF $u_i \in \text{UEFs}(t_d)$ is located along a specific street segment. Let

$$\text{StreetSet} = \{s_1, s_2, \ldots, s_k\}$$

denote the set of all street segments that contain at least one UEF. A mapping function

$$\text{located\_on}(u_i) = s_j$$

assigns each UEF $u_i$ to its corresponding street segment $s_j \in \text{StreetSet}$, based on the street containing the edge on which the UEF is positioned.

**Pickup requests and travel time estimation:** Each parent submits a pickup request defined as:

$$R_p = \langle p, t_p, l_p, u_p \rangle$$

where:

- $p$: unique identifier of the parent,
- $t_p$: the time the request is issued,
- $l_p$: the parent's current location at request time (e.g., a GPS coordinate),
- $u_p \in \text{UEFs}(t_d)$: the UEF where pickup is requested.

The request must be submitted before the dismissal time, i.e., $t_p < t_d$, to ensure the system has sufficient lead time to process and schedule the pickup.

The travel time $\tau_p$ from $l_p$ to $u_p$ is estimated by the system based on the shortest-path distance in the road network and an assumed travel speed, providing a deterministic approximation of travel time. To ensure schedule feasibility, the system guarantees that any assigned pickup interval for a parent allows sufficient lead time for travel from $l_p$.

**Scheduling output:** The scheduling system produces a pickup assignment:

$$S = \{(p, u_p, [t_p^s, t_p^e])\}$$

Each parent $p$ is assigned a pickup time interval $[t_p^s, t_p^e]$. To reflect operational constraints on different street segments, the duration of each pickup interval is defined as a parameter:

$$\delta(s_j) = t_p^e - t_p^s, \quad \text{where } s_j = \text{located\_on}(u_p)$$

This parameter $\delta(s_j)$ depends on the congestion level of the corresponding street segment $s_j$. In general, streets with higher anticipated congestion are assigned longer pickup durations to reduce overlap among vehicles and ensure safe, orderly pickup operations.

Given an assigned pickup interval $[t_p^s, t_p^e]$ and the estimated travel time $\tau_p$, the required departure time is derived from the start time of the assigned pickup interval as:

$$t_p^{\text{dep}} = t_p^s - \tau_p$$

The waiting time is defined as the duration from the request time to the required departure time. Specifically, the waiting time is computed as:

$$t_p^{\text{wait}} = t_p^{\text{dep}} - t_p$$

To ensure that a parent can reach the UEF on time, the schedule must satisfy the feasibility constraint:

$$t_p^s \geq t_p + \tau_p \tag{4.1}$$

**Capacity constraints:** Each street segment $s_j \in \text{StreetSet}$ is associated with a vehicle capacity $c(s_j)$, which limits the number of parent pickup vehicles that can be present on that street simultaneously. To manage both local and global congestion, the scheduling system enforces the following constraints.

At any given time $t$, let $N_p(s_j, t)$ denote the number of parent pickup vehicles scheduled to occupy street segment $s_j$. The number of vehicles on each street must not exceed its designated capacity:

$$N_p(s_j, t) \leq c(s_j), \quad \forall s_j \in \text{StreetSet} \tag{4.2}$$

In addition to per-street limitations, the system imposes a global constraint on the total number of active pickup vehicles across all street segments at time $t$, bounded by an upper limit $C_{\text{total}}$:

$$\sum_{s_j \in \text{StreetSet}} N_p(s_j, t) \leq C_{\text{total}} \tag{4.3}$$

These constraints ensure that traffic remains manageable both at the street level and across the entire pickup zone. The parameter $C_{\text{total}}$ is a system-level bound reflecting the maximum number of vehicles that can simultaneously participate in pickup activities within the region of interest. Note that

$$C_{\text{total}} \leq \sum_{s_j \in \text{StreetSet}} c(s_j) \tag{4.4}$$

**Evaluation metrics:** To evaluate different scheduling algorithms, we define the following metrics. Let $P$ denote the set of pickup requests.

- Waiting Time Standard Deviation - the variability of waiting times among parents:

$$\sqrt{\frac{1}{|P|}\sum_{p\in P}\left(t_p^{\text{wait}} - \bar{t}^{\text{wait}}\right)^2}, \quad \text{where } \bar{t}^{\text{wait}} = \frac{1}{|P|}\sum_{p\in P} t_p^{\text{wait}} \tag{4.5}$$

  This measure captures the fairness of waiting time distribution, indicating the degree of inequality among parents.
- Maximum Waiting Time - the longest waiting time experienced by any parent:

$$\max_{p\in P} t_p^{\text{wait}} \tag{4.6}$$

  This measure reflects the worst-case delay in the schedule.
- Makespan - the overall schedule duration measured as the time difference between the earliest arrival of any parent at a UEF and the latest departure:

$$\max_{p\in P} t_p^{\text{lev}} - \min_{p\in P} t_p^{\text{arr}} \tag{4.7}$$

  where $t_p^{\text{arr}}$ denotes the actual arrival time of parent $p$ at the assigned UEF, and $t_p^{\text{lev}}$ denotes the time when the pickup is completed and the parent departs. This measure evaluates the temporal compactness of the schedule.

**Problem goal:** Given the spatial concentration of UEFs, the limited capacity of surrounding streets, and the asynchronous nature of parental pickup requests, a scheduling system is needed to coordinate pickups in a way that is both feasible and fair. This study investigates multiple scheduling algorithms for coordinating pickup requests across nearby UEFs. The algorithms aim to:

- Assign feasible pickup intervals that allow sufficient travel time from the parent's location to the assigned UEF,
- Comply with per-street and global vehicle capacity constraints,
- Reduce unfairness in waiting times among parents,

- Support dynamic scheduling mechanisms that accommodate asynchronous arrival of pickup requests.

The quality of the generated schedules is evaluated using the performance metrics, including waiting-time standard deviation, maximum waiting time, and makespan. For clarity, Table 1 summarizes the main symbols used in the problem formulation.

**Table 1.** Notation summary.

| Symbol | Description |
|---|---|
| UEFs($t_d$) | Set of urban educational facilities (UEFs) within distance $d$ from the reference school at dismissal time $t_d$ |
| $u_i$ | The $i$-th UEF in UEFs($t_d$) |
| StreetSet | Set of street segments containing at least one UEF |
| $s_j$ | A street segment in StreetSet |
| $c(s_j)$ | Capacity of street segment $s_j$ |
| $C_{\text{total}}$ | Global capacity limit on the number of pickup vehicles |
| located_on($u_i$) | Mapping function assigning UEF $u_i$ to its hosting street segment $s_j$ |
| $P$ | Set of pickup requests $R_p$ |
| $R_p = \langle p, t_p, l_p, u_p \rangle$ | Pickup request submitted by parent $p$ |
| $t_p$ | Time when parent $p$ submits the pickup request |
| $l_p$ | Location of parent $p$ at request time |
| $u_p$ | UEF where pickup is requested |
| $\tau_p$ | Estimated travel time from $l_p$ to $u_p$ |
| $[t_p^s, t_p^e]$ | Pickup time interval assigned to parent $p$ |
| $\delta(s_j)$ | Duration of pickup intervals on street segment $s_j$ |
| $t_p^{\text{dep}}$ | Required departure time for parent $p$ |
| $t_p^{\text{wait}}$ | Waiting time experienced by parent $p$ |
| $N_p(s_j, t)$ | Number of pickup vehicles occupying street segment $s_j$ at time $t$ |
| $\bar{t}^{\text{wait}}$ | Average waiting time across all parents |
| $t_p^{\text{arr}}$ | Arrival time of parent $p$ at the assigned UEF |
| $t_p^{\text{lev}}$ | Time when parent $p$ completes pickup and leaves |

## 5. Four-phase pickup scheduling approach

To address the challenges of coordinating student pickups across spatially clustered UEFs with overlapping dismissal times, we propose a four-phase scheduling approach based on distance range queries. This method enables the system to proactively identify clusters of nearby UEFs, batch incoming pickup requests based on an estimated request threshold, prioritize requests under different operational considerations, and assign feasible pickup intervals according to spatial and temporal constraints. Algorithm 1 summarizes the four-phase scheduling process. The following subsections describe each phase in detail.

