information

Article

Study on Customized Shuttle Transit Mode Responding to Spatiotemporal Inhomogeneous Demand in Super-Peak

Hao Zheng, Xingchen Zhang and Junhua Chen *

School of Traffic and Transportation, Beiing Jiaotong University, No. 3 Shang Yuan Cun, Hai Dian District, Beijing 100044, China; 18251284@bjtu.edu.cn (H.Z.); xczhang@bjtu.edu.cn (X.Z.)
* Correspondence: cjh@bjtu.edu.cn; Tel.: +86-134-2602-5309

Abstract: Instantaneous mega-traffic flow has long been one of the major challenges in the management of mega-cities. It is difficult for the public transportation system to cope directly with transient mega-capacity flows, and the uneven spatiotemporal distribution of demand is the main cause. To this end, this paper proposed a customized shuttle bus transportation model based on the "boarding-transfer-alighting" framework, with the goal of minimizing operational costs and maximizing service quality to address mega-transit demand with uneven spatiotemporal distribution. The fleet application is constructed by a pickup and delivery problem with time window and transfer (PDPTWT) model, and a heuristic algorithm based on Tabu Search and ALNS is proposed to solve the large-scale computational problem. Numerical tests show that the proposed algorithm has the same accuracy as the commercial solution software, but has a higher speed. When the demand size is 10, the proposed algorithm can save 24,000 times of time. In addition, 6 reality-based cases are presented, and the results demonstrate that the designed option can save 9.93% of fleet cost, reduce 45.27% of vehicle waiting time, and 33.05% of passenger waiting time relative to other existing customized bus modes when encountering instantaneous passenger flows with time and space imbalance.

Keywords: shuttle transit service; pickup and delivery with time windows and transfers; super-peak; instantaneous demand

Citation: Zheng, H.; Zhang, X.; Chen, J. Study on Customized Shuttle Transit Mode Responding to Spatiotemporal Inhomogeneous Demand in Super-Peak. *Information* **2021**, *12*, 429. https://doi.org/10.3390/info12100429

Academic Editors: Nacima Labadie and Lyes Khoukhi

Received: 13 August 2021
Accepted: 25 September 2021
Published: 18 October 2021

1. Background

In megacities with complex travel demand structures, the transportation organization problems triggered by instantaneous peak passenger demand has troubled planners and operators for a long time. Despite the proved successfulness of fixed-route transit (FRT) in providing fixed and scheduled service, their performance is unsatisfying when dealing with irregular passenger flow with uneven spatiotemporal distribution. Their bad performance results from the stableness of service capacity which FRT provides, and the fact that it is difficult for FRT to do demand-responsive adjustments to operation lines. Thus, as the transportation capacity of FRT system always appears to be insufficient when coping with flow peaks, large, irregular passenger flow with uneven spatiotemporal distribution has always been the main cause of urban traffic problems under the current FRT system.

Under this background, Customized Shuttle Transit (CST) provides a new option for cities. CST is a user-oriented, demand-responsive, and individual customized transportation service, which generates operating routes and stops by collecting and calculating real-time requests. Thus, CST could operate efficiently in satisfying fluctuant and unpredictable requests, in dealing with peak flow, and providing services in the FRT-vacant period. In this way, the implementation of CST could alleviate peak traffic pressure, cover the unserviceable circumstances of the FRT, and provide an option to satisfy instantaneous peak passenger flow.

However, the distribution of requests submitted to CST system can be scattered, thus it might be difficult to integrate different customer demands into same bus routes. In this way, CST operators need to achieve balance between passenger satisfaction and operating cost.

This is always a difficult balance to reach: improving passengers' satisfaction always results in an increase in operational cost as well. Here is what always happens: in order to provide satisfactory services, the traveling time has to be shortened, as a result the number of passengers CST could serve per unit time will decrease, which will increase the average operating cost per customer, as operators has to spend more on vehicles and drivers, vice versa.

How to integrate transportation resources effectively to reach balance between service quality and operating cost has been a difficult problem. Transferring could help to alleviate this problem by splitting one service unit into one pick-up section and one delivery section and satisfy them separately, to avoid long-distance transregional travel of vehicles. However, new problems such as longer waiting time also emerges when transferring is introduced into the service mode. According to this, a new route planning system which could at the same time guarantee the service quality, keep the operational cost relatively low and provide adequate service capacity is in urgent need. To this end, this paper proposes a shuttle bus service that can gather and transport passengers efficiently. It can reduce operating costs and improve passenger satisfaction in facing the instantaneous mega-demand of spatiotemporal unevenness.

The contributions of this paper are as follows:

I. Aiming at solving the problems caused by instantaneous super-peak passenger demand, this paper proposed a shuttle transit mode that organize passenger flow by a boarding-transferring-alighting framework. This framework, which targets on minimizing total operating costs and maximizing passenger satisfaction, can be well adapted to the needs of spatiotemporal uneven passenger flows;
II. Considering time window, capacity, etc., constraints, a model based on PDPTWWT is proposed. Furthermore, a heuristic algorithm based on ALNS algorithm and Tabu-search algorithm is proposed to solve large scale problems. The proposed algorithm is proven outperform commercial solver;
III. The efficiency of the proposed method is verified by comparing the metrics with other models based on different spatiotemporal homogeneity cases on the background of the actual scale case of Beijing.

The rest of this paper is organized as follows. Section 2 surveyed the works related to model establishment and algorithm construction for related problems. Section 3 constructed a model for shuttle bus route planning for instantaneous large passenger flow. A heuristic solution combined tabu search algorithm with ALNS algorithm was developed in Section 4. A set of numerical experiments were given in Section 5 to verify the validity and the effectiveness of the approach. Section 6 concluded the paper and discussed the further directions.

2. Literature Reviews

The study of customized bus system was carried out 60 years ago, while no project has landed until the 1970s. In 1973, Operators in Wisconsin, America has launched "Merrill-Go-Round" as the very first customized bus system, which is still being used at present. After that, more customized bus systems were launched. In America, ADA (Americans with Disabilities Act) paratransit bus service emerged in accordance with the ADA campaign of 1990. Dial-a-ride Transit was launched by Hopelink, which can be reserved through phone call. Other successful examples are OmniLink service, Peninsula Transit and Hampton Roads Transit (HRT) in America, Kan-Go, Flexible Transport System and Pocket Ride in Australia, and Winnipeg Transit in Canada.

The model of CB route planning problem (CBRPP) is a special case of PDPTW (pickup and delivery problem with time window) or PDPTWT (pickup and delivery problem with time window and transfers). The PDPT has been intensively studied over recent years.

In model establishment, Masson et al. constructed three mathematical models for the PDPS (The Pickup and Delivery Problem with Shuttle routes). Mitrović-Minić [1] proved the usefulness of transshipment points in the context of pickup and delivery problems with time windows. Deleplanque [2] introduced transshipment into traditional dial-a-ride

problems (a variant of VRP problems) and solved it by an algorithm based on insertion techniques and constraints propagation [3].

In algorithm designs, exact algorithms are always used in cases. Sol (1994) discussed about general column generation techniques in solving pickup and delivery problems [4]. Masson et al. proposed a branch-and-price algorithm to solve the PDPS model they constructed [1]. Naccache et al. solved multi-PDPTWT firstly by a branch-and-bound algorithm, and then developed a hybrid adaptive large neighborhood search with improvement operations to solve large scale problems [5]. Parragh and Schmid modeled the underlying optimization problem of proposing specialized transportation systems to complement the traditional public transportation into a dial-a-ride problem, and proposed a hybrid column generation and large neighborhood search algorithm to solve it. Ropke et al. [6] introduced two new formulations to improve model performance for the PDPTW and dial-a-ride problem (DARP) problems, and developed branch-and-cut algorithms to solve instances with up to 8 vehicles and 96 requests [7]. However, the problem scales that exact algorithms can solve are less than 100 demands, and transferring are not allowed. Therefore, when facing large-scale problems, heuristic methods are always introduced.

Heuristic algorithms transcend in the ability of solving large-scale problems and are more often proposed to deal with more complicated matters. Among the researches in recent years, genetic algorithm (GA) [8–10], tabu search algorithm (TS) [11–14], ant colony optimization algorithm (ACO), simulated annealing algorithm (SA), Particle Swarm Optimization (PSO) [15–18], Adaptive large neighborhood search algorithm (ALNS) [5,6,19–21], and dynamic programming [22] are most frequently used.

GA has been studied by many scholars including Tan [8], Baker et al. [9] and Ombuki [10] on VRP and VRP-related problems. TS has been studied by Cordeau to solve dial-a-ride problem [11], by Joséto to solve open VRP problems in which vehicles do not have to go back to ending depot after delivering goods to customers [12], and by Lai [13] and Taillard [14] on PDPTW-related problems as well. ACO has been studied by Gambardella [15], Donati et al. [16], especially by Li to solve a multi-depot green vehicle routing problem (MDGVRP) [17] and by Zhang to solve the multi-objective vehicle routing problem [18]. SA has been studied by Bachem [23], Vincent [24], Chao [25], Chiang et al., and especially by Russell to study VRPTW [26]. PSO has been studied by Kachitvichyanukul to solve the GVRP-MDPDR (generalized multi-depot vehicle routing problem with multiple pickup and delivery requests) [27] and by Dridto to solve MDPDPTW (pickup and delivery problem with time windows and multiple depots) [28]. ALNS has been more frequently used in large scale PDPTW-related problems, which has been used by Ropke and Parragh [19], Ghilas [20], Côté [5], and Parragh and Schmid [6], and especially by Masson to solve PDPT (pickup and delivery problem with transfers) problems [21]. Dynamic programming has been studied by Ritzinger to solve VRP related problems as well [22].

