An Advanced and Adaptive Tabu Search Algorithm for Dynamic Shared Parking Reservation and Allocation

Shangbin NING[1,2], Zhenzhou YUAN[1,2,*], Zhenyu HAN[1], Yang YANG[3,4]

[1] School of Traffic and Transportation, Beijing Jiaotong University, Beijing 100044, P.R. China
zzyuan@bjtu.edu.cn (**Corresponding author*)

[2] Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport, Ministry of Transport, Beijing Jiaotong University, Beijing 100044, P.R. China
17114215@bjtu.edu.cn

[3] School of Transportation Science and Engineering, Beihang University, Beijing 100191, P.R. China

[4] Beijing Key Laboratory for Cooperative Vehicle Infrastructure Systems and Safety Control, Beihang University, Beijing 100191, P.R. China
yangphd@buaa.edu.cn

Abstract: Inefficient utilization of existing parking resources is the main cause of parking difficulties, especially in metropolises. Shared parking allocation based on the online parking reservation system (PRS) is an effective way to deal with the inefficient utilization since it increases available parking resources through sharing and avoids blind search through reservation and allocation. This study focuses on the dynamic shared parking allocation problem based on PRS. A meta-heuristic algorithm, namely Advanced and Adaptive Tabu Search (AATS), was designed to cope with the real-time updates of parking demands and shared parking space and to achieve good allocation effect and high allocation speed. The proposed algorithm is based on advanced initialization with multi-factor sequencing and on adaptive neighborhood generation with bi-operator competition. The results of a three-day district-level experiment show that AATS achieves an allocation effect close to the exact algorithm, while having a significant superiority in allocation speed. The proposed AATS is practical to deal with realistic parking problems.

Keywords: Resource allocation problem, Allocation optimization, Rolling horizon strategy, Meta-heuristic algorithm.

1. Introduction

Parking has become a big challenge in cities, especially in metropolises (Krpan, Marsanic & Milkovic, 2017; Yang et al., 2022a). The gap between the increase in the parking demand and the limited parking resources along with the inefficient use of these resources are responsible for the parking difficulties. Under these conditions, it has been confirmed that blind search for available parking spaces inevitably causes a waste of time. For example, research found that a driver may spend 3.5 to 14 minutes finding an on-street parking space (Hampshire & Shoup, 2018) and that looking for a parking space is accountable for about 8% to 74% of traffic congestion (Hampshire & Shoup, 2018; Yang et al., 2022b).

Except for constructing more parking resources, how to efficiently utilize existing parking resources is the key to solving parking problem. Therefore, researchers proposed the parking reservation systems (PRS) (Mei et al., 2020). Drivers can submit parking requests (e.g., the destinations, target occupancy time windows, even the parking preferences) to PRS and the basic idea of PRS is to guarantee drivers available parking spaces prior to arrival through online reservation. Theoretically, compared with parking without reservation, parking with reservation can avoid the futile and lengthy search for available parking spaces and traffic congestion, especially in the unfamiliar surroundings or during peak parking hours. In addition, parking reservation can guide drivers to carefully plan their trips before travelling, such as changing target arrival time or parking preferences according to the difficulty of reservation.

Among researches on PRS, dynamic parking allocation problem (DPAP) is significant. In DPAP, drivers who use PRS (called users below) submit demands dynamically, i.e., demand input to PRS is continuously updated. In existing studies, in order to optimize the allocation effect, static parking allocation problem (SPAP) was firstly modeled as resource allocation problem, which was then optimized to maximize the system managers' profit or minimize the users' cost. Then, DPAP was modeled based on Rolling Horizon (RH) strategy, in which the time was discretized into a sequence of equal-length intervals and SPAP was solved at the end time point of each interval. Geng & Cassandras (2013) firstly formulated the allocation model as a linear programming model to minimize users' cost based on queuing theory and conduct static allocation at each decision time point defined in a time-driven sequence. Once being successfully allocated, users would be reallocated at each subsequent decision time point until they reached destination zones. Mladenovic

https://doi.org/10.24846/v31i3y202209

et al. (2020) firstly rationalized the effectiveness of solving DPAP based on rolling horizon strategy and then proposed a 0-1 programming model with the objective of minimizing users' cost and a variable neighborhood search-based heuristic to compute approximate solutions for lager instances. Mladenović et al. (2021) proposed a four-layer RH framework to tackle the real-time updates of parking demands and parking spaces. During each time interval, users that hadn't arrived at their parking were reallocated. In these studies, users in each allocation shared the same priority and were allocated group by group Theoretically, RH based allocation approach is capable of improving allocation effect, ensuring allocation speed, and handling many unpredictable circumstances (e.g., a driver decides to change the destination) when solving DPAP. The advantages are significant, especially in cities with high parking demands.

Besides improving the utilization of public parking resources, the development of sharing economy gives birth to parking sharing, a novel parking management method that improves the utilization of private parking resources by sharing the available parking spaces at different times during a day. Different from DPAP, the availability of parking spaces in the dynamic shared parking allocation problem (DSPAP) depends not only on the real-time occupancy of users, but also on the real-time sharing of providers. Among the research studies on DSPAP based on the PRS, some focused on the sharing aspect and improved truthful auction mechanisms (Xu et al., 2016), other focused on the allocating aspect and regarded it as the DPAP, but most of them focused only on improving static allocation models. Shao et al. (2016) considered the shared use of residential parking spaces between residents and public users and assumed that time windows of parking demands and shared parking spaces were specific to PRS. Then, the authors proposed a simple binary linear programming model with the objective of maximizing parking lot utilization under the time constraints. On this basis, Ning et al. (2020) and Ning et al. (2022) enriched the demand preferences and proposed a binary linear programming model with the objective of maximizing system managers' profit.