**Algorithm 1** Four-Phase Pickup Scheduling Approach

**Require:** Reference school *sch*, dismissal time $t_d$, distance threshold $d$, incoming pickup requests
**Ensure:** Pickup schedule $S$

**Phase 1:** Identify nearby UEFs UEFs($t_d$) within distance $d$ from *sch*
**Phase 2:** Trigger batch scheduling when the accumulated requests reach the threshold $\theta_r(t_d)$
**Phase 3:** Prioritize the collected pickup requests according to the selected strategy
**Phase 4:** Assign feasible pickup intervals while satisfying travel-time feasibility and street-capacity constraints
**return** pickup schedule $S$

### *5.1. Identifying clusters of nearby UEFs*

To identify the set of UEFs UEFs($t_d$) that require scheduling at a given dismissal time $t_d$, the system performs a distance-bounded traversal of the road network starting from a designated reference school. A queue is maintained to iteratively explore reachable edges, with each traversal step keeping track of the accumulated distance from the starting point.

For each dequeued edge $e$, the algorithm checks whether the cumulative distance, including the length of $e$, remains within the specified threshold $d$. If this condition holds, all UEFs located along $e$ are examined via the `DismissalEvents` table, and those scheduled to dismiss at time $t_d$ are added to the result set UEFs($t_d$). If traversing the entire edge would exceed the threshold, the algorithm still considers partial edge coverage by identifying and including any UEFs located within the remaining allowable distance. This process continues until the queue is empty. The resulting set UEFs($t_d$) thus comprises all UEFs scheduled to dismiss at time $t_d$ and reachable within distance $d$ from the reference school. The pseudocode for this phase is presented in Algorithm 2.

**Algorithm 2** Identify UEFs within Distance $d$ and Dismissal Time $t_d$

**Require:** Reference school *sch*, distance threshold $d$, dismissal time $t_d$
**Ensure:** Set UEFs($t_d$) of qualifying UEFs

Initialize queue with edges adjacent to *sch*
Initialize result set UEFs($t_d$) $\leftarrow \emptyset$
**while** queue is not empty **do**
    Dequeue next edge and compute its accumulated distance from *sch*
    **if** accumulated distance $\leq d$ **then**
        Check UEFs on the edge and add those whose dismissal time equals $t_d$
        Enqueue unvisited outgoing edges and update their accumulated distances
    **else if** partial edge is within $d$ **then**
        Check and include UEFs within range and matching $t_d$
    **end if**
**end while**
**return** UEFs($t_d$)

To illustrate how Algorithm 2 operates, consider again the network in Figure 2, where the reference school *sch* is located at node $n_9$. Let the distance threshold be $d = 4$, and the dismissal time be $t_d = 20{:}00$.

According to Line 1 of the algorithm, the initial queue includes the four edges adjacent to $n_9$: $e_2$, $e_3$, $e_7$, and $e_8$. The result set UEFs($t_d$) is initialized as empty (Line 2). The system dequeues $e_7$ and computes its accumulated distance from *sch*, which equals the length of the edge (Line 4). Since the cumulative distance is less than $d$, the algorithm examines all UEFs on this edge. Here, UEF $u_1$ is located at the midpoint of $e_7$. Consulting the `DismissalEvents` table, $u_1$ is found to have a dismissal event at $t_d$ = 20:00, so it is added to UEFs($t_d$). Then, the algorithm enqueues the outgoing edges from $e_7$'s other endpoint $n_6$, namely, $e_6$, $e_{12}$, and $e_{13}$ (Line 7). This process repeats for each edge in the queue. For example, $e_3$ leads to UEF $u_2$, which is also located at its midpoint. Since $u_2$ dismisses at $t_d$, it is likewise added to the result set. Eventually, the system visits additional edges and discovers more UEFs including $u_3, u_4, u_5, u_6$. However, $u_6$ is excluded because its dismissal time is 20:30. All other UEFs encountered within the reachable region and satisfying the dismissal time condition are added to UEFs($t_d$). After the queue becomes empty (Line 12), the result set contains UEFs($t_d$) = $\{u_1, u_2, u_3, u_4, u_5\}$. These are the UEFs located within the shaded area in Figure 2 and scheduled to dismiss at 20:00.

### *5.2. Batch scheduling triggered by the request threshold*

To determine when to initiate a batch scheduling operation, the system estimates a request threshold based on spatial distribution, student demand, and street-level pickup constraints. This threshold represents the number of incoming requests that can feasibly be assigned within the current dismissal window.

At a given dismissal time $t_d$, let $U(t_d)$ denote the set of UEFs scheduled to dismiss. Each UEF $u_i \in U(t_d)$ is located on a street segment $s_j$, which has a defined vehicle capacity $c(s_j)$ and a fixed pickup duration $\delta(s_j)$. Using the `DismissalEvents` table, the total number of students requiring pickup on each street is computed as:

$$D(s_j, t_d) = \sum_{\substack{u_i \in U(t_d) \\ \text{located\_on}(u_i) = s_j}} \text{StudentCount}(u_i) \tag{5.1}$$

Assuming ideal scheduling conditions where each pickup interval is fully utilized up to the street's vehicle capacity, the minimum time required to complete all pickups on $s_j$ is:

$$T_{\min}(s_j, t_d) = \frac{D(s_j, t_d)}{c(s_j)} \cdot \delta(s_j) \tag{5.2}$$

To guarantee that all pickups across all streets can be scheduled feasibly, the system defines a global scheduling window based on the maximum of the street-level completion times:

$$T_{\min}^{\text{global}}(t_d) = \max_{s_j \in \text{StreetSet}} T_{\min}(s_j, t_d) \tag{5.3}$$

Given this global time budget, the number of pickup intervals each street can accommodate is:

$$I(s_j, t_d) = \left\lfloor \frac{T_{\min}^{\text{global}}(t_d)}{\delta(s_j)} \right\rfloor \tag{5.4}$$

This allows the system to estimate the average number of pickup requests scheduled per interval on street $s_j$ as:

$$A(s_j, t_d) = \frac{D(s_j, t_d)}{I(s_j, t_d)} \tag{5.5}$$

The following lemma establishes that this average does not exceed the street's capacity, ensuring that no overloading occurs under the estimated threshold.

**Lemma 1.** *For any street $s_j \in StreetSet$, the average number of requests per pickup interval satisfies:*

$$A(s_j, t_d) \leq c(s_j) \tag{5.6}$$

*Proof.* By construction, the global time window $T_{\min}^{\text{global}}(t_d)$ is defined as the longest time required by any street to complete its pickups under full capacity. Therefore, each street $s_j$ is guaranteed to have at least as much time as it needs when operating at its own maximum capacity $c(s_j)$. When this global time window is divided into equal pickup intervals of duration $\delta(s_j)$, the number of intervals $I(s_j, t_d)$ is always sufficient to spread out the total demand $D(s_j, t_d)$ without exceeding the per-interval capacity. As a result, the average number of requests per interval, $A(s_j, t_d)$, is always less than or equal to $c(s_j)$, ensuring that the estimated threshold does not cause local street overload. □

Summing over all streets yields a request-based threshold estimate:

$$\theta_r'(t_d) = \sum_{s_j \in \text{StreetSet}} A(s_j, t_d) \tag{5.7}$$

To ensure that the global vehicle capacity is not exceeded, the system applies a final safeguard:

$$\theta_r(t_d) = \min(C_{\text{total}}, \lceil \theta_r'(t_d) \rceil) \tag{5.8}$$

This estimated value $\theta_r(t_d)$ serves as the sole trigger condition for initiating a batch scheduling operation. Once the request threshold $\theta_r(t_d)$ is reached, the system triggers batch scheduling for the current dismissal window. The incoming request that causes the count to meet this threshold is excluded from the batch and retained for future scheduling. At the end of the scheduling process, any remaining requests that do not reach the threshold are also scheduled as a final batch.