The applicability of different heuristic algorithms in solving different PDPTW-related problems could be concluded from the researches above. GA's performance is satisfying when problem is less complicated and problem scale is relatively small, but when problem scale grows, the searching procedure prolongs rapidly. ACO has strong robustness, and performs well in multi-objective optimization of PDPTW-related problem, but is relatively easy to sink into local optimum when required iteration becomes more. SA performs very well in small scale PDPTW-related problems with the advantage of strong robustness in initial stage, but its performance also declines when the scale of problem gets larger. TS and ALNS, however, with the advantage of low computational complexity, have been proved to be efficient in solving large scale PDPTW-related problems, especially ALNS is always used as an alternative to rigorous algorithms when problem scale gets larger. In conclusion, when it comes to large scale complicated single objective optimization problem, researchers always prefer TS or th eALNS algorithm.

It could also be concluded that among the PDPTW-related problems, traditional PDPTW draws most of the attentions, while PDPTW with transfers is very rarely mentioned, which is mostly solve by adaptive large neighborhood search (ALNS). Due to the

spatiotemporal uncertainty and the short pickup and delivery time-windows, the transporting demand which generated by spatiotemporal homogeneity peak passenger flow is hard to satisfy. Thus, the service pattern of shuttle transit which is proposed to serve this demand might be different with the existing ones, with demand-responsive route planning functions, multiple flexible depots, and transferring allowed.

3. Customized Shuttle Transit for Instantaneous Large Passenger Flow Model Description (CSTILP)

To serve instantaneous large passenger flow, massive transportation resources is needed, and vehicles always need to travel through large acreage to satisfy every passenger. These make it difficult for transportation system to provide service for instantaneous large passenger flow. To get this problem solved, an operational system with transferring is needed, which could shrink proportion of transregional transportation by transferring and downsize the vehicle fleet.

The key point of the establishment of a transferring-allowed model is to avoid passengers or vehicles waiting long time at transfer point, and to guarantee the consistency of service after transferring has taken place.

3.1. Notations

3.1.1. Variables

We present the variables and the description of variables in Table 1.

Table 1. Names and Description of Variables.

Names of Variables	Description
x_{ij}^k	$\begin{cases} 1, \text{ if vehicle } k \text{ travels from location } i \text{ to location } j \\ 0, \text{otherwise} \end{cases}, (i,j) \in V$
$z_{n^\pm}^k$	$\begin{cases} 1, \text{ if the pickup section} (n^+) \text{ or delivery section } (n^-) \text{ is assigned to vehicle } k \\ 0, \text{otherwise} \end{cases}, k \in K, n \in N$
w_{tn}^{kl}	$\begin{cases} 1, \text{ if vehicle } k \text{ and } l \text{ transfered request } n \text{ at transfer point } t \\ 0, \text{otherwise} \end{cases}, k,l \in K, n \in N, t \in TS$
D_i^k	depicts the departure time of vehicle k, $i \in V, k \in K$
y_i^k	specifies the load of vehicle k when it arrives at location i, $i \in V, k \in K$

3.1.2. Sets

We present the sets and the description of sets in Table 2.

Table 2. Names and Description of Sets.

Names of Sets	Description
N	The set of service units, $N \triangleq N^+ \cup N^-$.
q_n	For each service unit n, there are q_n passengers need to be transported from position n^+ to position n^-, and $q_n = q_{n^+} = -q_{n^-}$.
N^+	$N^+ \triangleq \{n^+ \mid n \in N\}$ is defined as the set of pickup locations.
N^-	$N^- \triangleq \{n^- \mid n \in N\}$ as the set of delivery locations.
De	The set of depots, $De \triangleq De^+ \cup De^-$.
De^-	The set of starting depots, $De^- \triangleq \{de_1^-, de_2^-, \ldots\}$. Every vehicle has a fixed starting depot, vehicles in set M_α have p_α^- as their starting depot.
De^+	The set of ending depots, $De^+ \triangleq \{de_1^+, de_2^+, \ldots\}$. Number of vehicles returning to ending depot de_p^+ need to be bigger than the minimum number k_p.
De_k^+	The set of ending depots which vehicle k can go back to, $De_k^+ \triangleq \{de_k^+{}_1, de_k^+{}_2, \ldots\}$.
Ts	The set of transfer points, $Ts \triangleq \{t_1, t_2 \ldots t_n\}$.
V	The locations vehicles can visit, $V \triangleq N^+ \cup N^- \cup Ts \cup De^- \cup De^+$
V^N	$V^N \triangleq N^+ \cup N^-$ denotes the service locations.
V^{De+}	$V^{De+} \triangleq N^+ \cup N^- \cup Ts \cup De^+$
V^{De-}	$V^{De-} \triangleq N^+ \cup N^- \cup Ts \cup De^-$
V^{NoDe}	$V^{NoDe} \triangleq N^+ \cup N^- \cup Ts$
K	The vehicle set. $K \triangleq \{k_1, k_2 \ldots k_n\}$

3.1.3. Parameters

We present the parameters and the description of parameters in Table 3.

Table 3. Names and Description of Parameters.

Names of Parameters	Description
c_{ij}	Travel cos t of arc $(i,\ j)$
D_{ij}	Travel time of arc $(i,\ j)$
q_i	Number of passengers to be picked up or delivered
F_k	Fixed cos t of vehicle k
e_i	Earliest arrival time of request i
l_i	Latest arrival time of request i
Q_k	Maximum number of passengers vehicle k could load
pos_i^+	Pickup location of request i
pos_i^-	Delivery location of request i
op_i	Operation time of location i
k_p	Minimum number of vehicles to be arrived at depot p
t_{wp}	The maximum amount of time a passenger can wait at transfer points
t_{wc}	The maximum amount of time a vehicle can wait at transfer points
OP_i	The operation time at service point i
$tsop_t$	The operation time at transfer point t
d_{max}^k	The maximum operating distance of vehicle k
t_{max}^k	The maximum operating time of vehicle k
C_f	The total fixed cost
C_{op}	The total operating cost
C_{pwt}	Passenger waiting cost
C_{vwt}	Vehicle waiting cost
C_{qua}	Service quality penalty
$C_{pickup}^{unsatis}$	The unsatisfaction caused by the time gap between the expected pickup time and the actual ones
$C_{delivery}^{unsatis}$	The unsatisfaction caused by the time gap between expected delivery time and the actual ones
C_{dt}	The unsatisfaction caused by the difference between direct travel time and actual travel time
f_k	The fixed cost of vehicle k
c_p	The cost per unit time of passenger waiting
c_{wk}	The cost per unit of waiting time of vehicle k
W_j^k	The waiting time of vehicle k at point j

3.2. Model

We formally formulate the LIPPD as follows:
Objectives:

$$minf(x) = min\Big(C_f + C_{op} + C_{pwt} + C_{vwt} + C_{qua}\Big) \tag{1}$$

subject to

$$\sum_{j:(i,j)\in A} x_{ij}^k - \sum_{j:(j,i)\in A} x_{ji}^k = 0,\ k \in K, i \in V^{NoDe}, \tag{2}$$

$$\sum_{j:(i,j)\in A} x_{ij}^k - \sum_{j:(j,i)\in A} x_{ji}^k = z_n^k, k \in K,\ i \in V^{De^+}, n \in N, \tag{3}$$

$$\sum_{i:(i,j)\in A} x_{ij}^k - \sum_{i:(j,i)\in A} x_{ji}^k = z_n^k, k \in K,\ j \in V^{De^-},\ \forall n \in N, \tag{4}$$

$$\sum_{k\in M} z_{n^+}^k = \sum_{k\in M} z_{n^-}^k = 1,\ n \in N, \tag{5}$$

$$\sum_{j:(n^+,j)\in A} x_{n^+j}^k = z_{n^+}^k, k \in M,\ n \in N, \tag{6}$$

$$\sum_{j:(n^-,j)\in A} x_{n^-j}^k = z_{n^-}^k,\ k \in M, n \in N, \tag{7}$$

$$z_{n^-}^k + \sum_{t\in TS, l\in M} w_{tn}^{lk} - \sum_{t\in TS, l\in M} w_{tn}^{kl} = z_{n^+}^k, k \in K, n \in N, \tag{8}$$

$$\sum_{j\in V'} x_{de_k^- j}^k = 1,\ k \in K_p, de_k^- is\ the\ starting\ depot\ of\ k, \tag{9}$$

$$\sum_{p^+\in De^+} \sum_{i\in V'} x_{ip^+}^k = 1, k \in K, \tag{10}$$

$$\sum_{i\in V',k\in M} x_{ide^+}^k \geq k_{de}, de^+ \in De^+, \tag{11}$$

$$\sum_{i\in V} x_{it}^k \leq 1, k \in M, t \in TS, \tag{12}$$

$$\sum_{i\in V} x_{it}^k - \sum_{l\in M} w_{tn}^{kl} \geq 0, k \in K, t \in TS, n \in N, \tag{13}$$

$$\sum_{i\in V} x_{it}^l - \sum_{k\in M} w_{tn}^{kl} \geq 0, l \in K, t \in TS, n \in N, \tag{14}$$

$$\sum_{i\in V} x_{in}^l - \sum_{k\in M} w_{tn}^{kl} \geq 0, l \in K, t \in TS, n \in N, \tag{15}$$

$$\sum_{i\in V} x_{in}^k - \sum_{l\in M} w_{tn}^{kl} \geq 0, k \in K, t \in TS, n \in N, \tag{16}$$

$$D_i^k + t_{ij} - D_j^k + op_j \leq M\cdot\left(1 - x_{ij}^k\right),\ i \in V^N\ ,\ j \in V, k \in K, \tag{17}$$

$$D_i^k + t_{ide_k^+}^k - \sum_{i\in V'}\left(x_{ide^+}^k \cdot l_{de^+}\right) + op_{de_k^+} \leq M\cdot\left(1 - x_{ide_k^+}^k\right), i \in V^{NoDe}, k \in K, de^+ \in De^+, de_k^+ \in De_k^+, \tag{18}$$