This paper studies the dynamic shared parking allocation problem based on PRS, in which multiple destinations and parking facilities, the temporal and spatial heterogeneity of parking demands and shared parking spaces, effective integration of allocation and reallocation are considered.

The contribution of this study is the creative design of a heuristic algorithm, namely, Advanced and Adaptive Tabu Search, to achieve high allocation speed meanwhile maintaining allocation effect in the large-scale shared parking allocation problem. In AATS, multi-factor sequencing was introduced to advance the initialization, in which the demands are multi-factor sequenced so as to generate high quality initial solution. Moreover, an exchange operator, a replacement operator as well as a novel scoring mechanism were designed to adaptively generate high-quality neighborhood solutions.

The rest of this paper is organized as follows. Section 2 describes the dynamic shared parking allocation problem, Doubly Periodic Rolling Horizon allocation strategy, and mathematical programming models; Section 3 presents the meta-heuristic algorithm Advanced and Adaptive Tabu Search for solving allocation models; Section 4 offers comparison between TS, AATS and CPLEX Solver in a three-day district-level experiment. The paper ends with concluding remarks in Section 5.

2. Dynamic Shared Parking Allocation

This section describes in detail the DSPAP, a Doubly Periodic Rolling Horizon (DPRH) allocation strategy and the allocation models.

2.1 Problem Description

A scenario of dynamic shared parking allocation is studied as follows: multiple parking facilities are distributed in one area, which are the sources of shared parking spaces and the destinations of users. The set $S_f \{1,2,\ldots,F\}$ is used to represent these parking facilities. At any time of the day, users submit parking demands to PRS for parking guarantee, while providers submit shared parking spaces to PRS for extra income.

Each parking demand $i \in S_i \{1,2,\ldots,I\}$ contains the following information: submission time t_i^P, time window $\left[t_i^S, t_i^E\right]$, destination location $\left[long_i, lat_i\right]$, parking preferences including the upper bounds of acceptable walking distance d_i^{max}, of acceptable parking price p_i^{max} and of acceptable waiting time w_i^{max}.

Among the submitted parking spaces, some are short-time shared (e.g., nine-to-five sharing) while some are long-time shared (e.g., 24-hour sharing).

Each shared parking space $j \in S_j \{1,2,\ldots,J\}$ contains the following information: time window $\left[t_j^S, t_j^E\right]$, parking space location $\left[long_j, lat_j\right]$, parking price P_j, short-time rental price r_j, or long-time rental price $r_j^{'}$.

The overall time horizon is divided into a set of time units $S_t \{1,2,\ldots,T\}$ with equal length τ. Each time unit corresponds to its end time point so that S_t is also a set of time points. In each time unit, PRS collects real-time submitted parking demands and parking spaces. At specific time point t, PRS conducts static narrow allocation.

In addition to the unallocated demands $S_i^U(t)$, the allocated but unoccupied demands $S_i^A(t)$, the occupied demands $S_i^O(t)$, the terminated demands $S_i^T(t)$ and the failed demands $S_i^F(t)$ exist simultaneously in PRS. Figure 1 describes the state transition of demands. PRS must ensure that allocated but unoccupied demands are successfully allocated in each broad allocation.

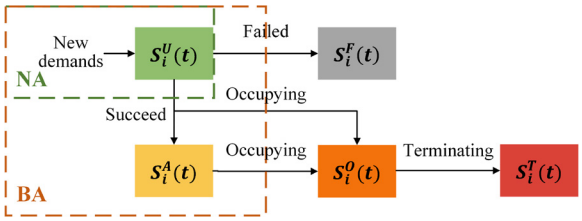

Figure 1. State transition of parking demands
Source: Figure 1 is adapted based on the previous work of Ning et al. (2022)

After allocation, the successful provider must ensure that the parking space is available in the shared time window according to the submitted information, and users are required to use the parking spaces according to the submitted target arrival time and departure time. Providers whose parking spaces are failed to be allocated will not be affected, and their parking spaces are still available in the rest of the shared time windows. Users whose demands are failed to be allocated have to wait for the subsequent allocation. If the waiting time exceeds the upper bound of acceptable waiting time, users will exit PRS and park by themselves.

2.2 Doubly Periodic Rolling Horizon Allocation Strategy

Denote $S_t \left\{T^1, T^2, \ldots, T^n\right\}$ as the set of corresponding time points. In a time unit, PRS collects parking spaces and parking demands. Denote τ_N as narrow allocation period, $\tau_N = m_N \cdot \tau$ and the narrow allocation time points set is $S_t^N \left\{T_N^1, T_N^2, \ldots, T_N^n\right\}$. Denote τ_B as broad allocation period, $\tau_B = m_B \cdot \tau$ and the broad allocation time points set is $S_t^B \left\{T_B^1, T_B^2, \ldots, T_B^n\right\}$.

When $m_N \neq 0$, $m_B \neq 0$ and $m_B > m_N$, the strategy is a Doubly Periodic Rolling Horizon allocation strategy, $S_t^N \cup S_t^B \subseteq S_t$, $S_t^N \cap S_t^B = \varnothing$. Figure 2 shows an example of DPRH, in which $m_N = 1$, $\tau_N = \tau$; $m_B = m$, $\tau_B = m \cdot \tau$ and $S_t^N = S_t \setminus S_t^B$. The black arrows point to the direction of time rolling.