Algorithm 3 describes how the system determines when to initiate a batch scheduling operation. First, the request threshold $\theta_r(t_d)$ for dismissal time $t_d$ is computed based on the estimation method described in Section 5.2 (Line 1). The set used to accumulate requests for the current batch is then initialized as empty (Line 2). As pickup requests arrive over time (Line 4), the system checks whether the current batch size has reached the estimated threshold. When the threshold condition is satisfied, the scheduling procedure is triggered for the accumulated requests in the batch (Line 6). The newly arrived request that causes the threshold to be met is not included in that batch and instead becomes the first request of the next batch (Line 7). Otherwise, the request is simply appended to the current batch (Line 9), and the system continues waiting for additional incoming requests.

**Algorithm 3** Trigger Batch Scheduling Based on the Request Threshold

**Require:** Dismissal time $t_d$, incoming request stream
**Ensure:** Trigger decision and request batch for scheduling

```
Compute request threshold θ_r(t_d) based on Section 5.2
Initialize R_batch ← ∅
while system is active do
    Wait for new incoming request R_p = ⟨p, t_p, l_p, u_p⟩
    if |R_batch| + 1 ≥ ⌈θ_r(t_d)⌉ then
        TriggerScheduling(R_batch)
        Reset R_batch ← {R_p}
    else
        R_batch ← R_batch ∪ {R_p}
    end if
end while
```

### *5.3. Prioritizing requests based on strategy*

Once a batch scheduling operation is triggered, the system must determine the processing order of the collected pickup requests in a coordinated and efficient manner. To accommodate different scheduling objectives and operational scenarios, the system supports four prioritization strategies: (1) request-time-based, (2) travel-time-based, (3) request-distribution-based, and (4) UEF-distribution-based. Rather than prescribing a single universally optimal strategy, these approaches represent different operational priorities that may arise in practice. Each strategy reflects a distinct design rationale and aims to balance scheduling fairness, spatial dispersion, or travel feasibility.

To operationalize these strategies, the system defines a unified sorting procedure that applies the appropriate ordering logic based on the specified strategy parameter, as detailed in Algorithm 4. The resulting request order serves as the basis for assigning pickup intervals during the subsequent scheduling phase. The four strategies are described as follows:

- **Request-time-based strategy.** This strategy prioritizes fairness by assigning earlier pickup intervals to parents who submit their requests earlier. Formally, for any two requests $R_{p_1} = \langle p_1, t_{p_1}, l_{p_1}, u_{p_1} \rangle$ and $R_{p_2} = \langle p_2, t_{p_2}, l_{p_2}, u_{p_2} \rangle$. If $t_{p_1} < t_{p_2}$, then $R_{p_1}$ is scheduled before $R_{p_2}$. This approach ensures that parents are rewarded for early submissions and aligns with first-come-first-served principles.
- **Travel-time-based strategy.** This strategy aims to improve schedule feasibility by prioritizing requests with shorter estimated travel times. Let $\tau_p$ denote the travel time for request $R_p$. Requests are sorted in ascending order of $\tau_p$. Since the assigned pickup start time $t_p^s$ must satisfy the feasibility constraint $t_p^s \geq t_p + \tau_p$, requests with larger $\tau_p$ have later earliest feasible pickup start times and therefore more constrained effective scheduling windows. In practice, scheduling is performed within a limited time horizon under capacity constraints, so later feasible start times reduce the number of available assignment opportunities and thus limit scheduling flexibility. Prioritizing requests with shorter travel times preserves greater flexibility in assigning feasible pickup intervals, while preserving sufficient scheduling slack for requests with longer travel times to satisfy the feasibility constraint.

- **Request-distribution-based strategy.** This strategy considers spatial dispersion and parking load balance by dividing the area around the reference school into four quadrants based on geometric coordinates. Each request is mapped to a quadrant according to the parent's location $l_p$. Within each triggered batch, requests assigned to the same quadrant are placed into separate queues, and the final order is obtained by cyclically selecting one request from each non-empty quadrant in a round-robin manner. By alternating pickup intervals among spatial zones, the system avoids having multiple vehicles from the same direction converge simultaneously, thus reducing localized congestion near the school.
- **UEF-distribution-based strategy.** This strategy seeks to balance pickup activity across street segments. Each UEF $u \in \text{UEFs}(t_d)$ is associated with a street segment $s_j = \text{located\_on}(u)$. Within each triggered batch, requests are first grouped according to the street segment of their destination UEF. Requests from the same street are placed into the same queue, and the scheduling procedure generates the final order by cyclically selecting one request from each non-empty street queue in a round-robin manner. This approach distributes pickup operations across different streets, preventing excessive concentration on any single road segment and helping maintain traffic stability in constrained urban environments.

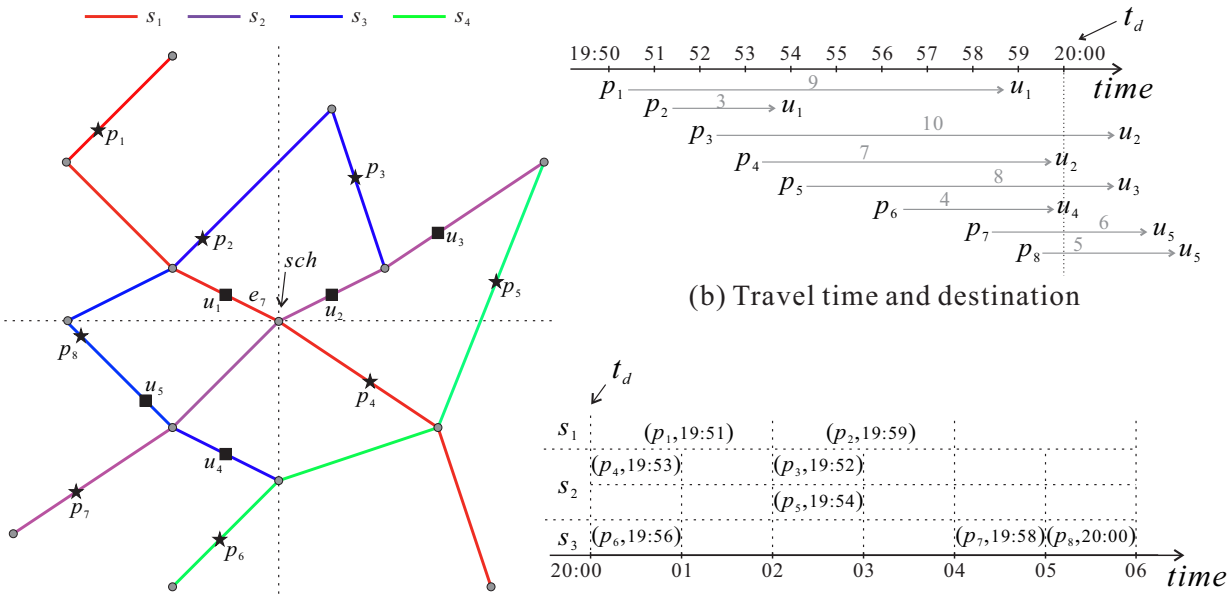
**Algorithm 4** Sort Requests Based on Selected Prioritization Strategy

**Require:** Batch of requests $R_{\text{batch}}$, strategy parameter $stg$
**Ensure:** Ordered list of requests $R_{\text{sorted}}$