$$D_t^k + t_{wp} + tsop_t - D_t^l \geq M\cdot\left(\sum_{n\in N} w_{tn}^{kl} - 1\right), t \in TS,\ k, l \in K, \tag{19}$$

$$D_t^k - t_{wc} + tsop_t - D_t^l \leq M\cdot\left(1 - \sum_{n\in N} w_{tn}^{kl}\right), t \in TS, k, l \in K, \tag{20}$$

$$D_{n^+}^k - D_t^k \leq M\cdot\left(1 - \sum\nolimits_{l\in M} w_{tn}^{kl}\right), n \in N, t \in TS, k \in K, \tag{21}$$

$$D_t^l - D_{n^-}^l \leq M\cdot\left(1 - \sum\nolimits_{k\in M} w_{tn}^{kl}\right), n \in N, t \in TS, l \in K, \tag{22}$$

$$D_{n+}^k - D_{n-}^l \leq M\cdot\left(1 - \sum\nolimits_{t\in TS} w_{tn}^{kl}\right),\ k, l \in K, n \in N, \tag{23}$$

$$D_{n+}^k - D_{n-}^k \leq M\cdot\sum\nolimits_{t\in TS, l\in M} w_{tn}^{kl},\ k \in M, n \in N, \tag{24}$$

$$y_j^k - y_i^k - q_j \leq M\cdot\left(1 - x_{ij}^k\right), i \in V_N, j \in V, \tag{25}$$

$$y_j^k - y_i^k - q_j \geq M\cdot\left(1 - x_{ij}^k\right), i \in V_N, j \in V, \tag{26}$$

$$y_t^k - y_i^k + \sum_{l\in M, n\in N}\left(w_{tn}^{kl} * q_n\right) - \sum_{l\in M, n\in N}\left(w_{tn}^{lk} * q_n\right) \leq M\cdot\left(1 - x_{it}^k\right), i \in V, k \in K, t \in TS, \tag{27}$$

$$y_t^k - y_i^k + \sum_{l\in M, n\in N}\left(w_{tn}^{kl} * q_n\right) - \sum_{l\in M, n\in N}\left(w_{tn}^{lk} * q_n\right) \geq M\cdot\left(x_{it}^k - 1\right), i \in V, k \in K, t \in TS, \tag{28}$$

$$\sum_{i\in V', j\in V'} x_{ij}^k d_{ij} \leq d_{max}^k, k \in M \tag{29}$$

$$\sum_{i\in V', j\in V'} x_{ij}^k t_{ij}^k \leq t_{max}^k, k \in M, \tag{30}$$

$$y_{de^-}^k = 0, de^- \in De^-, \tag{31}$$

$$D_{de^-}^k = 0, de^- \in De^-, \tag{32}$$

$$e_i \leq D_i^k \leq l_i, i \in V^N, \tag{33}$$

$$0 \leq y_i^k \leq Q_k, k \in K, i \in V, \tag{34}$$

$$x_{ij}^k \in \{0, 1\}, i, j \in V, \tag{35}$$

$$z_n^k \in \{0, 1\}, n \in N, \tag{36}$$

$$w_{tn}^{kl} \in \{0, 1\}, n \in N. \tag{37}$$

Objective function (1) is composed of C_f (fixed cost), C_{op} (operating cost), C_{pwt} (passenger waiting cost), C_{vwt} (vehicle waiting cost), and C_{qua} (service quality cost).

(1) C_f (fixed cost)

The fixed cost refers to the cost of fixed equipment, including sales personnel salary, cost of vehicle maintenance, enterprise management, and vehicle depreciation. The calculation of total fixed cost is as shown in the formula:

$$C_f = \sum_{j \in V^+} \sum_{k \in M} f_k x^k_{p^- j}. \tag{38}$$

(2) C_{op} (operating cost)

The operating cost is related to the total operating time and other factors. The total operating time cost is calculated as shown in the formula:

$$C_{op} = \sum_{i \in V} \sum_{j \in V} \sum_{k \in M} x^k_{ij} c_{ij}. \tag{39}$$

(3) C_{pwt} (passenger waiting cost)

Passenger waiting cost refers to the waiting time cost caused by time window limitations when passengers are waiting for follow-up passengers to get on the vehicle, which focus on the perspective of passengers. The total cost of passenger waiting time is calculated as follows:

$$C_{pwt} = \sum_{k \in K} \sum_{i \in V'} \sum_{j \in V'} c_p x^k_{ij} y^k_i W^k_j. \tag{40}$$

(4) C_{vwt} (vehicle waiting cost)

Vehicle waiting time cost refers to the time cost caused by time window limitations, which focus on the perspective of vehicles. The total cost of vehicle waiting time is calculated as shown in the formula:

$$C_{pwt} = \sum_{k \in K} \sum_{i \in V} cw_k W^k_i. \tag{41}$$

(5) Service quality penalty

$$C_{qua} = C^{unsatis}_{pickup} + C^{unsatis}_{delivery} + C_{dt} \tag{42}$$

In order to measure the service quality, we define three quality indexes. Research shows that most passengers prefer the vehicle to arrive at the mid-point of the pickup time window and the low limit of the delivery time window [29]. $C^{unsatis}_{pickup}$ indicates the time gap between the expected pickup time and the actual ones, $C^{unsatis}_{delivery}$ indicates the time gap between the expected delivery time and the actual ones. C_{dt} is the difference between direct travel time and actual travel time which caused by necessary detouring distance.

$$C^{unsatis}_{pickup} = \sum_{i \in N} \sum_{k \in K} \left| \frac{e_{i^+} + l_{i^+}}{2} - D^k_i \right| \cdot \sum_{i \in V} x^k_{p^- i}, \tag{43}$$

$$C^{unsatis}_{delivery} = \sum_{i \in N} \sum_{k \in K} \left| e_{i^-} - D^k_{i^-} \right| \cdot \sum_{i \in V} x^k_{p^- i}, \tag{44}$$

$$C_{dt} = \sum_{i \in N} \sum_{k \in K} \left| t^{\,p}_{serve_i} - t_{i,i+n} \right| \cdot \sum_{i \in V} x^k_{p^- i}. \tag{45}$$

Equations (2)–(4) is the equilibrium constraint. Equation (5) ensures each section of every service unit is assigned to exactly one vehicle. Equations (6) and (7) ensures only if the vehicle passes certain section of the service unit, can the demand be served. Equation (8) ensures the continuity of service with transfers. Equation (9) ensures the starting point of every vehicle is corresponded to the starting station it belongs to. Equation (10) ensures each vehicle only goes back to one of the returning stations. Equation (11) ensures the total number of vehicles returned to every station is higher than the minimum value. Equation (12) ensures vehicles will not visit a same transfer point repeatedly in one service procedure. Equations (13)–(16) ensures the condition for the occurrence of transferring: only if vehicle k and vehicle l passes transfer point t in a same service procedure, can they exchange passengers in transfer point t, and only if demand n has been picked up by vehicle k, can it be transferred from vehicle k to vehicle l. Equations (17) and (18) defines

the arrival time of each node. Equation (19) ensures the transfer waiting time of passengers is shorter than twp. Equation (20) ensures the transfer waiting time of vehicles is shorter than twc. Equations (21) and (22) ensures if $w_{ti}^{kl} = 1$, $D_{i^+}^k < D_t^k$ and $D_t^l < D_{i^-}^k$, which defined the time sequence of the procedure of transferring. Equations (23) and (24) furtherly define the sequence of arrival. Equations (25) and (26) is car loading constraint which define how vehicle pickup and deliver customers when without passing transfer points. Equations (27) and (28) define how vehicle pickup and deliver customers when it comes to transfer. Equations (29) and (30) define the maximum operating time and maximum travelling distance of each vehicle.

3.3. Operational Mechanism

A five-step process which describes the mechanism of CSTILP system is designed, as shown in Figure 1.