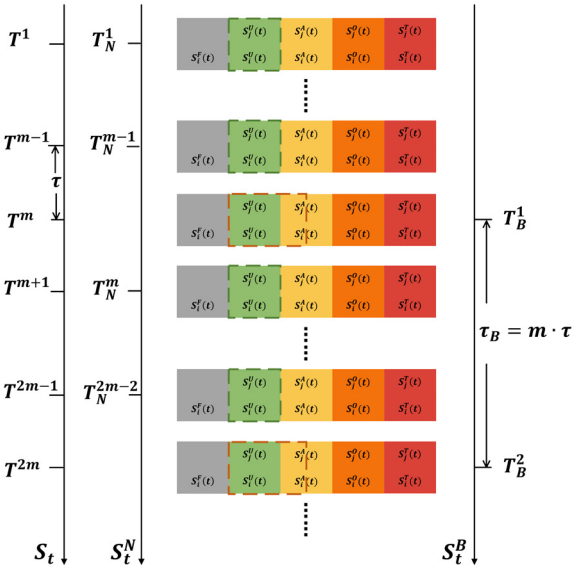

Figure 2. Process of DPRH
Source: Figure 2 is adapted based on the previous work of Ning et al. (2022)

Generally, narrow allocation period $\tau_N = \tau$ is set to improve allocation speed and broad allocation period is set to be longer than narrow allocation period to improve allocation effect and avoid wasting computation resources.

Two appropriate time thresholds T_{AP} and T_{AR} are designed to control allocation scale: for a parking demand $i \in S_i^A(t)$, if $t_i^S \leq t + T_{AP}$, demand i and its allocated parking space j should be involved. The demand set $S_i^{AP}(t)$ and the parking space set $S_j^{AP}(t)$ are settled to represent corresponding demands and parking spaces. Otherwise, demand i and its allocated parking space j are temporarily put aside. For a parking demand $i \in S_i^A(t)$, if $t + T_{AR} < t_i^S$, PRS can change its allocated parking facilities. The demand set $S_i^{AR}(t)$ and the parking space set $S_j^{AR}(t)$ are settled to represent corresponding demands and parking spaces. Otherwise, PRS must fix the allocated parking facility and reallocate parking spaces in the facility to demand

i. The relationship between two-time thresholds is $T_{AP} \geq T_{AR}$.

2.3 Static Allocation Models

To solve the static shared parking allocation problem (SSPAP), the mathematical programming models are formulated. Both models are formulated to maximize the integrated profit of the system, which is decided by revenue and cost. Revenue comes from the parking fees paid by allocated users. Cost refers to the rental paid to providers and the demand waiting penalty. The unit penalty of waiting is q^W.

A binary variable x_{ij} is introduced to represent the allocation result between parking demand i and parking space j. $x_{ij}=1$ if parking space j is allocated to parking demand i, otherwise $x_{ij}=0$. The objective function is given by Equation (1).

$$\max Z(t)=\sum_{i\in S_i(t)}\sum_{j\in S_j(t)}\left(p_j-r_j\right)\cdot\left(t_i^E-t_i^S\right)\cdot x_{ij} -q^W\cdot\tau_N\cdot\sum_{i\in S_i(t)}\left(1-\sum_{j\in S_j(t)}x_{ij}\right) \tag{1}$$

In principle, $S_i(t)=S_i^N(t)$ and $S_j(t)=S_j^N(t)$ if it is narrow allocation, while $S_i(t)=S_i^B(t)$ and $S_j(t)=S_j^B(t)$ if it is broad allocation.

2.3.1 Static Narrow Allocation Model

Narrow allocation is conducted at each narrow allocation time point $t\in S_t^N$. Three binary parameters are designed. c_{ij} expresses the time window relationship between parking demand i and parking space j. $c_{ij}=1$ if $\left[t_i^S,t_i^E\right]$ is within time window $\left[t_j^S,t_j^E\right]$, otherwise $c_{ij}=0$. $c_{ii'}$ expresses the time window relationship between parking demand i and parking demand i'. $c_{ii'}=1$ if time window $\left[t_i^S,t_i^E\right]$ and time window $\left[t_{i'}^S,t_{i'}^E\right]$ have no conflict or $i=i'$, otherwise $c_{ii'}=0$. b_{jf} indicates the belonging relationship between parking space j and parking facility f. $b_{jf}=1$ if parking space j belongs to parking facility f, otherwise $b_{jf}=0$. Narrow allocation is subject to constraints (2)-(7).

$$x_{ij}\leq c_{ij},\forall i\in S_i^N(t),\forall j\in S_j^N(t). \tag{2}$$

$$x_{ij}+x_{i'j}\leq c_{ii'}+1,\ \forall i,i'\in S_i^N(t),\forall j\in S_j^N(t). \tag{3}$$

$$d_{ij}\cdot x_{ij}\leq d_i^{\max},\forall i\in S_i^N(t),\forall j\in S_j^N(t). \tag{4}$$

$$p_j\cdot x_{ij}\leq p_i^{\max},\forall i\in S_i^N(t),\forall j\in S_j^N(t). \tag{5}$$

$$\sum_{j\in S_j^N(t)}x_{ij}\leq 1,\forall i\in S_i^N(t). \tag{6}$$

$$x_{ij}\in\{0,1\},\forall i\in S_i^N(t),\forall j\in S_j^N(t). \tag{7}$$

Constraint (2) claims that each parking demand can be allocated only to the parking space without time window conflict. Constraint (3) implies that every two conflicted parking demands cannot be allocated to the same parking space. Constraints (4) and (5) ensure that only parking spaces satisfying parking demand preferences (i.e., walking distance and parking price) can be allocated to the parking demand. Constraint (6) restricts each parking demand to be allocated to at most one parking space. Constraint (7) defines variable x_{ij}.