**if** $stg$ = request-time-based **then**
  $R_{\text{sorted}} \leftarrow$ Sort $R_{\text{batch}}$ by ascending $t_p$
**else if** $stg$ = travel-time-based **then**
  **for** each $R_p \in R_{\text{batch}}$ **do**
    Compute travel time $\tau_p$
  **end for**
  $R_{\text{sorted}} \leftarrow$ Sort $R_{\text{batch}}$ by ascending $\tau_p$
**else if** $stg$ = request-distribution-based **then**
  Divide area around school into four quadrants
  **for** each $R_p \in R_{\text{batch}}$ **do**
    Assign $R_p$ to a quadrant based on location $l_p$
  **end for**
  Cyclically initialize round-robin queue $Q_{\text{quadrants}} = \{Q_1, Q_2, Q_3, Q_4\}$
  $R_{\text{sorted}} \leftarrow$ Interleave requests from quadrants in round-robin order
**else if** $stg$ = UEF-distribution-based **then**
  **for** each $R_p \in R_{\text{batch}}$ **do**
    Determine street segment $s_j = \text{located\_on}(u_p)$
    $Q_{s_j} \leftarrow Q_{s_j} \cup \{R_p\}$
  **end for**
  Cyclically initialize round-robin queue $Q_{\text{streets}} = \{Q_{s_1}, Q_{s_2}, \dots\}$
  $R_{\text{sorted}} \leftarrow$ Interleave requests from streets in round-robin order
**end if**
**return** $R_{\text{sorted}}$

To illustrate the effects of different prioritization strategies, we extend the example in Figure 2 with the scenario shown in Figure 3. In this case, eight parents $p_1$ through $p_8$ issue pickup requests to five UEFs distributed across different street segments, as depicted in Figure 3(a). Each parent is associated with a target UEF and a corresponding travel time, as illustrated in Figure 3(b). For instance, parent $p_1$is scheduled to pick up at $u_1$. The coordinates of $u_1$ are (4,5.5) (see Figure 2), and the estimated travel time from $l_{p_1}$ to $u_{p_1} = u_1$ is $\tau_{p_1} = 9$ minutes as shown in Figure 3(b). According to the batch scheduling mechanism introduced in Section 5.2, scheduling is triggered every time three requests are collected. As such, the eight requests are grouped into three batches: $\{p_1, p_2, p_3\}$, $\{p_4, p_5, p_6\}$, and $\{p_7, p_8\}$.

When applying the request-time-based strategy, each batch is ordered by the time of request submission, resulting in the sequences $p_1 \to p_2 \to p_3$, $p_4 \to p_5 \to p_6$, and $p_7 \to p_8$, respectively. In contrast, the travel-time-based strategy favors parents with shorter travel durations, producing the orderings $p_2 \to p_1 \to p_3$, $p_6 \to p_4 \to p_5$, and $p_8 \to p_7$. The request-distribution-based strategy considers spatial balance by assigning each parent to one of four quadrants centered at the reference school, and rotating the quadrant priority for each batch. Using a counterclockwise sequence starting from the first quadrant (northeast), the resulting request orders are $p_3 \to p_1 \to p_2$, $p_6 \to p_4 \to p_5$, and $p_7 \to p_8$. Lastly, the UEF-distribution-based strategy organizes requests according to the street segment on which the target UEF is located. Requests are first grouped by street, and each batch is ordered using a rotating street priority in a round-robin manner. The resulting sequences are $p_1 \to p_3 \to p_2$, $p_4 \to p_6 \to p_5$, and $p_7 \to p_8$.




(a) Locations of requests and UEFs (b) Travel time and destination (c) Request-time-based schedule

**Figure 3.** Requests, travel times, and request-time-based schedule.

### 5.4. Assigning pickup intervals to requests

Given a triggered batch of requests, the system sequentially assigns pickup intervals to each request according to the prioritized order determined in the previous phase (Line 1). For each request $R_p = \langle p, t_p, l_p, u_p \rangle$, the system identifies the street $s_j$ on which the assigned UEF $u_p$ is located, retrieves the pickup duration $\delta(s_j)$ and capacity $c(s_j)$ (Lines 2–4), and computes the parent's earliest feasible arrival time based on the estimated travel time $\tau_p$, yielding the readiness time $t_p^{\text{ready}} = t_p + \tau_p$ (Line 4).

The system then searches for the earliest feasible pickup interval $[t, t + \delta(s_j)]$, starting from the dismissal time $t_d$ and progressing in increments of $\delta(s_j)$ (Line 5). For each candidate interval, two feasibility conditions are checked: (1) the number of already assigned pickups in that interval does not exceed the capacity $c(s_j)$, and (2) the interval's start time is no earlier than the parent's readiness time $t_p^{\text{ready}}$ (Line 7). Once a valid interval is found, it is assigned to the parent as $[t_p^s, t_p^e]$, and the interval's usage count is updated accordingly (Lines 8–9). This process continues until all requests in the batch have been assigned pickup intervals.

**Algorithm 5** Assign Pickup Intervals to Requests

**Require:** Ordered list of requests $R_{\text{sorted}}$, dismissal time $t_d$, travel times $\tau_p$, street metadata
**Ensure:** Assigned pickup intervals $[t_p^s, t_p^e]$ for each $R_p \in R_{\text{sorted}}$

1: **for** each request $R_p = \langle p, t_p, l_p, u_p \rangle \in R_{\text{sorted}}$ **do**
2: Identify street $s_j$ associated with UEF $u_p$
3: Retrieve pickup duration $\delta(s_j)$ and capacity $c(s_j)$
4: Compute readiness time: $t_p^{\text{ready}} \leftarrow t_p + \tau_p$
5: Initialize interval start $t \leftarrow t_d$
6: **while** true **do**
7: **if** interval $[t, t + \delta(s_j)]$ has fewer than $c(s_j)$ assigned pickups **and** $t \geq t_p^{\text{ready}}$ **then**
8: Assign pickup interval: $t_p^s \leftarrow t$, $t_p^e \leftarrow t + \delta(s_j)$
9: Increment usage count of interval $[t, t + \delta(s_j)]$ on $s_j$
10: **break**
11: **else**
12: $t \leftarrow t + \delta(s_j)$
13: **end if**
14: **end while**
15: **end for**

Continuing the example from Section 5.3, using the request-time-based strategy, Figure 3(c) lists each scheduled request in the form $(p_i, t_{p_i}^{\text{dep}})$, where $t_{p_i}^{\text{dep}} = t_{p_i}^s - \tau_{p_i}$ is the assigned departure time. For example, $(p_1, 19{:}51)$ means that $p_1$ is scheduled to depart at 19:51. Requests are processed in order $p_1 \to p_2 \to p_3$, $p_4 \to p_5 \to p_6$, and $p_7 \to p_8$. On $s_1$, $p_1$ is placed in the 20:00 slot (19:51); $p_2$ would fit there but the slot is full, so it is deferred to 20:02 (19:59). On $s_2$, $p_4$ takes 20:00 (19:53), while $p_3$ and $p_5$ share 20:02 with (19:52) and (19:54). On $s_3$, $p_6$ is assigned to 20:00 (19:56), $p_7$ to 20:04 (19:58), and $p_8$ to 20:05 (20:00). Each request is assigned to the earliest available slot on its street that meets both readiness-time and capacity constraints. Consequently, the final schedule is $\{(p_1, 19{:}51), (p_2, 19{:}59), (p_3, 19{:}52), (p_4, 19{:}53), (p_5, 19{:}54), (p_6, 19{:}56), (p_7, 19{:}58), (p_8, 20{:}00)\}$.

## 6. Performance evaluation

In this section, three sets of experiments are conducted to evaluate the proposed four-phase pickup scheduling approach. The first set compares four prioritization strategies under different travel times and UEF distributions, measuring both the standard deviation and maximum of waiting times. The second set examines the impact of varying the UEF count and parent count on makespan. The third set investigates how spatial and operational parameters, including the distance threshold $d$, pickup duration $\delta(s_j)$, and street capacity $c(s_j)$, affect makespan. We first describe the performance settings and then present the experimental results with detailed discussions, respectively.