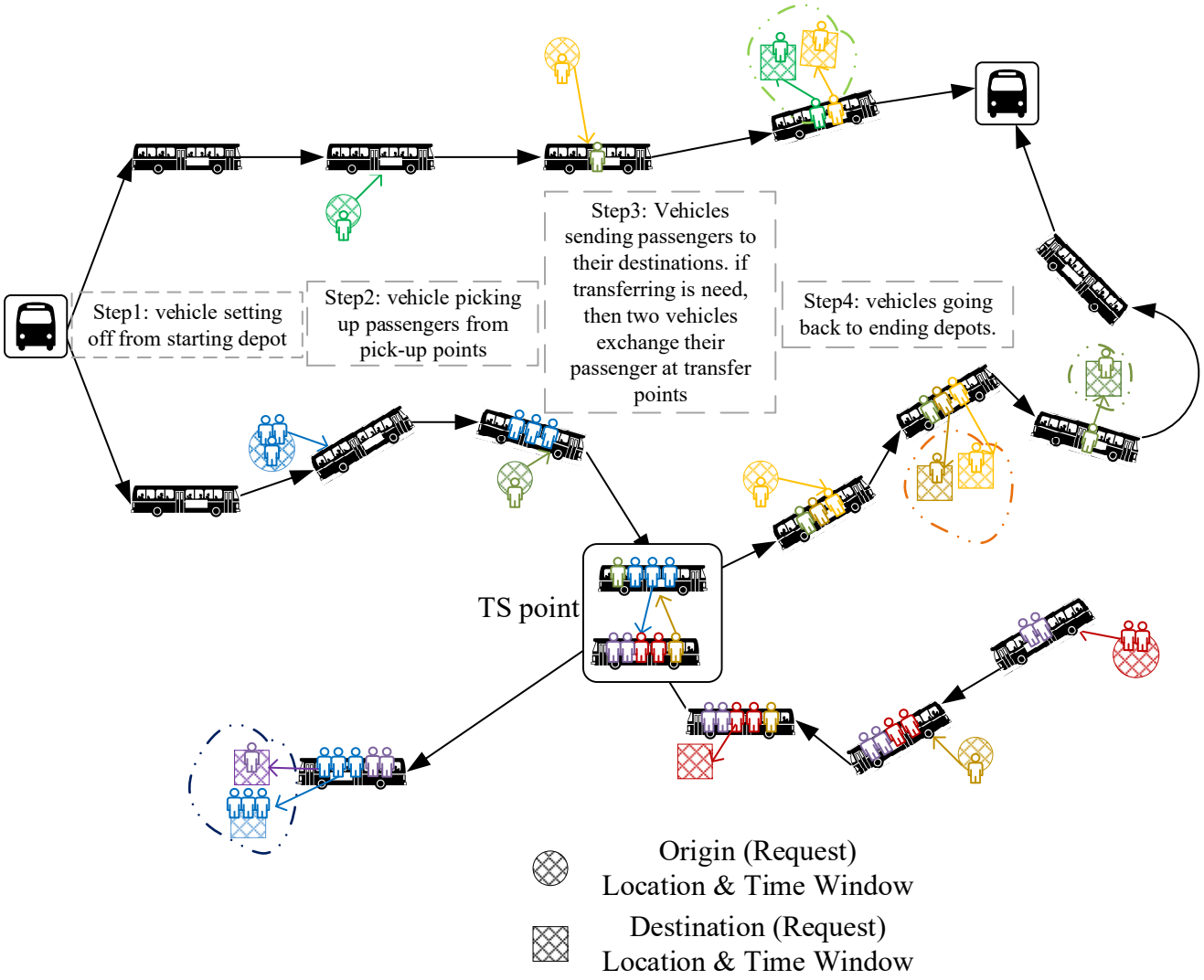

Figure 1. Illustration of the CSTILP design model. This figure demonstrates the main optimization content of the proposed model. There are four types of nodes in this figure, where the cycles represent origins of requests, the squares represent destination of requests, the rounded squares with car icon represents depots, and the bigger rounded square represents the transfer points. The request includes information about location and time window. The figure shows the designed operational process of CSTILP design model.

Step 0: Request reservations are submitted before the operational process of vehicles starts. The information each reservation provides include the number of passengers, the pickup and delivery locations and the starting and ending time windows.

Step 1: Vehicles depart from their assigned starting depot vacantly, described by constraint (31).

Step 2: Vehicles pick up passengers. This could happen by passing the pickup location and pick passengers up directly which is described by constraints (6)~(7), or by accepting passengers from other vehicles at transfer locations, which was described by constraint (8).

Step 3: At the same time, vehicles deliver passengers into their destinations or transfer them to other vehicles. In the end, every picked-up passenger will be sent to their destinations, described by constraint (5).

Step 4: Vehicles go back to ending depots. The ending depot for each vehicle is not fixed, vehicles are assigned to ending depots according to the minimum returning number of vehicles for each depot, which is described by constraint (10).

4. ALNS-TS Algorithm

As accurate algorithm performs poorly when solving large-scale instances, introducing heuristic algorithm (hybrid ALNS-TS) is necessary.

Different from the standard ALNS-TS, hybrid ALNS-TS has the following characteristics:

(1) Destroying and repairing only two routes at a time. This is used to improve the performance of transfer operator. Transferring happens between only two routes (which constructed by two vehicles), thus it is easier for transferring to happen if the optimization process takes place on only two vehicles at a time, where the destroy operator and transfer operator are able to work corporately during whole optimization procedure. The advantage of this is that it allows pick-up and delivery sequence to be lined up in a way which could take place only if transferring is allowed. For example, if 10 passengers all boarded from the same place, and the destination of eight of them is the same (let us call it D_1), while the destination of the remaining two passengers is D_2, which is very far from D_1 that makes it impossible for one vehicle to deliver passengers to their coordinate destinations without violating time windows. In this way, if the repair operator and the transfer operator did not work corporately in the optimization procedure, it would be very likely for the algorithm to assign another vehicle to serve this passenger, or refuse to provide service. In this way, second-best solutions without transferring would very likely be generated. Although the optimization process can further improve the solution, this unneeded "detouring" process reduce the efficiency of this algorithm by making the happening of transfers harder. Moreover, as transfer operator only work with the second half of repairing operator which inserts only delivery points instead of pick-up points, this second-best solution has to wait for another iteration to be fixed, which is likely not to happen due to the randomness of this algorithm.

However, by optimizing only two routes at a time, repairing operator, destroy operator and transfer operator could work corporately at the same time, which provides the maximum advantageous conditions for transferring to happen. For the previous example (10 passengers, the destination of 8 of them was D_1, the destination of the rest two was D_2), if transfer operator was able to work together with repair operator in the optimization process, all the 10 passengers will be allowed to be picked up by a same vehicle. Then, this vehicle could transfer the two passengers with D_2 destination to other vehicles, maybe to vehicles which has passengers whose destinations have similar geographical location with the transferred ones. Considering that transferring is very helpful in serving instantaneous large passenger flows, we think it is necessary to put the happening of transferring in priority.

(2) Tabu list is introduced. The traditional ALNS is characterized by large amount of neighborhood solutions with low similarities, and it is unlikely for same neighborhood solutions be reached in different iterations. Thus, the risk of being sank in local optimum is of low possibility, thus seldom to be considered. However, as the improved ALNS

algorithm shrinks the search domain, the risk of falling into local optimum cannot be neglected. Thus, the technique of tabu search is introduced in the optimization procedure.

(3) 2-opt destroy operators are introduced to enhance the possibility for the length of vehicle routes to change in the optimization process, in order to improve the performance of ALNS-TS algorithm.

4.1. Main Scheme of Hybrid ALNS-TS

4.1.1. Generating Initial Solution

The Clarke and Wright savings algorithm (C-W algorithm) was implemented to generate initial solution [30]. The initial solution was obtained by inserting pickup and delivery points into routes, and accepting the insertion option with maximum savings. The initial solution is feasible and without any transfer point.

4.1.2. ALNS-TS Algorithm

The fundamental optimization technique of ALNS-TS is to start from initial solution and search its adjacent domain for satisfactory solutions, where ALNS operators (op_i^+, op_i^-), TS (tabu) operators, and Transfer operator (Trs) work in collaboration. ALNS operators can be characterized into destroy (op_i^-) and repair (op_i^+) operators, which work together to obtain adjacent solution by destroying and repairing the current solution iteratively in order to improve it. Trs works together with repair operator in order to add transfer points in route. TS operator delimits the searching procedure to avoid trapping into local optimization, rejecting moves to points already visited in the search space.

Based on the proposed ALNS-TS, the framework is depicted as follows.

Generating IS by C-W algorithm
$TS\_ALNS\_Algorithm(IS,\ N)$
  $BS = IS,\ CS = IS$
$NeigCS'$= *Null*
  while the termination criterion is not satisfied
    Repeat:
      Select destroy operator op_i^- and repair operator op_i^- according to probabilities p_i^-, p_i^+
      $CS'' = (op_i^-(CS))$,
      $CS'' = (op_i^+, Trs(CS''))$,
      Add CS' to NeigCS'
    Until: Termination criterion is satisfied
    Repeat:
      if CS_j' is in tabu list
        if $f\left(CS_j'\right) < f(BS),\ BS = CS_j',\ CS^{new} = CS_j'$
      else
        if $f\left(CS_j'\right) < f(BNS),\ BNS = CS_j'$
      end
    Until: all neighborhood solutions have been searched.
    if $f(BNS) < f(BS),\ BS = BNS, CS^{new} = BNS$
     $CS = CS^{new}$
    Update p_i^+, p_i^- according to adaptive Mechanism
    Update tabu list
  end while
return $BS, CS,\ p_i^+, p_i^-$
end

Where CS: Current-solution, CS': Neighborhood-solution, $NeigCS'$:Set of Neighborhood solutions, IS: Initial-solution, BS: Best-solution, BNS: Best-solution in this neighborhood.

4.2. ALNS Components

ALNS iteratively destroys and repairs current solution in order to improve it. In this way, the components of a complete ALNS algorithm can be categorized into destroy operators (op_i^-) which remove request from current solution and repair operators (op_i^+) which

reinsert request back. In each iteration of improving current solution, there are $NeiNum$ neighborhood solutions need to be generated. To generate each neighborhood solution, only one destroy operator and one repair operator is chosen and used to optimize the selected route.

To shrink the computational labor and avoid trapping into local optimization, we introduced modified version of standard ALNS operators and Tabu (TS) operator into this algorithm. Different from the standard version of ALNS operators, the modified version generates neighborhood solution by optimizing two routes at a time instead of optimizing all the given routes.

4.2.1. Destroy Operators

The destroy operator set $\left(OP^- \triangleq \{op_1^-, op_2^-, \ldots, op_5^-\}\right)$ contains five operators, each of them belongs to one of these two subsets: OP_{swap}^- and OP_{2-opt}^-,$OP^- = OP_{swap}^- \cup OP_{2-opt}^-$. One of the destroy operators in OP is selected each time when generating one neighboring solution. The possibility of op_i^- to be selected is determined by the corresponding possibility p_i^-.

(1) Swap Operator

The general working procedure of swap operator is as follows.

Procedure of removal (CS, op_i^-) {
    Randomly choose two routes : R_a and R_b.
    $R_a''' = R_a$, $R_b''' = R_b$, $CS''' = CS$
    Calculating the temporary variables.
    Selecting the requests to be removed according to the removing rule of op_i^- and the corresponding temporary variables.
    Remove the chosen request from R_a''' and R_b'''
    Define the request set removed from R_a''' as Set_a^r, and which removed from R_b''' as Set_b^r.
    Return CS''' , R_a''', R_b''', Set_a^r, Set_b^r
}

The general framework of each operator is the same, the difference is displayed in the procedure of generating temporary variables and the rules they follow when removing requests. The corresponding rule to be followed of each operator is as follows.