2.3.2 Static Broad Allocation Model

Broad allocation is conducted at each broad allocation time point $t\in S_t^B$. A binary parameter y_{ij} is designed to represent the last allocation result between parking demand i and parking space j. $y_{ij}=1$ if parking space j was allocated to parking demand i in the last allocation, otherwise $y_{ij}=0$. Broad allocation model is subject to constraints (8)-(15).

$$x_{ij}\leq c_{ij},\forall i\in S_i^B(t),\forall j\in S_j^B(t). \tag{8}$$

$$x_{ij}+x_{i'j}\leq c_{ii'}+1,\ \forall i,i'\in S_i^B(t),\forall j\in S_j^B(t). \tag{9}$$

$$d_{ij}\cdot x_{ij}\leq d_i^{\max},\forall i\in S_i^B(t),\forall j\in S_j^B(t). \tag{10}$$

$$p_j\cdot x_{ij}\leq p_i^{\max},\forall i\in S_i^B(t),\forall j\in S_j^B(t). \tag{11}$$

$$\sum_{j\in S_j^B(t)}x_{ij}\leq 1,\forall i\in S_i^U(t). \tag{12}$$

$$\sum_{j\in S_j^B(t)}x_{ij}=1,\forall i\in S_i^{AP}(t). \tag{13}$$

$$\sum_{j\in S_j^B(t)}b_{jf}\cdot x_{ij}=\sum_{j\in S_j^B(t)}b_{jf}\cdot y_{ij},\ \forall i\in S_i^{AR}(t),\forall f\in S_f. \tag{14}$$

$$x_{ij}\in\{0,1\},\forall i\in S_i^B(t),\forall j\in S_j^B(t). \tag{15}$$

Constraints (8) - (12) and (15) are derived from constraints (2)-(7), respectively. Constraint (13) specifies that each parking demand in $S_i^{AP}(t)$ must be reallocated to one parking space. Constraint (14) stipulates that each parking demand in $S_i^{AR}(t)$ can be reallocated only to another parking space within the same parking facility.

3. Advanced and Adaptive Tabu Search Algorithm

In districts with heavy parking demands, there could be more than thousands of users reserving and thousands of providers sharing simultaneously through PRS (Yang, Yuan & Meng, 2022). In this condition, the allocation would be excessively time-consuming since the allocation model could have millions of variables and constraints. Therefore, even a greedy heuristic algorithm followed by any heuristic local search algorithm could provide good quality solutions (Mladenovic et al., 2020).

In this section, An Advanced and Adaptive Tabu Search (AATS) algorithm is developed for solving SSPAP, in order to achieve good allocation effect and high allocation speed.

3.1 Main Procedures

Given the initial solution A_{ini}, AATS initializes the current solution A_{cur} and the best solution so far A_{best} with A_{ini}. Naturally, the objective values of the current solution Z_{cur} and the best solution so far Z_{best} are equal to that of the initial solution Z_{ini}. The search iteration starts after initializing candidate list CL, candidate list length CN, tabu list TL, tabu list length TN and termination rule TR. In each iteration, AATS adaptively generates neighborhood solutions A_{can}^{k} from the current solution A_{cur} by an exchange operator and a replacement operator selected by a novel scoring mechanism. CL collects the neighborhood solutions until the size of CL reaches CN. Afterwards, AATS sequences all candidate solutions in CL in descending order of objective values. The first candidate solution is recoded as A^{*}, and its objective value is recoded as Z^{*}. If Z^{*} is not in TL, A^{*} becomes the new A_{cur} and Z^{*} becomes the new Z_{cur}. If Z^{*} is in TL, the next candidate solution becomes A^{*} and its objective value becomes Z^{*} until the new Z_{cur} is generated. If no new Z_{cur} is generated until all candidate solutions are tested, the first candidate solution becomes the new A_{cur} and its objective value becomes the new Z_{cur} according to aspiration criterion. Once the new Z_{cur} is generated, AATS updates TL by adding Z_{cur} as a new tabu object and deleting the headmost tabu object if the size of TL exceeds TN. If Z_{cur} is higher than Z_{best}, A_{cur} becomes the new A_{best} and Z_{cur} becomes the new Z_{best}. Otherwise, A_{best} and Z_{cur} remain unchanged. The iteration stops when the termination rule TR is satisfied. AATS terminates after performing Λ iterations or after maintaining the best solution so far in Λ' iterations. Table 1 shows the pseudo code of AATS. A detailed description and improvements of AATS are given in the following subsection.