### *6.1. Computational and experimental environment*

All experiments are performed on a PC with an Intel Core i7 CPU (3.40 GHz) and 64 GB RAM, and the algorithms are implemented in Java. The experimental road network is extracted from OpenStreetMap [56] for the Taipei City area in Taiwan, containing approximately 35,000 nodes and 37,000 edges. Within this network, 30 schools are selected as reference points for scheduling. UEFs are generated within a distance threshold $d$ from the school, where $d$ is defined as the number of edges traversed along the road network, following the approach proposed in [57]. The value of $d$ ranges from 10 to 50. The number of UEFs per school, denoted by $n_u$, varies between 10 and 200. In the same area, $n_p$ parent locations (i.e., request origins) are generated, with $n_p$ ranging from 100 to 2000, and each parent issues a pickup request at a time $t \in [0, 10]$ time units. Note that the number of pickup requests associated with each UEF represents only the subset of parents who actively use the pickup scheduling service, rather than the total number of students in that facility. For each street $s_j$, the pickup duration $\delta(s_j)$ is randomly assigned within $[1, \delta_M]$ time units, where $\delta_M$ ranges from 1 to 5. The travel time $\tau_p$ from a parent's location to the assigned UEF is generated within $[1, \tau_M]$ time units, where $\tau_M$ ranges from 5 to 100. Time-related parameters are represented using abstract simulation time units rather than fixed real-world minutes, since the evaluation focuses on comparing the relative performance of different scheduling strategies under the same time scale. The per-street vehicle capacity $c(s_j)$ is randomly set within $[1, c_M]$, where $c_M$ ranges from 1 to 9. Table 2 summarizes the experimental parameters, including their default values and ranges.

**Table 2.** System parameters.

| Parameter | Default | Range |
|---|---|---|
| Number of UEFs ($n_u$) | 50 | 10, 30, 50, 100, 200 |
| Number of parents ($n_p$) | 500 | 100, 200, 500, 1000, 2000 |
| Distance threshold ($d$) | 30 | 10, 20, 30, 40, 50 |
| Pickup duration ($\delta_M$) | 3 | 1, 2, 3, 4, 5 |
| Travel time ($\tau_M$) | 30 | 5, 10, 30, 50, 100 |
| Street capacity ($c_M$) | 5 | 1, 3, 5, 7, 9 |

For performance evaluation, we execute the scheduling process for all 30 reference schools and record the results using three metrics defined in Section 4: (1) the standard deviation of waiting times, (2) the maximum individual waiting time, and (3) the makespan, i.e., the time difference between the earliest parent arrival and the latest parent departure at the UEFs. These metrics respectively measure

the fairness of waiting time distribution, the worst-case waiting time experienced by any parent, and the overall temporal span of the schedule.

From the students' perspective, waiting time corresponds to the duration between the dismissal time and the pickup time. Let $W_p$ and $W_s$ denote the parent-side and student-side waiting times, respectively, where $t_p$, $\tau_p$, and $t_d$ represent the request time, travel time, and dismissal time. Their relationship can be expressed as $W_s = W_p + (t_p + \tau_p - t_d)$. This relation indicates that the student-side waiting time is directly related to the parent-side waiting time through the difference between the parent's arrival time ($t_p + \tau_p$) and the dismissal time. Since the dismissal time is fixed for students at the same UEF, variations in scheduling outcomes (such as the maximum waiting time and the variability of waiting times) can also be interpreted through the parent-side waiting metrics defined above.

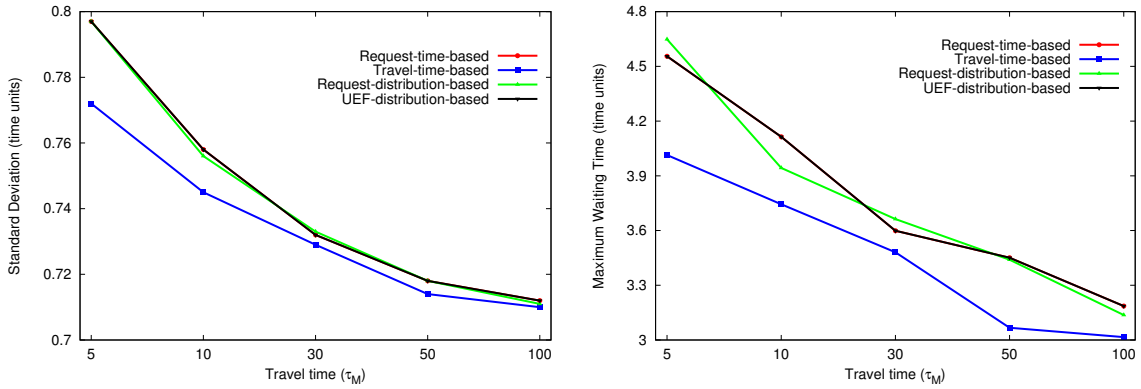
To evaluate the effectiveness of the proposed scheduling approach, we compare it with a baseline scenario without coordinated scheduling. In this baseline, each parent departs immediately after issuing a pickup request and arrives near the UEF after the corresponding travel time. Pickups are then processed according to the earliest arrival time while satisfying the capacity constraint of the corresponding street.

### *6.2. Comparison of scheduling strategies*

This section compares the performances of the four proposed prioritization strategies under two sets of experiments. Since the four strategies mainly affect the prioritization of pickup requests rather than the overall schedule span, the evaluation focuses on waiting-time metrics, which better reflect differences among the strategies. The first experiment investigates the impact of the maximum travel time parameter ($\tau_M$), while the second examines the effect of different spatial distributions of UEFs around the school. In all experiments, unless otherwise specified, the parameter settings follow the default values described in Section 6.1, ensuring a consistent evaluation environment.

To evaluate the effect of the maximum travel time parameter ($\tau_M$), Figure 4 presents the results when $\tau_M$ varies from 5 to 100, while all other parameters follow the default settings in Section 6.1. For both performance metrics, the standard deviation of waiting time and the maximum waiting time, all four strategies exhibit a decreasing trend as $\tau_M$ increases. This is because a larger $\tau_M$ increases the variation in travel times among requests, which provides more flexibility for assigning pickup intervals while satisfying the feasibility constraint, thereby reducing temporal disparities among requests. Among the four strategies, the travel-time-based strategy consistently achieves the smallest values for both metrics, as it directly prioritizes minimizing travel time differences, which also indirectly reduces waiting time variability. In contrast, the other three strategies show similar but slightly higher values, with the request-distribution-based approach generally performing better than the UEF-distribution-based and request-time-based strategies for smaller $\tau_M$ values. As $\tau_M$ increases, the performance gap among these three strategies narrows, while the travel-time-based strategy maintains a clear advantage across all settings.

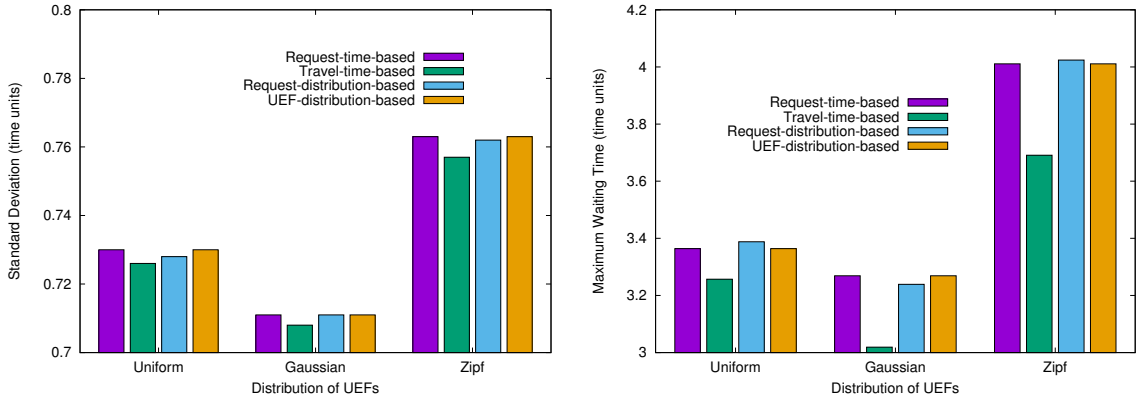
The second experiment evaluates how different spatial distributions of UEFs around the school influence the performance of the four prioritization strategies. Three types of distributions are considered: *Uniform*, where UEFs are evenly scattered within the target area; *Gaussian*, where UEFs are more concentrated near the school; and *Zipf*, where UEFs are predominantly clustered in a specific quadrant.