I. Random Removal

Randomly remove p_r% of the requests from route a and b.

II. Worst Removal

The cost-savings by the removal of each request in route a and b are computed, then p_w% of the requests are selected to be removed according to the probability which increases with the cost-savings of their removal.

III. Related Removal

This heuristic operates basing on the belief that "similar requests are more likely to be exchanged between routes and might lead to better solution." We define the relatedness between request i and request j according to their attributes: location, starting time, end time, quantity of their customers, etc. According to the paper written by Ropke and Pisinger [19], We define the relatedness as

$$R_L(i,j) = \varphi d_{ij} + \zeta \left| D_{ik} - D_{jl} \right| + \kappa \left(1 - \frac{\left| K_i \cap K_j \right|}{min\{ |K_i|, |K_j| \}} \right) \tag{46}$$

The operating process of this operator is as follows:

- Randomly chooses the first request to be removed, named n_{a_i}.
- Calculate the relatedness between n_{a_i} and every request in route b.
- List them in descending order according to the relatedness measure to form a list L.
- Choose requests to be removed. For each request n_0, its possibility of being chosen is $\frac{R_L(i,n_0)}{\sum_{n \in N_l} R_L(i,n)}$.

(2) 2-opt operator

The general working procedure of 2-opt operator is as follows.

Destroying (CS, op_i^-) {
Randomly choose two routes : R_a and R_b.
$R_a''' = R_a$, $R_b''' = R_b, CS''' = CS$
Select the chosen section $R_{a[i,j]}'''$ and $R_{b[k,l]}'''$ to be removed according to the corresponding rules of each operator. $R_{a[i,j]}'''$ is the section between position i and position j in R_a''', $i, j < length(R_a'''))$, section R_b is the section between position k and position l in R_b''', $k, l < length(R_b'''))$
Extract $R_{a[i,j]}'''$ from R_a''' and $R_{b[k,l]}'''$ from R_b'''.
Define the request set removed from R_a''' as Set_a^r, and which removed from R_b''' as Set_b^r.
Return$CS''', R_a''', R_b''', Set_a^r, Set_b^r, R_{a[i,j]}''', R_{b[k,l]}''', i, j, k, l$
}

The general framework of each operator is the same, the difference between them is displayed in the procedure of the rule of choosing sections to be removed. The corresponding rule to be followed of each operator is as follows.

I. Random Removal

Randomly pick one piece of section from each route.

II. Best Removal

Generate two random numbers (r, t). Calculate the average cost saving of route R_a if a section with a length of r is deleted iteratively until all possibilities have been calculated. Calculate the average cost saving of route R_b if a section with a length of t is deleted iteratively until all possibilities have been calculated. Choose the option with maximum saving and delete the chosen section from the original chains.

4.2.2. Repair Operator

The repair operator set $\left(OP^+ \triangleq \{op_1^+, op_2^+, op_3^+\}\right)$ contains three operators. One of the repair operators in OP^+ is selected each time when generating one neighboring solution. The possibility of op_i^+ to be selected is determined by the corresponding possibility p_i^+.

The general working procedure of repair operator op_k^+ is as follows.

Repairing (Route R_a, R_b, unoccupied request set UR, Set_a^r, Set_b^r) {
Define $UR_B = UR \cup Set_a^r$, $UR_A = UR \cup Set_b^r$
for R_a, R_b
do
pick one request $R$from set $UR_A(R_B)$following Rul_k
do
look for a position where R can be inserted in $R_a(R_b)$ according to the repairing rule of op_i^+
if at least one feasible solution exists
accept this insertion and renew $R_a(R_b)$.
$UR_A = UR_A - R$, $UR_B = UR_B - R$.
break
else
continue
until both chains have been processed.
if there is not any feasible solution
continue
until all unoccupied requests have been processed.
set$CS'(R_a) = R_a'$, $CS'(R_b) = R_b'$
set CS = set CS'
$UR = UR_A \cup UR_B$
return set CS, UR
}

The general framework of each operator is the same, the difference is displayed in the procedure of the rule of choosing and inserting requests. The corresponding rule to be followed of each operator is as follows.

I. Random Insertion

Random insertion heuristic is a serial computational process. At each iteration, unoccupied requests are randomly selected and randomly inserted in random positions of selected vehicle chain.

II. Greedy Insertion

Define the cost increase after the insertion of request i into route r as

$$c_{i,r}^{add} = c_r^{ori} - c_{i,r}^{ins}$$

Define C_g as the minimum cost of the insertion of request i:

$$c_g = min_{r \in CS} c_{i,r}^{add}$$

where g is the best inserting position.

This operator works by iteratively assigns requests to the batch according to best inserting position g. Note that this operator also ensures removed requests not being inserted into the original route.

III. Regret-2 insertion

The operating process of this operator is as follows:

- Define α_k^i as the number k best insertion option for request i.
- Define β_2^i as the difference between the optimal inserting solution and the suboptimal inserting solution. $\beta_2^i = c_{\alpha_2^i} - c_{\alpha_1^i}$
- Choose the operating request i_{op} by the following expression:

$$i_{op} = \arg max_{i \in N} \beta_2^i$$

- Iteratively assigns requests by descending order to the batch according to best inserting position. Note that this operator also ensures removed requests not being inserted into the original route.

4.2.3. Transfer (*Trs*) Operator

In the procedure of generating neighboring solutions, this is how op_i^+ and op_i^- work after they are chosen:

$$CS \xrightarrow[\text{destorying current solution}]{op_i^-, \text{Trs operator}} CS''' \xrightarrow[\text{inserting pickup points}]{op_i^+(1)} CS'' \xrightarrow[\text{inserting delivery points}]{op_i^+(2), \text{Trs operator}} CS',$$

where op_i^- deletes both the pickup section and the delivery section from vehicle routes. *Ts*(tabu) operator determines whether and which transfer point will be removed. $op_i^+(1)$ inserts the pickup section into the current solution. *Trs* operator works together with $op_i^+(2)$ to generate solutions with different numbers of transfer points. In this way, after the process of repairing, two neighboring solutions with exactly same sequence of pickup points are generated, one with transfer points and one without, and only the better ones will be accepted.

Define the maximum number of transfer points in one route as n_t^m. In this paper, the value of n_t^m equals to 1.

The pseudocode of how this procedure works is as follows.

Generating qth CS′($op_i^-, op_i^+(1), op_i^-(2)$, *Trs operator*, R_a, R_b) {
——————————— DELETIING TS POINTS ———————————
  if (number of Ts points in route $CS(R_x'')$ equals to or is larger than n_t^m,):
    Define the route which has transferring relationship with R_x as: R_x^{corres}
    Replace R_x with $\text{Set}_{Solution}^{NoTs}(x)$ and R_x^{corres} with $\text{Set}_{Solution}^{NoTs}(x^{corres})$
  end
  optimizing R_x: $R_x \xrightarrow[destorying\ neighborhood]{op_i^-,\ Trs\ operator} R_x''' \xrightarrow[inserting\ pickup\ points]{op_i^+(1)} R_x''$
——————————— GENERATING TSBESTSOLUTION ———————————
  Repeat:
    Repeat:
      judge whether transfer point t can be inserted in $R_x(i)$.
      if insertion(i, t, R_x) is feasible
        $add = [i,\ t,\ EarliestArrivalTime,\ LastestLeavingTime]$,
        $SetTs(R_x)$= $SetTs(R_x)$+add
      end
    Until all Transfer points have been checked
  Until all possible positions in route R_x have been checked
  Determine where can transfer point be inserted in order to construct transferring relationship between R_a and R_b.
  Describe the inserting option as:
  $FeasibleTs$=$[i, j,\ t,\ EarliestArrivalTime,\ LastestLeavingTime]$, where i denote the inserting position of transfer point t in R_a, j denote the inserting position of transfer point t in R_b. $SetTs = SetTs + FeasibleTs$
  Repeat:
    Determine which of the request in each route can be transferred by their time windows. Develop sets Set_a^{ts} and Set_b^{ts}. List all the combinations of request from two groups in set $TsOptions$.
    Repeat:
      Judge whether this option is feasible. If it is, replace the original $R_a, R_a^{corres}, R_b, R_b^{corres}$ with the new ones. calculate the total cost.
    Until all options of transferring in $TsOptions$ have been calculated.
  Until all options of inserting transfer point in $SetTs$ have been calculated.
  Pick the ones with minimum cost as $BestTsSolution$
——————————— GENERATING NOTSSOLUTION ———————————
  Use the traversal methods which were discussed in 4.2.2 to repair R_a and R_b.
  Replace the original R_a and R_b in CS with R_a' and R_b'. Name the new solution as $NoTsSolution$ and calculate the total cost.
  $\text{Set}_{Solution}^{NoTs}(q) = NoTsSolution$.
  if $\text{Cost}_{NoTsSolution} > \text{Cost}_{BestTsSolution}$:
    Neig$CS' =$ Neig$CS' + NoTsSolution$.
  else
    Neig$CS' =$ Neig$CS' + BestTsSolution$.
end
}

Where, $\text{Set}_{Solution}^{NoTs}$:a set of solutions with no transfer points which was generated through the optimizing process. $\text{Set}_{Solution}^{NoTs}$ is restored for usage, routes X in this set will be used to replace the current route when corresponding route X in current solution with transfers is being operated in another iteration. NST: $NoTsSolution$.

4.3. Adaptive Mechanism

Adaptive mechanism records the performance α and usage count θ of each destroy or repair operator and adjust their possibilities to be chosen according to their performance in past iterations.

The detailed mechanism is as follows.