Table 1. Pseudo code of AATS

Algorithm 1: AATS

1. Start
2. If $t \in S_t^B$ **then**
3. $S_i \leftarrow S_i^B(t),\ S_j \leftarrow S_j^B(t)$;
4. Else
5. $S_i \leftarrow S_i^N(t),\ S_j \leftarrow S_j^N(t)$;
6. $A_{ini} \leftarrow$ Advanced_Initialization(S_i, S_j);
7. $Z_{ini} \leftarrow$ Objective_Evaluation(A_{ini});
8. $A_{cur} \leftarrow A_{ini}$, $Z_{cur} \leftarrow Z_{ini}$;
9. $A_{best} \leftarrow A_{ini}$, $Z_{best} \leftarrow Z_{ini}$;
10. $CL \leftarrow \varnothing$, $TL \leftarrow \varnothing$, $P \leftarrow 0.5$;
11. While TR is not met **do**
12. $k \leftarrow 1$;
13. **While** $|CL| < CN$ **do**
14. $A_{can}^{k} \leftarrow$ Adaptive_Neighborhood_Generation (A_{cur}, P, S_i, S_j);
15. $Z_{can}^{k} \leftarrow$ Objective_Evaluation(A_{can}^{k});
16. **If** $A_{can}^{k} \notin CL$ **then**
17. $CL \leftarrow CL \cup \{A_{can}^{k}\}$;
18. $k \leftarrow k+1$;
19. $CL \leftarrow CL$ sorted descending order of Z_{can}^{k};
20. $k \leftarrow 1$;
21. **Do**
22. $A_{cur} \leftarrow CL[k]$;
23. $Z_{cur} \leftarrow$ Objective_Evaluation(A_{cur});
24. $k \leftarrow k+1$;
25. **While** $Z_{cur} \in TL \,\&\&\, k \le CN$
26. **If** $Z_{cur} \in TL$ **then**
27. $A_{cur} \leftarrow CL[1]$;
28. $Z_{cur} \leftarrow$ Objective_Evaluation(A_{cur});
29. $TL \leftarrow TL \setminus \{Z_{cur}\}$;
30. **If** $Z_{cur} > Z_{best}$ **then**
31. $A_{best} \leftarrow A_{cur}$, $Z_{best} \leftarrow Z_{cur}$;
32. $TL \leftarrow TL \cup \{Z_{cur}\}$;
33. **If** $|TL| > TN$ **then**
34. $TL \leftarrow TL \setminus \{TL[1]\}$;
35. $P \leftarrow (\beta_\pi / \alpha_\pi) / (\beta_\pi / \alpha_\pi + \beta_\omega / \alpha_\omega)$;
36. Return A_{best};
37. Stop

3.2 Advanced Initialization with Multi-factor Sequencing

Among all modules in AATS, initialization affects the performance of the algorithm directly. Though bidimensional 0-1 matrix $X_{I\times J}$ made of $I\times J$ ($I=|S_i|$ and $J=|S_j|$) binary variables x_{ij} can directly represent a solution.

When solving the large-scale real-world instance, the size of the solution will be too large. Therefore, a unidimensional array A made of I integer variables a_i is chosen to represent a solution. Each variable $a_i \in A$ ranges within $[0,J]$, in which 0 means the status of allocation failure, while $[1,J]$ is the index of allocated parking space. Accordingly, a typical solution is expressed as $A=\{1,2,J,0,\ldots,1\}$.

Instead of randomly sequencing demands, the advanced initialization sequences demands based on priority. According to time window constraints and parking preferences constraints, available parking spaces set S_j^i and conflict demands set S_i^i of each demand are constructed. Above all, reallocation demands take precedence over allocation demands since reallocation should keep parking guarantees. Then, for all demands of the same type (i.e., reallocation demands or allocation demands), the earlier the parking start time, the lower the parking availability (i.e., the number of available parking spaces) and the higher the target parking charge (i.e., the upper bound of acceptable parking price multiplies target parking duration), the higher the demand priority.

Instead of randomly allocating a parking space in S_j^i to each demand, the advanced initialization allocates parking spaces based on priority. On the one hand, each reallocation demand is pre-allocated to the previously allocated parking space. On the other hand, each allocation demand is pre-allocated to the currently best parking space. Specially, only BA requires the above pre-allocations to keep parking guarantees. Then, each demand is allocated to the currently best parking space (no worse than the pre-allocated parking space). In this process, S_j^i is dynamically updated according to S_i^i.

The advanced multi-factor sequencing initialization sequences not only the demands based on multiple factors, but also the parking spaces based on objective values. Compared to the standard initialization without multi-factor sequencing, the advanced initialization is more accessible to a high-quality initial solution. Table 2 shows the pseudo code of the advanced initialization.

Table 2. Pseudo code of the advanced initialization

Algorithm 2: Advanced_Initialization(S_i, S_j)