**(a)** Standard deviation of waiting time **(b)** Maximum waiting time

**Figure 4.** Effect of travel time.

Figure 5(a) and Figure 5(b) present the results for the standard deviation of waiting time and the maximum waiting time, respectively. For both metrics, the *Uniform* and *Gaussian* cases exhibit relatively low values, with the Gaussian distribution producing the most consistent schedules due to shorter and more homogeneous travel times from parents to assigned UEFs. By contrast, the *Zipf* distribution leads to markedly higher variability and longer delays, as the heavy concentration of UEFs in one area creates imbalanced demand across streets and congestion around over-concentrated locations. Among the four strategies, the travel-time-based method consistently performs best in reducing both variability and extreme delays, whereas the request-time-based approach tends to produce larger peaks, particularly under skewed spatial distributions.




**(a)** Standard deviation of waiting time **(b)** Maximum waiting time

**Figure 5.** Effect of UEF spatial distribution.

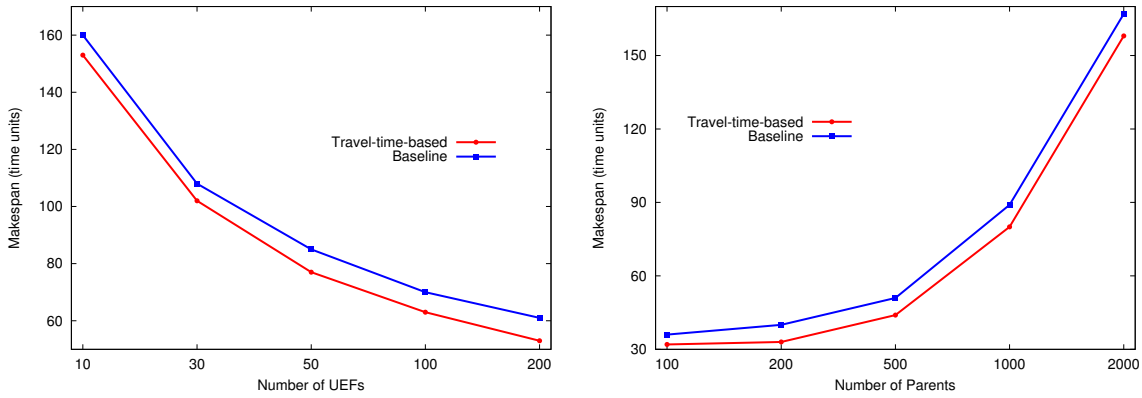
### *6.3. Impact of UEF count and parent count*

Building on the previous set of experiments, the travel-time-based strategy consistently demonstrated superior performance over the other scheduling approaches. In this section, we therefore compare the travel-time-based strategy against the baseline method under varying numbers of UEFs ($n_u$) and parents

($n_p$). For both methods, the makespan is defined as the time interval from the arrival of the first parent at their assigned UEF until the departure of the last parent after completing the pickup.

Figure 6(a) shows that increasing $n_u$ naturally results in a more dispersed spatial distribution of requests across streets, which lowers the per-street load and shortens service queues. This wider distribution enables earlier completion of all pickups, as each street can process a smaller set of requests in parallel. The travel-time-based strategy consistently achieves smaller makespans by aligning arrivals with service start times and avoiding premature occupation of capacity, thereby reducing idle waiting periods. Conversely, Figure 6(b) indicates that a larger $n_p$ leads to longer makespans, since more requests per street increase queue lengths and cause cumulative delays before the last parent departs. Although both methods exhibit this upward trend, the baseline is more affected due to inefficient sequencing and idle waiting from early arrivals, whereas the travel-time-based approach regulates arrival timing more effectively, maintaining a clear advantage under all tested settings.

In summary, the travel-time-based strategy maintains a consistent advantage across varying demand patterns, suggesting its potential to support more orderly and predictable pickups. Such an approach could be useful for easing congestion and improving overall traffic conditions in school neighborhoods.




**(a)** Varying $n_u$ **(b)** Varying $n_p$

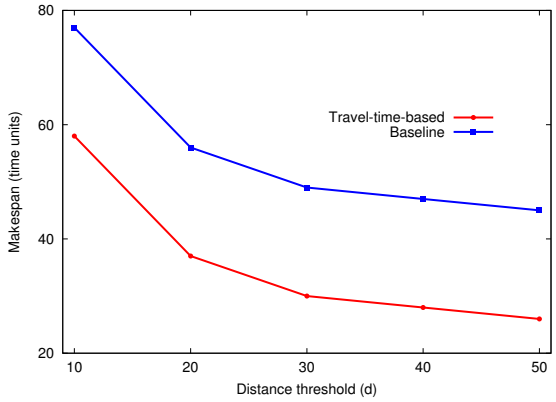
**Figure 6.** Impact of UEF count and parent count.

### *6.4. Impact of spatial and operational parameters*

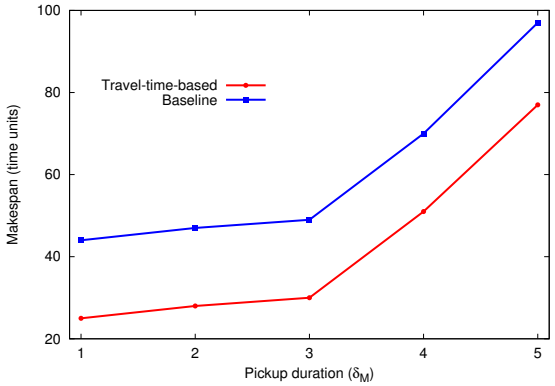
This set of experiments examines three parameters, including the distance threshold ($d$), the maximum pickup duration ($\delta_M$), and the maximum street capacity ($c_M$). The distance threshold specifies the spatial range used to identify UEFs for scheduling. The maximum pickup duration defines the time allocated to complete each pickup on a street segment. The maximum street capacity determines the number of vehicles that can simultaneously operate on a street. These parameters collectively influence the scope of scheduling, the temporal allocation of pickups, and the management of vehicle flow.

Figure 7(a) illustrates that the makespan decreases as the distance threshold $d$ increases. This is because a larger search range disperses UEFs across more streets, which reduces the number of facilities and pickups per street and alleviates per-street loading. In Figure 7(b), the makespan shows an upward trend as the maximum pickup duration $\delta_M$ increases. The reason is that longer service times on a street inevitably delay subsequent pickups assigned to the same street, and these sequential delays accumulate

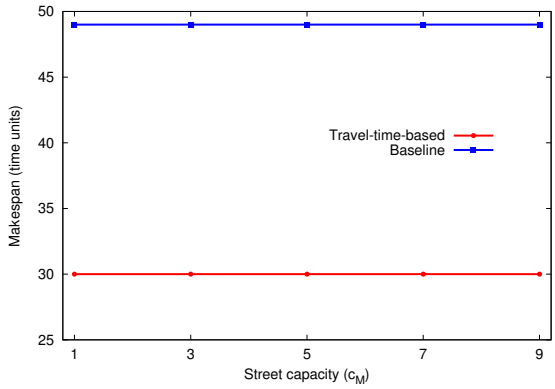
throughout the schedule. For Figure 7(c), varying the maximum street capacity $c_M$ produces almost no change in makespan. This can be attributed to the fact that the schedule length is mainly determined by the interval between the earliest parent's arrival and the last parent's assigned pickup time, where the latter depends on the time that a parent can reach the UEF rather than on changes in street capacity. Across all three figures, the makespan difference between the travel-time-based method and the baseline remains constant, because in the baseline, the earliest-arriving parent must wait at the UEF until the assigned pickup time, whereas in the travel-time-based method, the schedule times the parent's arrival just before pickup. This waiting period in the baseline is unaffected by changes in $d$, $\delta_M$, or $c_M$, so the relative gap in schedule length between the two methods does not vary with these parameters.