Before the searching process starts, all destroy or repair operators are set to the same initial weight ω. The ALNS searching process is divided into several segments, each segment contains λ iterations. After λ iterations, weights are updated according to the

quality of solutions the corresponding operator reached, which are measured by four scenarios $(\alpha_1 > \alpha_2, \alpha_3)$.

In every iteration:

1. $\beta_i = \beta_i + \alpha_1$, if a new best solution is reached by op_i in this iteration.
2. $\beta_i = \beta_i + \alpha_2$, if the solution op_i reached in this iteration is better than the current solution.
3. $\beta_i = \beta_i + \alpha_3$, if the solution op_i reached in this iteration is worse than the current solution but is accepted.
4. $\beta_i = \beta_i$, else.

After λ iterations, renew corresponding weight of each operator:

$$\omega_i = \begin{cases} \omega_i(1-r) + r\frac{\beta_i}{\theta_i}, & \theta_i > 0 \\ \omega_i(1-r), & \theta_i = 0 \end{cases}. \tag{47}$$

For all the destroy or repair operator to be chosen, their possibility to be chosen is $\frac{\omega_i}{\sum_i \omega_i}$.

5. Numerical Experiment

In the numerical experiment, we proved the validity of the proposed ALNS-TS algorithm and the CSTILP model by comparing the performance of the ALNS-TS algorithm with that of GAMS (A commonly used commercial solver), based on a set of cases with incremental scale. Besides, the comparison can further prove efficiency of the proposed algorithm on relatively large-scale cases.

After that, we evaluated the performance of ALNS-TS algorithm on six different cases derived from realistic demands. These six cases differ in their temporal and spatial homogeneity and can be used to verify the adaptation of the proposed method in different scenarios.

5.1. Validation of the ALNS-TS Method

We developed instances with 4–10 service units which were solved by both GAMS and proposed ALNS-TS algorithm. The solving result GAMS presented were regarded as the standard solution, and the feasibility and efficiency of ALNS-TS algorithm is proven by comparing the solutions and operation time of these two kinds of solutions.

Figure 2 presents a comparison of the GAMS solutions and ALNS-TS solutions from the perspectives of the operation time. The vehicle routing solutions and operation time are also shown in Table 4, which shows that ALNS-TS algorithm obtains the same results as GAMS does; thus, the proposed method is feasible.

To prove the effectiveness of this algorithm, we discuss the operation time of these two methods. As the number of service units increases, the calculation times for GAMS grows exponentially, as is shown in Figure 2. When there are four service units, the calculation time is 4.55 s, but when the number grows from 4 to 5, the operation time increases by about 48 times to 168.20 s. The operation time reaches 294,506.23 s (approximately 82 h) when the number of service units reaches 10, which is regarded as the limit, that is at least 24,000 times that of the time cost by ALNS-TS algorithm.

Although the calculation time of ALNS-TS generally grows when the scale of service units gets larger, it shows a decrease when the number of service unit grows from 4 to 5, then shows an increase when the number of service unit reaches 7. This results from the fact that when number of service units decreases, it becomes harder for ALNS-TS to reach new feasible solutions, in this way the algorithm spends more time in searching for better solutions.

These observations demonstrate the effectiveness of the proposed method.

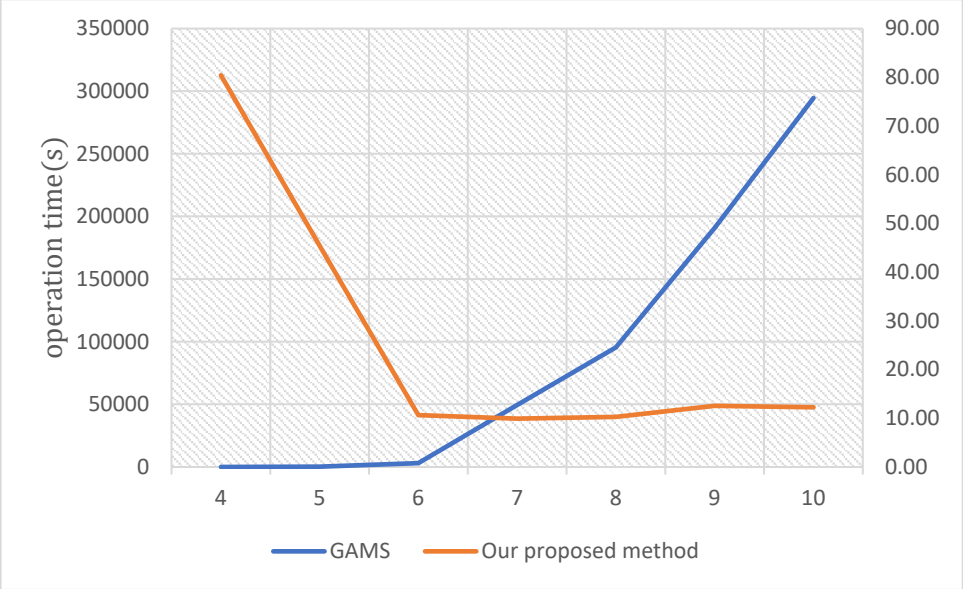

Figure 2. Comparison of the operation time of two methods.

Table 4. Comparison of the operation time of two methods.

Case ID	Number of Service Units	Operating Time of Solving Method/Sec		Vehicle Routing Solution
		GAMS	Our Proposed Method	
1	4	4.55	80.42	$route_1$: $de_1-n_2^+-n_2^--n_1^+-n_1^--n_3^+-n_4^+-n_3^--n_4^--de_1$
2	5	168.2	45.42	$route_1$: $de_1-n_1^+-n_1^--n_2^+-n_5^+-n_5^--n_2^--n_4^+-n_3^+-n_4^--n_3^--de_1$
3	6	3027.8	10.64	$route_1$: $de_1-n_2^+-n_2^--n_5^+-n_5^--n_3^+-n_6^+-n_6^--n_3^--n_4^+-n_4^--n_1^+-n_1^--de_1$
4	7	49,402.26	9.91	$route_1$: $de_1-n_4^+-n_3^+-n_4^--n_5^+-n_5^--ts_1-n_1^--n_5^--n_2^+-n_2^--de_1$ $route_2$: $de_1-n_1^+-n_7^+-n_6^+-ts_1-n_6^--n_7^--de_1$
5	8	95,373.58	10.27	$route_1$: $de_1^--n_5^+-n_4^+-n_5^--n_6^+-n_4^--n_8^+-n_7^+-n_8^--n_7^--n_1^+-n_3^+-ts_1-n_6^--de_1^+$ $route_2$: $de_1^--n_2^+-n_2^--ts_1-n_1^--n_3^--de_1^+$
6	9	194,795.16	12.54	$route_1$: $de_1^--n_2^+-n_3^+-n_2^--n_7^+-n_8^+-n_3^--n_8^--n_7^--n_1^+-n_5^+-ts_1-n_1^--n_6^--de_1^+$ $route_2$: $de_1^--n_9^+-n_9^--n_4^+-n_4^--n_6^+-ts_1-n_5^--de_1^+$
7	10	294,506.23	12.25	$route_1$: $de_1-n_1^+-n_1^--n_7^+-n_2^+-n_8^+-n_6^+-n_5^+-n_5^--n_7^--n_6^--n_2^--n_9^+-n_{10}^+-n_4^+-n_9^--n_{10}^--n_3^+-n_4^--n_3^--de_1$

5.2. ALNS Computational Experiments

The method was coded in MATLAB and experiments were run on an E74830-CPU computer operated by Windows Server 2016. We discussed ALNS settings and used data based on real-life to evaluate the benefits of routing with transfer points.

5.2.1. Cases Design

To discuss the applicability of this algorithm, we developed six cases based on Beijing's layout to carry out experimentation. As train stations and airports are important passenger distributing centers, the passenger flow of the instances used in this paper mainly arose from sources of stations, airports and residential area.

We developed six cases which were grouped into three case pairs with different spatiotemporal distribution characteristics to examine the spatiotemporal applicability of this algorithm. Besides, one of the two cases in a same case pair is with transfer option, the other is without transfer option, in order to examine whether improvement will be made and how much improvements will be achieved after transferring are implemented in operation.

The flow-generation points include: Beijingxi Railway Station, Beijingbei Railway Station, Beijingnan Railway Station, Beijingchaoyang Railway Station, Beijing Railway Station, airports: Beijing Daxing International Airport, Beijing Capital International Airport, and 13 different kinds of trip generation centers in residentials areas, known as housing estates attachments, other housing estates, community centers, dormitories, residential

estates, universities, hotel, resort, apartment hotel, economy hotel, hotel affiliations, guest house, other hotels.

A request n is composed of OD (origin-destination) pair with time windows. In the first four instances, we took airports or stations as request origin or destination, and took trip generation centers in residential areas as the other end of OD pair, to generate the spatial distribution of requests. After that, we used real-time timetable of airports and train stations to get the OD time windows of airports/stations end, and used distances between origin and destination to generate time windows of the other end. We divided passenger's demand into 34 categories, including travelling from different airports or stations to residential area, travelling from residential area to different airports or stations, and travelling between different airports and stations. The proportion of each kind of request was decided by the actual number of passengers arrive with flights and trains. In the fifth and sixth instance, we used trip generation centers in residential areas to generate both origin and destination, whose one end of time windows were the same as the first and second instance, the other end was generated based on the travelling distance divided by traveling speed.