Start
For each demand $i \in S_i$
  $S_j^i \leftarrow \varnothing$, $S_i^i \leftarrow \varnothing$;
  For each parking space $j \in S_j$
    If $\sum_{f\in S_f} f\cdot b_{jf} \neq \sum_{f\in S_f}\sum_{j'\in S_j(t)} f\cdot b_{j'f}\cdot y_{ij'}$ $\&\&\, t+T_{AR} \geq t_i^S$ **then**
      Continue;
    If $c_{ij}=1 \,\&\&\, d_{ij} \leq d_i^{max} \,\&\&\, p_j \leq p_i^{max}$ **then**
      $S_j^i \leftarrow S_j^i \cup \{j\}$;
  $SS_j^i \leftarrow S_j^i$;
  For each demand $i' \in S_i \setminus \{i\}$
    If $c_{ii'}=0$ **then**
      $S_i^i \leftarrow S_i^i \cup \{i'\}$;
$S_i \leftarrow S_i$ sorted in descending order of $p_i^{max}\cdot\left(t_i^E - t_i^S\right)$;
$S_i \leftarrow S_i$ sorted in ascending order of $|S_j^i|$;
$S_i \leftarrow S_i$ sorted in ascending order of t_i^S;
If $t \in S_t^B$ **then**
  $S_i \leftarrow S_i$ sorted in descending order of $\sum_{j\in S_j^i} y_{ij}$;
  For each demand $i \in S_i$
    If $\sum_{j\in S_j^i} y_{ij}=1$ **then**
      $a_i^0 \leftarrow \sum_{j\in S_j^i} j\cdot y_{ij}$;
    Else if
      $a_i^0 \leftarrow 0$;
      If $S_j^i \neq \varnothing$ **then**
        $S_j^i \leftarrow S_j^i$ sorted in descending order of $\left(p_j - r_j\right)\cdot\left(t_i^E - t_i^S\right)$;
        $a_i^0 \leftarrow S_j^i[1]$;
      For each demand $i' \in S_i^i$
        If $a_i^0 \in S_j^{i'}$ **then**
          $S_j^{i'} \leftarrow S_j^{i'} \setminus \{a_i^0\}$;
$A_{ini}=\varnothing$;
For each demand $i \in S_i$
  $a_i \leftarrow 0$;
  If $S_j^i \neq \varnothing$ **then**
    $S_j^i \leftarrow S_j^i$ sorted in descending order of $\left(p_j - r_j\right)\cdot\left(t_i^E - t_i^S\right)$;
    $a_i \leftarrow S_j^i[1]$;
  $A_{ini} \leftarrow A_{ini} \cup \{a_i\}$;
  For each demand $i' \in S_i^i$
    If $a_i \in S_j^{i'}$ **then**
      $S_j^{i'} \leftarrow S_j^{i'} \setminus \{a_i\}$;
    If $a_i^0 \in SS_j^{i'}$ **then**
      $S_j^{i'} \leftarrow S_j^{i'} \cup \{a_i^0\}$;
Return A_{ini};
Stop

3.3 Adaptive Neighborhood Generation with Bi-Operator Competition

When generating neighborhood solutions, blindly and arbitrarily swapping the parking spaces of two

demands will lead to a large number of unfeasible solutions. Therefore, an exchange operator π and a replacement operator ω are proposed to enrich the selectivity of the operators and improve the efficiency of generating a feasible solution. In each iteration, any operator can be selected as a move.

Exchange operator π works on two allocated demands. It randomly selects allocated demands i and i' until they meet the following criteria. Firstly, their allocated parking spaces j and j' are available for each other based on original S_j^i and $S_j^{i'}$. Secondly, demand i has time conflict with other demands allocated to parking space j' based on conflict demands set S_i^i, and demand i' has time conflict with other demands allocated to parking space j based on conflict demands set $S_i^{i'}$. Once the criteria are achieved, the operator exchanges parking spaces for demands i and i'.

Replacement operator ω works on an allocated demand and an unallocated demand. The replacement operator randomly selects allocated demand i and unallocated demand i' until they meet the following criteria. Firstly, the allocated parking space j is available for demand i' based on original $S_j^{i'}$. Secondly, demand i' has time conflict with other demands allocated to parking space j based on conflict demands set $S_i^{i'}$. Once the criteria are achieved, the operator replaces allocation status for demands i and i'.

In each iteration, the selection of operators determines the effectiveness of neighborhood solutions so that the probability of each operator is the key of selection. Based on a previous work (Han et al., 2021), it can be considered that the probability of each operator can be related to its historical performance.

Therefore, a scoring mechanism is proposed to estimate the historical performances of the operators. In recent Λ^0 iterations, exchange operator π generates α_π candidate solutions and replacement operator ω generates α_ω candidate solutions. The candidate solutions in each iteration are sorted in descending order of objective value and the nth candidate solution is scored with $CN-n+1$. In recent Λ^0 iterations, the total score of candidate solutions generated by exchange operator π is β_π, and the total score of candidate solutions generated by replacement operator ω is β_ω. On this basis, the probability of exchange operator P is calculated in Equation (16). Obviously, the probability of replacement operator is $1-P$.

$$P=\frac{\beta_\pi/\alpha_\pi}{\beta_\pi/\alpha_\pi+\beta_\omega/\alpha_\omega} \tag{16}$$

Table 3. Pseudo code of the adaptive neighborhood generation

Algorithm 3: Adaptive_Neighborhood_Generation (A_{cur} , P, S_i, S_j)

1. **Start**
2. $\rho \leftarrow$ A decimal generated randomly in $[0,1]$;
3. **If** $\rho <$ P **then**
4. **Do**
5. $f \leftarrow 0$;
6. $i,i' \leftarrow$ Two different allocated demands selected randomly from S_i;
7. $a_i \leftarrow A_{cur}[i]$, $a_{i'} \leftarrow A_{cur}[i']$;
8. **If** $a_i \in SS_j^{i'}$ && $a_{i'} \in SS_j^i$ **then**
9. $f \leftarrow 1$;
10. **For each** demand $i'' \in S_i^i$
11. **If** $A_{cur}[i''] = a_{i'}$ **then**
12. $f \leftarrow 0$;
13. **Break**;
14. **For each** demand $i'' \in S_i^{i'}$
15. **If** $A_{cur}[i''] = a_i$ **then**
16. $f \leftarrow 0$;
17. **Break**;
18. **While** $f = 0$
19. **For each** demand $i'' \in S_i^i$
20. **If** $a_{i'} \in S_j^{i''}$ **then**
21. $S_j^{i''} \leftarrow S_j^{i''} \setminus \{a_{i'}\}$;
22. **If** $a_{i_{i''}} \in SS_j^{i''}$ **then**
23. $S_j^{i''} \leftarrow S_j^{i''} \cup \{a_i\}$;
24. **For each** demand $i'' \in S_i^{i'}$
25. **If** $a_i \in S_j^{i''}$ **then**
26. $S_j^{i''} \leftarrow S_j^{i''} \setminus \{a_i\}$;
27. **If** $a_{i'} \in SS_j^{i''}$ **then**
28. $S_j^{i''} \leftarrow S_j^{i''} \cup \{a_{i'}\}$;
29. **Else**
30. **Do**
31. $f \leftarrow 0$;
32. $i \leftarrow$ An allocated demand selected randomly from S_i;
33. $i' \leftarrow$ An unallocated demand selected randomly from S_i;
34. $a_i \leftarrow A_{cur}[i]$, $a_{i'} \leftarrow 0$;
35. **If** $a_i \in SS_j^{i'}$ **then**
36. $f \leftarrow 1$;
37. **For each** demand $i'' \in S_i^{i'}$
38. **If** $A_{cur}[i''] = a_i$ **then**
39. $f \leftarrow 0$;
40. **Break**;
41. **While** $f = 0$
42. **For each** demand $i'' \in S_i^i$
43. **If** $a_i \in SS_j^{i''}$ **then**
44. $S_j^{i''} \leftarrow S_j^{i''} \cup \{a_i\}$;
45. **For each** demand $i'' \in S_i^{i'}$
46. **If** $a_i \in S_j^{i''}$ **then**
47. $S_j^{i''} \leftarrow S_j^{i''} \setminus \{a_i\}$;
48. $A_{can} \leftarrow A_{cur}$;
49. $A_{can}[i] \leftarrow a_{i'}$, $A_{can}[i'] \leftarrow a_i$;
50. **Return** A_{can};
51. **Stop**