**(a)** Varying $d$




**(b)** Varying $\delta_M$




**(c)** Varying $c_M$

**Figure 7.** Impact of spatial and operational parameters.

## 7. Conclusions

This paper addresses the problem of uncoordinated parental pickups around spatially clustered UEFs with overlapping dismissal times. To address this issue, we propose a four-phase scheduling approach based on distance range queries, including identifying relevant UEFs, triggering batch scheduling using a request-count threshold, prioritizing requests through multiple strategies, and assigning pickup intervals under travel-time feasibility and capacity constraints. Experiments under varying spatial and

operational parameters showed that the proposed approach, particularly the travel-time-based strategy, effectively reduced waiting-time variability, maximum waiting time, and makespan compared with a baseline without scheduling. These results demonstrate the method's ability to stagger arrivals, shorten idle vehicle time, and improve pickup coordination in dense urban neighborhoods.

Future work may investigate stochastic travel-time variations and conduct sensitivity analysis to evaluate the robustness of the proposed scheduling approach under uncertain traffic conditions. Another direction is to incorporate improvement heuristics, such as local search-based strategies (e.g., reinsertion or swap operations), to further refine the constructed schedules. In addition, exploring more advanced optimization frameworks and alternative coordination strategies across UEFs represents an interesting direction for future research.

## Use of Generative-AI tools declaration

The author(s) declare(s) that they have used Artificial Intelligence (AI) tools in the creation of this article.

AI tools used: ChatGPT (OpenAI)

How were the AI tools used? ChatGPT was used for English language editing and grammar correction. All technical content, analysis, and conclusions were developed and verified by the authors.

Where in the article is the information located? Throughout the manuscript.

## Acknowledgments

This work was supported by the National Science and Technology Council, Taiwan, under Grants NSTC 113-2622-M-992-001 and NSTC 114-2622-M-992-001.

## Conflict of interest

The authors declare that they have no conflict of interest.

## References

1. PikMyKid, Pikmykid [mobile app], App Store, 2025. Available from: https://apps.apple.com/us/app/pikmykid-parent/id1559538995. Accessed: 2025-07-21.
2. CurbSmart, Curbsmart [mobile app], App Store, 2025. Available from: https://apps.apple.com/tw/app/curbsmart/id1235680339. Accessed: 2025-07-21.
3. PickMeUp, Pickmeup [mobile app], App Store, 2020. Available from: https://apps.apple.com/tw/app/pick-me-up/id1447715891?l=en-GB. Accessed: 2025-07-21.
4. M. F. Mokbel, X. Xiong, W. G. Aref, Sina: Scalable incremental processing of continuous queries in spatio-temporal databases, *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, (2004), 623–634. https://doi.org/10.1145/1007568.1007638
5. Y. K. Huang, L. F. Lin, Continuous within query in road networks, *2011 7th International Wireless Communications and Mobile Computing Conference, Istanbul, Turkey*, (2011), 1176–1181. https://doi.org/10.1109/IWCMC.2011.5982707

6. A. D. Zhu, H. Ma, X. Xiao, S. Luo, Y. Tang, S. Zhou, Shortest path and distance queries on road networks: towards bridging theory and practice, *SIGMOD '13: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, (2013), 857–868. https://doi.org/10.1145/2463676.2465277
7. A. Michalopoulos, D. Tsitsigkos, P. Bouros, N. Mamoulis, M. Terrovitis, Efficient distance queries on non-point data, *ACM Trans. Spatial Algorithms Syst.*, **11** (2024), 1–37. https://doi.org/10.1145/3698194
8. H. Samet, *Foundations of Multidimensional and Metric Data Structures*, Morgan Kaufmann, San Francisco, CA, 2006.
9. D. Taniar, W. Rahayu, A taxonomy for region queries in spatial databases, *J. Comput. Syst. Sci.*, **81** (2015), 1508–1531. https://doi.org/10.1016/j.jcss.2014.12.025
10. Y. Tao, M. L. Yiu, N. Mamoulis, Reverse nearest neighbor search in metric spaces, *IEEE Trans. Knowl. Data Eng.*, **18** (2006), 1239–1252. https://doi.org/10.1109/TKDE.2006.148
11. W. Li, M. Cai, M. Gao, D. Wen, L. Qin, W. Wang, Expanding reverse nearest neighbors, *Proceedings of the VLDB Endowment*, **17** (2023), 630–642. https://doi.org/10.14778/3636218.3636220
12. D. Papadias, Y. Tao, K. Mouratidis, C. K. Hui, Aggregate nearest neighbor queries in spatial databases, *ACM Trans. Database Syst.*, **30** (2005), 529–576. https://doi.org/10.1145/1071610.1071616
13. C. F. Costa, J. Machado, J. A. F. Macêdo, M. A. Nascimento, Aggregate k-nearest neighbors queries in time-dependent road networks, in *International Conference on Advances in Geographic Information Systems*, 2015. https://doi.org/10.1145/2834126.2834129
14. Y. K. Huang, Location-based aggregate queries for heterogeneous neighboring objects, *IEEE Access*, **5** (2017), 4887–4900. https://doi.org/10.1109/ACCESS.2017.2688472
15. I. De Felipe, V. Hristidis, N. Rishe, Keyword search on spatial databases, in *24th IEEE International Conference on Data Engineering*, 2008, 656–665. https://doi.org/10.1109/ICDE.2008.4497474
16. X. Cao, G. Cong, C. S. Jensen, Retrieving top-k prestige-based relevant spatial web objects, *Proceedings of the VLDB Endowment*, **3** (2010), 373–384. https://doi.org/10.14778/1920841.1920891
17. H. B. Ekomie, K. Yao, J. Li, G. Li, Y. Li, Group top-k spatial keyword query processing in road networks, in *Database and Expert Systems Applications*, 2017, 395–408. https://doi.org/10.1007/978-3-319-64468-4_30
18. X. Cao, G. Cong, C. S. Jensen, B. C. Ooi, Collective spatial keyword querying, in *ACM SIGMOD Conference*, 2011, 373–384. https://doi.org/10.1145/1989323.1989363
19. X. Cao, G. Cong, T. Guo, C. S. Jensen, B. C. Ooi, Efficient processing of spatial group keyword queries, *ACM Trans. Database Syst.*, **40** (2015), 1–48. https://doi.org/10.1145/2772600
20. S. Su, S. Zhao, X. Cheng, R. Bi, X. Cao, J. Wang, Group-based collective keyword querying in road networks, *Inf. Process. Lett.*, **118** (2017), 83–90. https://doi.org/10.1016/j.ipl.2016.10.008
21. D. V. Kalashnikov, S. Prabhakar, S. Hambrusch, W. G. Aref, Efficient evaluation of continuous range queries on moving objects, in *Database and Expert Systems Applications*, 2002, 731–740. https://doi.org/10.1007/3-540-46146-9_72

22. Y. K. Huang, L. F. Lin, Efficient processing of continuous min–max distance bounded query with updates in road networks, *Inf. Sci.*, **278** (2014), 187–205. https://doi.org/10.1016/j.ins.2014.03.040

23. M. Orabi, Z. Al Aghbari, I. Kamel, Skyeye: continuous processing of moving spatial-keyword queries over moving objects, *GeoInformatica*, **28** (2024), 559–603. https://doi.org/10.1007/s10707-024-00512-0

24. Z. Yu, X. Yu, T. Zhou, Y. Chen, Y. Liu, B. Li, Odin: Object density aware index for cknn queries over moving objects on road networks, *IEEE Trans. Knowl. Data Eng.*, **36** (2024), 6758–6771. https://doi.org/10.1109/TKDE.2023.3344662