In the first two instances, the temporal distribution of request was comparatively even, which was based on real-time timetable from 23:00 to 7:00 the next day. However, the spatial distribution of which was centralized, where one end of the tour was airports or train stations, and the other end was randomly picked in residential area. In the second instance, transferring was allowed, while in first transferring was not allowed. The spatial distribution of the third and fourth instances was the same as the first and the second one, but in temporal distribution, service requests were distributed more densely from 5:00 to 7:00 to form super-peak. In the fourth instance, transferring was allowed, while in the third transferring was not allowed. The spatiotemporal distribution of both the fifth and sixth instance were scattered, with time windows generated randomly from 23:00 to 7:00 the next day, and requests picked up randomly from Beijing's residential area. In the sixth instance, transferring was allowed, while in the fifth transferring was not allowed.

The settings of these six instances are shown in Table 5. Vehicles with different capacity and different fixed cost are assigned to each depot before the operational process starts.

Table 5. Instance Settings.

Case Index	Spatial Distribution	Temporal Distribution	Spatiotemporal Distribution	Transferring Allowed or Not	Number of Requests
1	Centralized	Scattered, from 23:00 to 7:00	Centralized	Not Allowed	488
2	Centralized	Scattered, from 23:00 to 7:00	Centralized	Allowed	488
3	Centralized	Centralized, from 5:00 to 7:00	Super-centralized	Not Allowed	538
4	Centralized	Centralized, from 5:00 to 7:00	Super-centralized	Allowed	540
5	Scattered	Scattered, from 23:00 to 7:00	Scattered	Not Allowed	396
6	Scattered	Scattered, from 23:00 to 7:00	Scattered	Allowed	396

5.2.2. Parameter Settings

For most of the parameters of the ALNS algorithm, we validate the values proposed by Ropke and Pisinger [19]. The parameter description and settings are shown in Table 6.

5.3. Research Results

The operational results of these six instances are shown in Table 7. The improvements in quality index of different case pairs after transferring was introduced are shown in Table 8. Figure 3 visualized the improvements in quality index after transferring is introduced by a bar chart.

It could be concluded from the experimental result that fixed cost reduced by 4.51% and the total cost reduced by 6.84% when transferring became allowed in cases with centralized passenger flow, and the number was 9.90% and 9.93% in super-centralized ones, while in cases with scattered passenger flow the fixed cost did not change and total cost reduced by 4.11%. This shows transferring can to a large extent reduce the operating cost of CB system, especially when the spatiotemporal distribution of passengers is centralized. Comparing the case with transfer option and the case without transfer option in all three

case pairs, the travelling time of one passenger on average reduced after transferring is allowed, where the case with centralized passenger flow reduced by 7.84%, the case with super-centralized flow reduced by 9.95%, which was very satisfying, and the one with scattered passenger flow reduced by 5.80%. Both passenger waiting time and vehicle waiting time reduced when transferring became allowed in all three instance pairs, but the improvement appeared to be more significant only when the concentration degree got relatively high. In centralized cases numbers of passenger waiting time and vehicle waiting time were 0.05% and 18.69%, which were even less than the number in scattered case, that were 15.72% and 36.94%. Improvement in passenger waiting time and vehicle waiting time reached 33.05% and 45.27% when concentration ratio continued to increase. The number of vehicles in operation reduced by 9.40% after transferring became allowed in super-centralized flow case and reduced by 5.66% in centralized flow case, whereas in scattered ones the number stayed still. This might because when the requests are scattered, even if transferring could integrate some requests in other routes, it might be difficult to put all of the requests especially the ones with remote origins or destinations into other routes as well. Table 9 shows the details of the optimized solutions.

Table 6. Parameters in algorithm.

Notation	Description	Value
ω_i	Initial weight of operator i	0.1
σ_1	Operator evaluation parameter 1	33
σ_2	Operator evaluation parameter 2	9
σ_3	Operator evaluation parameter 3	13
r	Operator weight adjustment parameter	0.1
λ	Length of iteration fragments	100
$iteration$	Max iteration	2000
φ	Similarity parameter 1	9
ζ	Similarity parameter 2	3
κ	Similarity parameter 3	2
t_{wp}	Passenger waiting time	5 min
t_{wc}	Vehicle waiting time	3 min
op_i	Operation time at point i	2 min
$tsop_t$	Operation time at transfer point t	3 min
d^k_{max}	Maximum travelling distance	300 km
t^k_{max}	Maximum travelling time	480 min

Table 7. Operational Results.

	1	2	3	4	5	6
Total cost	81,418.72	76,209.29	121,587.18	111,011.54	77,595.96	74,534.88
Vehicles in use	168	159	326	298	153	153
Passengers served	488	488	538	540	396	396
c_{fx}	12,040	11,520	23,520	21,480	10,860	10,860
c^{per}_{total}	166.84	156.17	226.00	205.58	195.95	188.22
c^{per}_{fx}	24.67	23.61	43.72	39.78	27.42	27.42
t^{per}_{tra}(min)	117.50	108.95	138.56	126.02	141.10	133.37
t^{per}_{pw}(min)	31.01 ± 25.11	31.00 ± 26.44	39.36 ± 27.34	29.58 ± 23.87	40.16 ± 29.92	34.71 ± 27.52
t^{per}_{vw}(min)	126.78 ± 82.69	106.82 ± 74.60	71.49 ± 44.29	49.21 ± 47.81	109.66 ± 62.58	80.08 ± 60.77
t^{gap}_{up}(min)	16.58 ± 25.80	15.25 ± 21.78	12.36 ± 11.33	11.82 ± 10.92	9.78 ± 8.49	10.33 ± 10.35
t^{gap}_{down}(min)	17.60 ± 16.51	14.09 ± 13.28	24.10 ± 16.64	14.88 ± 16.76	25.17 ± 19.52	18.16 ± 16.00
t^{per}_{dt}(min)	4.34 ± 15.31	7.18 ± 19.69	0.86 ± 5.08	7.06 ± 17.83	0.51 ± 1.32	7.10 ± 21.00

where: c_{fx}: total fixed cost, c^{per}_{total}: total cost per passenger on average, c^{per}_{fx}: fixed cost per passenger on average, t^{gap}_{up}: difference between the expected picking up time and the actual picking up time of one passenger on average, t^{gap}_{down}: difference between the expected delivery time and the actual delivery time of one passenger on average, t^{per}_{tra}: Travelling time of one passenger on average, t^{per}_{pw}: Waiting time of one passenger on average, t^{per}_{vw}: Waiting time of one vehicle on average, t^{per}_{dt}: detouring time of one passenger on average.

Table 8. Improvements in quality index after transferring is introduced.

Index	Case 1 vs. Case 2	Case 3 vs. Case 4	Case 5 vs. Case 6
Vehicle in operation	5.66%	9.40%	0.00%
c_{total}^{per}	6.84%	9.93%	4.11%
c_{fx}^{per}	4.51%	9.90%	0.00%
t_{trg}^{per}	7.84%	9.95%	5.80%
t_{pw}^{per}	0.05%	33.05%	15.72%
t_{vw}^{per}	18.69%	45.27%	36.94%

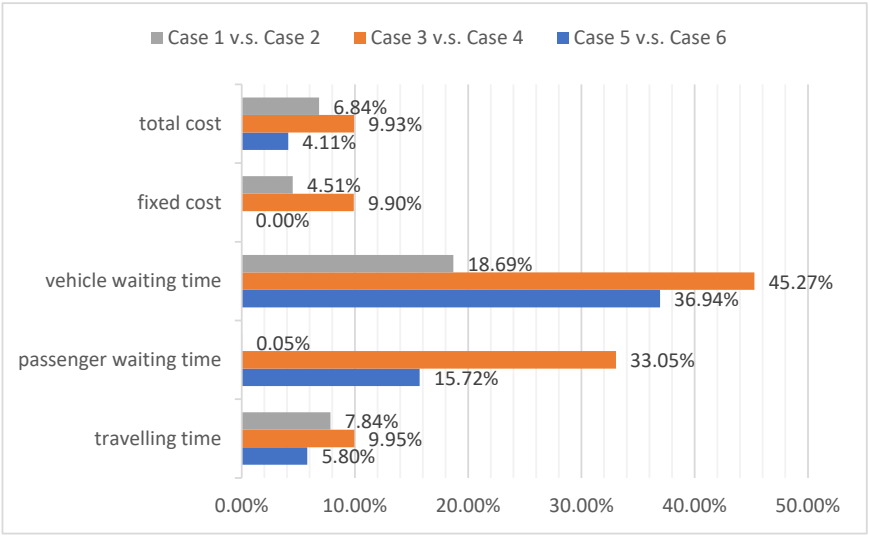

Figure 3. Improvements in indexes after transferring is introduced.

Table 9. Details of the optimized solutions.