The adaptive neighborhood generation with bi-operator competition generates not only more feasible neighborhood solutions based on two operators, but also more effective neighborhood solutions based on the scoring mechanism. Compared to the standard neighborhood generation without bi-operator competition, the adaptive neighborhood generation is more capable of providing high-quality neighborhood solutions. Table 3 shows the pseudo code of the adaptive neighborhood generation.

4. Numerical experiment

This section studies the performance of proposed AATS in a three-day district-level numerical experiment by comparing it with TS and CPLEX solver.

4.1 Experiment Setup

A district in Chaoyang District, Beijing, People's Republic of China, is studied, which has mixed land-use. The historical parking data of the district are selected from 00:00 on April 20th, 2018 until 00:00 on April 23rd, 2018. The overall time horizon is three days and is divided into 4320 time-units, i.e., $\tau = 1$.

The total number of historical parking demands is 20000. For each parking demand, the submission time ranges within [5, 1440] minutes ahead of demand target start time. The upper bound of acceptable walking distance ranges within [100, 700] meters. The upper bound of acceptable parking price ranges within [0.5, 1.2] CNY per 5 minutes according to Beijing Parking Charge Standard (Yang et al., 2017). The upper bound of acceptable waiting time ranges within [1, 10] minutes.

The total number of parking spaces is 1800. For each short-time sharing parking space, the time window ranges within [1, 4320]. The rental price is 0.5 CNY per 5 minutes and the parking price ranges within [0.6, 1.2] CNY per 5 minutes. For each long-time sharing parking space, the time window ranges within [1, 4320]. The rental price is 0.1 CNY per 5 minutes and the parking price is 0.5 or 0.7 CNY per 5 minutes. The unit profit of waiting q^W is set to be 0.025 CNY per minute.

AATS and TS programs are encoded in Visual Studio 2017 using C# language. CPLEX Solver 12.6.3 is evoked via concert technology, coded in C# on Visual Studio 2017. All computations were performed on an ordinary PC.

4.2 Comparison of AATS with TS and CPLEX

Total integrated profit (TIP), total allocated demands (TAD), total space utilization (TSU) and total computing time (TCT) are selected to evaluate the actual allocation effect and allocation speed throughout the overall time horizon.

Apart from the comparison of AATS with CPLEX solver, it is also compared with the Tabu Search (TS). In TS, the initialization is based on greedy strategy (GS), from which the multi-factor sequencing is excluded. The neighborhood generation is based on a swap operator and the scoring mechanism is excluded.

By combining the present data with the ones from previous study (Ning et al., 2022), $\tau_N = 1$, $\tau_B = 10$, $T_{AR} = 15$, and $T_{AP} = 30$ are selected. Table 4 shows the comparison of performance metrics values obtained by AATS, TS and CPLEX.

Table 4. Comparison between AATS, TS and CPLEX on allocation effect and allocation speed

Algorithms	TIP (CNY)	TAD	TSU (%)	TCT (Hour)
CPLEX	218600.60	15107	99.33	1.95
TS	215235.01	14873	99.17	0.25
Gap (%)	**1.54**	**1.55**	**0.17**	**87.18**
AATS	218173.49	15095	99.33	0.28
Gap (%)	**0.20**	**0.08**	**0.00**	**85.47**

Based on TS, the total integrated profit is 215235.01 CNY, the amount of total allocated demands is 14873, total space utilization is 99.17% and the total computing time over three days is 0.25 hour. The gaps of TIP, TAD, TSU and TCT between TS and CPLEX are 1.54%, 1.55%, 0.17%, and 87.18%, respectively.

Based on AATS, the total integrated profit is 218173.49 CNY, the amount of total allocated demands is 15095, total space utilization is 99.33% and the total computing time over three days is 0.28 hour. The gaps of TIP, TAD, TSU and TCT between TS and CPLEX are 0.20%, 0.08%, 0.00%, and 85.74%, respectively.

Apparently, even though there are gaps in allocation effect, both TS and AATS show significant superiority in allocation speed compared to CPLEX. Moreover, AATS is obviously superior to TS in allocation effect since the gaps in TIP and TAD between TS and AATS are up to 1.36% and 1.49%, respectively.