25. Y. K. Huang, C. Lee, Efficient evaluation of continuous spatio-temporal queries on moving objects with uncertain velocity, *GeoInformatica*, **14** (2010), 163–200. https://doi.org/10.1007/s10707-009-0081-8

26. K. Zheng, G. Trajcevski, X. Zhou, P. Scheuermann, Probabilistic range queries for uncertain trajectories on road networks, in *International Conference on Extending Database Technology*, 2011, 283–294. https://doi.org/10.1145/1951365.1951400

27. J. Bernad, C. Bobed, E. Mena, Uncertain probabilistic range queries on multidimensional data, *Inf. Sci.*, **537** (2020), 334–367. https://doi.org/10.1016/j.ins.2020.05.068

28. A. Züfle, Uncertain spatial data management: An overview, in *Handbook of Big Geospatial Data*, eds. M. Werner and Y. Y. Chiang, 2021, 355–378. https://doi.org/10.1007/978-3-030-55462-0_14

29. F. Lettich, S. Orlando, C. Silvestri, C. S. Jensen, Manycore processing of repeated range queries over massive moving objects observations, arXiv preprint arXiv:1411.3212, 2022. https://doi.org/10.48550/arXiv.1411.3212

30. H. Zhu, Z. Yu, Distributed processing of continuous range queries over moving objects, in *Intelligent Computing*, 2022, 403–418. https://doi.org/10.1007/978-981-19-8915-5_35

31. A. Bhandari, P. Hamandawana, M. Attique, H. J. Cho, T. S. Chung, Parscl: A parallel and distributed framework to process all nearest neighbor queries on a road network, *IEEE Access*, **11** (2023), 94043–94055. https://doi.org/10.1109/ACCESS.2023.3308684

32. A. Karabinos, P. Tampakis, C. Doukelis, A. Vlachou, Processing of spatial-keyword range queries in apache spark, in *11th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, 2023. https://doi.org/10.1145/3615833.3628592

33. M. M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, *Oper. Res.*, **35** (1987), 254–265. https://doi.org/10.1287/opre.35.2.254

34. J. Park, B. I. Kim, The school bus routing problem: A review, *Eur. J. Oper. Res.*, **202** (2010), 311–319. https://doi.org/10.1016/j.ejor.2009.05.017

35. C. K. Y. Lin, A vehicle routing problem with pickup and delivery time windows, and coordination of transportable resources, *Comput. Oper. Res.*, **38** (2011), 1596–1609. https://doi.org/10.1016/j.cor.2011.01.021

36. M. Bögl, K. F. Doerner, S. N. Parragh, The school bus routing and scheduling problem with transfers, *Networks*, **65** (2015), 180–203. https://doi.org/10.1002/net.21589

37. X. Chen, Y. Kong, L. Dang, Y. Hou, X. Ye, Exact and metaheuristic approaches for a bi-objective school bus scheduling problem, *PLOS ONE*, **10** (2015), 1–20. https://doi.org/10.1371/journal.pone.0132600
38. A. Shafahi, S. Aliari, A. Haghani, Balanced scheduling of school bus trips using a perfect matching heuristic, *Transp. Res. Rec.*, **2672** (2018), 1–11. https://doi.org/10.1177/0361198118787803
39. L. Zeng, S. Chopra, K. Smilowitz, A bounded formulation for the school bus scheduling problem, *Transp. Sci.*, **56** (2022), 1148–1164. https://doi.org/10.1287/trsc.2022.1130
40. J. P. Orejuela Cabrera, M. A. Londoño, V. L. C. Pantoja, School bus routing problem considering affinity among children, *Decis. Sci. Lett.*, **10** (2021), 535–548. https://doi.org/10.5267/j.dsl.2021.5.002
41. S. Ezquerro-Eguizábal, J. L. Moura-Berodia, A. Ibeas-Portilla, J. Benavente-Ponce, Optimization model for school transportation design based on economic and social efficiency, *Transp. Policy*, **67** (2018), 93–101. https://doi.org/10.1016/j.tranpol.2018.01.015
42. M. Mateo-Doll, E. Aghezzaf, Optimizing extracurricular activities assignment and related logistical deployment cost between collaborating schools, *Int. Trans. Oper. Res.*, **32** (2025), 745–768. https://doi.org/10.1111/itor.13458
43. Y. Liu, H. Xu, X. Yu, J. Zhou, Heuristic feeder-bus operation strategy considering weather information: a chance-constrained model, *Int. Trans. Oper. Res.*, **31** (2024), 2446–2471. https://doi.org/10.1111/itor.13211
44. S. Sun, Z. Duan, Q. Xu, School bus routing problem in the stochastic and time-dependent transportation network, *PLOS ONE*, **13** (2018), 1–17. https://doi.org/10.1371/journal.pone.0202618
45. Z. Wang, A. Haghani, Column generation-based stochastic school bell time and bus scheduling optimization, *Eur. J. Oper. Res.*, **286** (2020), 1087–1102. https://doi.org/10.1016/j.ejor.2020.03.071
46. J. Díaz-Ramírez, C. M. Leal-Garza, C. Gómez-Acosta, A smart school routing and scheduling problem for the new normalcy, *Comput. Ind. Eng.*, **166** (2022), 108101. https://doi.org/10.1016/j.cie.2022.108101
47. Z. Fu, J. Y. J. Chow, The pickup and delivery problem with synchronized en-route transfers for microtransit planning, *Transp. Res. Part E: Log. Transp. Rev.*, **157** (2022), 102562. https://doi.org/10.1016/j.tre.2021.102562
48. Y. Wang, L. Ran, X. Guan, J. Fan, Y. Sun, H. Wang, Collaborative multicenter vehicle routing problem with time windows and mixed deliveries and pickups, *Expert Syst. Appl.*, **197** (2022), 116690. https://doi.org/10.1016/j.eswa.2022.116690
49. S. Effendy, R. H. C. Yap, Real-time passenger bus routing problems with preferences and tradeoffs, *Ann. Math. Artif. Intell.*, **91** (2023), 287–307. https://doi.org/10.1007/s10472-022-09812-3
50. G. H. Wu, P. Pourhejazy, W. X. Li, T. H. Wu, A new dispatching mechanism for parallel-machine scheduling with different efficiencies and sequence-dependent setup times, *Decis. Anal. J.*, **10** (2024), 100432. https://doi.org/10.1016/j.dajour.2024.100432
51. C. Ferreira, G. Figueira, P. Amorim, Effective and interpretable dispatching rules for dynamic job shops via guided empirical learning, *Omega*, **111** (2022), 102643. https://doi.org/10.1016/j.omega.2022.102643

52. H. Xiong, H. Wang, S. Shi, K. Chen, Comparison study of dispatching rules and heuristics for online scheduling of single machine scheduling problem with predicted release time jobs, *Expert Syst. Appl.*, **243** (2024), 122752. https://doi.org/10.1016/j.eswa.2023.122752

53. D. Kasper, M. J. Land, R. H. Teunter, Towards system-state dispatching in high-variety manufacturing, *Omega*, **114** (2023), 102726. https://doi.org/10.1016/j.omega.2022.102726

54. M. Durasevic, F. J. Gil-Gala, D. Jakobovic, C. A. Coello Coello, Combining single-objective dispatching rules into multi-objective ensembles for the dynamic unrelated machines environment, *Swarm and Evol. Comput.*, **80** (2023), 101318. https://doi.org/10.1016/j.swevo.2023.101318

55. KIDSDAY, Kidsday [mobile app], Android App. Available from: `https://play.google.com/store/apps/details?id=com.kidsdayparents&hl=zh_TW`. Accessed: 2025-07-21.

56. OpenStreetMap contributors, OpenStreetMap map at zoom 7, center 23.611, 120.768. Available from: `https://www.openstreetmap.org/#map=7/23.611/120.768`, 2025. Accessed: 2025-08-10.

57. T. Brinkhoff, A framework for generating network-based moving objects, *GeoInformatica*, **6** (2002), 153–180. https://doi.org/10.1023/A:1015231126594


AIMS Press