ID	Vehicle Routing Solutions	Travel Distance (km)
1	$route\ 1:\ de_1, 330^+, 330^-, 363^+, 363^-, 380^+, 521^+, 380^-,\ 521^-, de_1$	136.10
	$route\ 2:\ de_1, 389^+, 168^+, 168^-, 407^+, 407^-, 389^-, 440^+, 440^-, de_1$	103.31
	...	...
	$route\ 168:\ de_3, 338^+, 338^-, 272^+, 272^-, 299^+, 8^+, 299^-, 8^-, de_3$	139.30
2	$route\ 1: de_1, 330^+, 373^+, 330^-, 380^+, 373^-, 380^-, 697^+, ts_2, 650^-, 697^-, de_1$	179.36
	$route\ 2: de_1, 389^+, 392^+, 392^-, 407^+, 407^-, 389^-, ts_2, 445^-, 522^+, 522^-, de_1$	102.94
	...	...
	$route\ 159:\ de_3, 338^+, 338^-, 296^+, 296^-, 352^+, 439^+, 439^-, 352^-, 554^+, ts_2, de_3$	116.18
3	$route\ 1: de_1, 300^+, 300^-, 303^+, 303^-, de_1$	33.93
	$route\ 2: de_1, 293^+, 293^-, 38^+, 38^-, de_1$	75.61
	...	...
	$route\ 298: de_2, 61^+, 285^+, 285^-, 61^-, de_2$	64.49
4	$route\ 1: de_1, 300^+, 300^-, ts_3, 212^-, 234^+, 234^-,\ de_1$	83.26
	$route\ 2: de_1, 293^+, 293^-, ts_2, 295^-, 124^+, 124^-,\ de_1$	124.94
	...	...
	$route\ 326: de_2, 285^+, 285^-, ts_1, 249^-, 149^+, 149^-,\ de_2$	135.67
5	$route\ 1:\ de_1, 431^+, 432^+, 432^-, 461^+, 431^-, 461^-, de_1$	91.63
	$route\ 2: de_2, 388^+\ , 388^-, 421^+, 421^-, 481^+, 481^-, de_2$	115.18
	...	...
	$route\ 153:\ de_3, 306^+, 306^-, 381^+, 401^+, 381^-, 401^-, 497^+, 497^-, de_3$	151.15
6	$route\ 1:\ de_1, 431^+, 144^+, 144^-, ts_1, 432^-, 461^+, 431^-, 461^-, de_1$	104.28
	$route\ 2:\ de_2, 388^+, 388^-, 1^+, 481^+, ts_3, 481^-, de_2$	149.30
	...	...
	$route\ 153:\ de_3, 306^+, 306^-, 381^+, 401^+, 381^-, 401^-, 185^+, ts_3, 497^-, 185^-, de_3$	197.87

It could be concluded from the comparison of these three case pairs that transferring could enhance the performance of ALNS-TS when spatiotemporal distribution of requests is centralized. It could increase the indexes which indicate passengers' satisfaction and reduce the operating cost at the same time. However, the enhancement brought by transferring is not significant in when dealing with instances with scattered spatiotemporal distribution of requests.

6. Conclusions

Aiming on making customized bus an option in dealing with instantaneous passenger flow, this paper proposed a CSTILP model to satisfy the traffic demands of these passengers. Modes such as multi-depots, multi-vehicle types and transferring are introduced in this model, which help reach a satisfying balance between passenger satisfaction and operating cost.

When the scale of problem grows, the combination explosion problem emerges, which limits the solving capacity of commercial solvers to a relatively small scale. In order to provide feasible solutions of large-scale problems, this paper proposed a matching algorithm which combines traditional ALNS and TS problem into a hybrid one according to the uniqueness of this problem.

In order to prove the feasibility and efficiency of the algorithm, seven small-scale instances and six large-scale ones were proposed. We proved the feasibility of the algorithm by comparing the one solved by GAMS and by our proposed method, and after that we ran six large-scale experiments.

Experiments show: (1) the proposed algorithm has the same accuracy as that of GAMS, a commercial solution software, but has a higher speed: when the demand size is set as 10, the proposed algorithm can save 24,000 times of time; (2) by comparing results of six reality-based cases, we found that the designed option could save 9.93% times of fleet cost and reduce 33.05% times of passenger waiting time relative to other existing customized bus modes when encountering instantaneous passenger flows with time and space imbalance.

Then we can conclude that (1) in dealing with cases of instantaneous super-peak passenger demand, comparing our proposed method with which without transferring, ours can at the same time shrink the operational cost and improve the service quality index from the perspective of both passengers and drivers; (2) our proposed method specializes in satisfying instantaneous passenger flow, it is proved to have better performance in cases where there exists group of people making transportation requests to similar geographical area; (3) our proposed algorithm could not only to a large extent shrink the operational time, but also reach relative optimal solution.

In this study, we only considered providing service for instantaneous large passenger flow with centralized spatiotemporal distribution, and took customized shuttle bus as the only transport means. Therefore, we will seek to expand the scope of application in future research, and integrate more means of transportation in our operational mode by cooperating our service mode with other public transports.

Author Contributions: Conceptualization, H.Z.; data curation, J.C.; formal analysis, J.C.; funding acquisition, J.C.; supervision, X.Z.; writing—original draft, H.Z.; writing—review & editing, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the National Science Foundation of China (XZ) (Grant No. U1734204), the Science and Technology Department of the China Railway Corporation (XZ) (Grant No. N2020x022) and Fundamental Research Funds for the Central Universities (Grant No. 2020JBZD006).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The study did not report any data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Masson, R.; Ropke, S.; Lehuédé, F.; Péton, O. A branch-and-cut-and-price approach for the pickup and delivery problem with shuttle routes. *Eur. J. Oper. Res.* **2014**, *236*, 849–862. [CrossRef]
2. Mitrović-Minić, S.; Laporte, G. The pickup and delivery problem with time windows and transshipment. *INFOR* **2006**, *44*, 217–227. [CrossRef]
3. Deleplanque, S.; Quilliot, A. Dial-a-ride problem with time windows, transshipments, and dynamic transfer points. *IFAC Proc.* **2013**, *46*, 1256–1261. [CrossRef]
4. Sol, M. Column generation techniques for pickup and delivery problems. Ph.D. Dissertation, Eindhoven University of Technology, Eindhoven, The Netherlands, 8 November 1994.
5. Naccache, S.; Côté, J.; Coelho, L.C. The multi-pickup and delivery problem with time windows. *Eur. J. Oper. Res.* **2018**, *296*, 353–362. [CrossRef]
6. Parragh, S.N.; Schmid, V. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Comput. Oper. Res.* **2013**, *40*, 490–497. [CrossRef] [PubMed]
7. Ropke, S.; Cordeau, J.F.; Laporte, G. Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks* **2010**, *49*, 258–272. [CrossRef]
8. Tan, K.C.; Lee, T.H.; Ou, K.; Lee, L.H. A messy genetic algorithm for the vehicle routing problem with time window constraints. In Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, Korea, 27–30 May 2001; pp. 679–686.
9. Wester; Dawn, V. A Genetic Algorithm for the Vehicle Routing Problem. Available online: https://trace.tennessee.edu/cgi/viewcontent.cgi?article=6358&context=utk_gradthes (accessed on 24 September 2021).
10. Ombuki, B.; Ross, B.J.; Hanshar, F. Multi-objective genetic algorithms for vehicle routing problem with time windows. *Appl. Intell.* **2006**, *24*, 17–30. [CrossRef]
11. Cordeau, J.F.; Laporte, G. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transp. Res. Part B Methodol.* **2003**, *37*, 579–594. [CrossRef]
12. Brandão, J. A tabu search algorithm for the open vehicle routing problem-sciencedirect. *Eur. J. Oper. Res.* **2004**, *157*, 552–564. [CrossRef]
13. Lai, D.S.; Demirag, O.C.; Leung, J.M. A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. *Transp. Res. E Logist. Transp. Rev.* **2016**, *86*, 32–52. [CrossRef]
14. Taillard, P.E. A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transp. Res. Part C Emerg. Technol.* **1997**, *5*, 109–122.
15. Gambardella, L.M.; Dorigo, M. Solving symmetric and asymmetric tsp by ant colonies. In Proceedings of the IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 20–22 May 1996.
16. Donati, A.V.; Montemanni, R.; Casagrande, N.; Rizzoli, A.E.; Gambardella, L.M. Time dependent vehicle routing problem with a multi ant colony system. *Eur. J. Oper. Res.* **2008**, *185*, 1174–1191. [CrossRef]
17. Li, Y.; Soleimani, H.; Zohal, M. An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives. *J. Cleaner Prod.* **2019**, *227*, 1161–1172. [CrossRef]
18. Zhang, H.; Zhang, Q.; Ma, L.; Zhang, Z.; Liu, Y. A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows. *Inf. Sci.* **2019**, *490*, 166–190. [CrossRef]
19. Ropke, S.; Pisinger, D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transp. Sci.* **2006**, *40*, 455–472. [CrossRef]
20. Ghilas, V.; Demir, E.; Woensel, T.V. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Comput. Oper. Res.* **2016**, 12–30. [CrossRef]
21. Masson, R.; Lehuede, F.; Peton, O. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transp. Sci.* **2013**, *47*, 344–355. [CrossRef]
22. Ritzinger, U.; Puchinger, J.; Hartl, R.F. Dynamic programming based metaheuristics for the dial-a-ride problem. *Ann. Oper. Res.* **2016**, *236*, 341–358. [CrossRef]
23. Bachem, A.; Malich, M. The simulated trading heuristic for solving vehicle routing problems. *Discrete Appl. Math.* **1996**, *65*, 47–72. [CrossRef]
24. Yu, V.F.; Lin, S.W.; Lee, W.; Ting, C.J. A simulated annealing heuristic for the capacitated location routing problem. *Comput. Ind. Eng.* **2010**, *58*, 288–299. [CrossRef]
25. Chao, W.; Dong, M.; Fu, Z.; Sutherland, J.W. A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. *Comput. Ind. Eng.* **2015**, *83*, 111–122.
26. Chiang, W.C.; Russell, R.A. Simulated annealing metaheuristics for the vehicle routing problem with time windows. *Ann. Oper. Res.* **1996**, *63*, 3–27. [CrossRef]
27. Kachitvichyanukul, V.; Sombuntham, P.; Kunnapapdeelert, S. Two solution representations for solving multi-depot vehicle routing problem with multiple pickup and delivery requests via pso. *Comput. Ind. Eng.* **2015**, *89*, 125–136. [CrossRef]

28. Dridi, I.H.A.; Alaa, E.B.; Borne, P.; Bouchriha, H. Optimisation of the multi-depots pick-up and delivery problems with time windows and multi-vehicles using pso algorithm. *Int. J. Prod. Res.* **2020**, *58*, 4201–4214. [CrossRef]
29. Paquette, J.; Cordeau, J.F.; Laporte, G.; Pascoal, M.M. Combining multicriteria analysis and tabu search for dial-a-ride problems. *Transp. Res. B Meth.* **2013**, *52*, 1–16.
30. Clarke, G.; Wright, J.W. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* **1964**, *12*, 568–581. [CrossRef]