5. Conclusion

In this paper, the dynamic shared parking allocation problem based on PRS with multiple destinations and parking facilities is studied. To improve allocation speed meanwhile maintaining allocation effect, a meta-heuristic algorithm AATS is designed, which consists in the advanced initialization with multi-factor sequencing and the adaptive neighborhood generation achieved with bi-operator competition.

In the three-day district-level experiment in Beijing, P.R. China, the performance of AATS was studied in comparison with TS and CPLEX solver. The gaps between AATS and CPLEX solver in allocation effect are lower than 0.2% while the computing time of AATS is compressed to 85.47%. All these indicate that in large-scale dynamic allocation problems, AATS can achieve high allocation speed while ensuring the allocation effect.

Future research will focus on solving the unpunctuality of users and providers in DSPAP by improving the allocation strategy and allocation models.

Acknowledgements

The research reported in this study was supported by the Beijing Natural Science Foundation (J210001), for whose assistance the authors are very grateful.

REFERENCES

Geng, Y. & Cassandras, C. G. (2013). New "Smart Parking" System Based on Resource Allocation and Reservations, *IEEE Transactions on Intelligent Transportation Systems, 14*(3), 1129-1139. DOI: 10.1109/TITS.2013.2252428

Hampshire, R. C. & Shoup, D. (2018). What Share of Traffic Is Cruising for Parking?, *Journal of Transport Economics and Policy, 52*(3), 184-201.

Han, Z., Han, B., Li, D., Ning, S., Yang, R. & Yin, Y. (2021). Train timetabling in rail transit network under uncertain and dynamic demand using Advanced and Adaptive NSGA-II, *Transportation Research Part B: Methodological, 154*, 65-99. DOI: 10.1016/j.trb.2021.10.002

Krpan, L., Marsanic, R. & Milkovic, M. (2017). A Model of the Dimensioning of the Number of Service Places at Parking Lot Entrances by Using the Queuing Theory, *Tehnicki Vjesnik-Technical Gazette, 24*(1), 231-238. DOI: 10.17559/TV-20160128161848

Mei, Z., Zhang, W., Zhang, L. & Wang, D. (2020). Optimization of Reservation Parking Space Configurations in City Centers through an Agent-Based Simulation, *Simulation Modelling Practice and Theory, 99*, article ID 102020. DOI: 10.1016/j.simpat.2019.102020

Mladenovic, M., Delot, T., Laporte, G. & Wilbaut, C. (2020). The Parking Allocation Problem for Connected Vehicles, *Journal of Heuristics, 26*(3), 377-399. DOI: 10.1007/s10732-017-9364-7

Mladenović, M., Delot, T., Laporte, G. & Wilbaut, C. (2021). A Scalable Dynamic Parking Allocation Framework, *Computers & Operations Research, 125*, article ID 105080. DOI: 10.1016/j.cor.2020.105080

Ning, S., Han, Z., Yang, Y., Yuan, Z. & Wu, X. (2022). CARSP: A Smart Parking System Based on Doubly Periodic Rolling Horizon Allocation Approach, *Journal of Advanced Transportation, 2022*, article ID 1373391. DOI: 10.1155/2022/1373391

Ning, S., Yuan, Z., Han, Z. & Yang, Y. A (2020). Novel Reservation-Based Allocation Mechanism of Private Parking Slots Sharing. In *Proceedings of International Conference on Transportation and Development 2020: Traffic and Bike/Pedestrian Operations, ICTD 2020*, Seattle, WA, United States, (pp. 227-238). American Society of Civil Engineers (ASCE). DOI: 10.1061/9780784483152.020

Shao, C., Yang, H., Zhang, Y. & Ke, J. (2016). A Simple Reservation and Allocation Model of Shared Parking Lots, *Transportation Research Part C: Emerging Technologies, 71*, 303-312. DOI: 10.1016/j.trc.2016.08.010

Xu, S., Meng, C., Xiang, T. R. K., Hai, Y. & Huang, G. Q. (2016). Private Parking Slot Sharing, *Transportation Research Part B, 93*(Part A), 596-617. DOI: 10.1016/j.trb.2016.08.017

Yang, Y., He, K., Wang, Y., Yuan, Z., Yin, Y. & Guo, M. (2022a). Identification of Dynamic Traffic Crash Risk for Cross-Area Freeways Based on Statistical and Machine Learning Methods, *Physica A: Statistical Mechanics and its Applications, 595*, article ID 127083. DOI: 10.1016/j.physa.2022.127083

Yang, Y., Wang, K., Yuan, Z., Liu, D. & Li, G. (2022b). Predicting Freeway Traffic Crash Severity Using Xgboost-Bayesian Network Model with Consideration of Features Interaction, *Journal of Advanced Transportation, 2022*, article ID 4257865. DOI: 10.1155/2022/4257865

Yang, Y., Yuan, Z., Chen, J. & Guo, M. (2017). Assessment of Osculating Value Method Based on Entropy Weight to Transportation Energy Conservation and Emission Reduction, *Environmental Engineering and Management Journal, 16*(10), 2413-2423. DOI: 10.30638/eemj.2017.249

Yang, Y., Yuan, Z. & Meng, R. (2022). Exploring Traffic Crash Occurrence Mechanism Towards Cross-Area Freeways Via an Improved Data Mining Approach, *Journal of Transportation Engineering - Part A Systems, 148*(9). DOI: 10.1061/JTEPBS.0000698