Neural Computing and Applications
https://doi.org/10.1007/s00521-019-04237-2

REAL-WORLD OPTIMIZATION PROBLEMS AND META-HEURISTICS

Solving multiple travelling officers problem with population-based optimization algorithms

Kyle K. Qin[1] · Wei Shao[1] · Yongli Ren[1] · Jeffrey Chan[1] · Flora D. Salim[1]

Received: 4 December 2018 / Accepted: 9 May 2019

Abstract

The travelling officer problem (TOP) is a graph-based orienteering problem for modelling the patrolling routines of a parking officer monitoring an area. Recently, a spatiotemporal probabilistic model was built for TOP to estimate the leaving probability of parking cars, and relevant algorithms were applied to search for the optimal path for a parking officer to maximize the collection of parking fines from cars in violation. However, there are often multiple parking officers on duty during business hours in the central business district, which provides us with the opportunities to introduce cooperation among officers for efficient car-parking management. The multiple travelling officers problem (MTOP) is a more complex problem than the TOP because multiple officers are involved simultaneously in paths construction. In this study, the MTOP is formulated and new components are established for solving the problem. One essential component called the leader-based random-keys encoding scheme (LERK) is developed for the representation of possible solutions. Then, cuckoo search (CS), genetic algorithm (GA) and particle swarm optimization (PSO) are implemented using the proposed components and compared with other state-of-the-art GA and PSO using other solution encoding schemes to solve MTOP. In addition, two greedy selection algorithms are adopted as baselines. The performance of the algorithms is evaluated with real parking sensors data and different metrics. The experimental results show that the performance of CS and GA using LERK is considerably improved in comparison with that of other implemented algorithms.

Keywords Travelling officer problem · Parking officers management · Cuckoo search · Genetic algorithm · Particle swarm optimization

1 Introduction

Rapid urbanization increasingly imposes heavy burdens on the resources and public services of cities. To realize the concept of Smart City, many nations have been seeking solutions by deploying the Internet of things (IoT) for monitoring and managing public affairs [56]. Further, a number of studies have been conducted to facilitate the utilization of urban resources [3, 21]. The deployment of urban IoT could eventually bring many advantages to the administration or optimization of traditional public services, including transportation and car parking [46, 47]. In addition, smart use of the data from mobile devices and infrastructures can offer citizens high-quality travel experiences [2, 31]. As one of the intelligent systems, car-parking management system can work by observing the availability of car-parking spots via in-ground sensors and make the information available to administrators or the public [6]. With the goal of transforming into a Smart City, the City of Melbourne has deployed thousands of in-ground sensors in on-street parking spaces, and these sensors can detect the arrivals and departures of vehicles at parking bays around the city areas [30]. Therefore, the system can inform whether a car exceeds its legal parking period according to the parking rules on that bay. Finally, the parking information can be collected by a management centre that will assign parking officers to the parking violation locations and issue parking tickets.

In 2017, patrolling among different parking slots by a parking officer was modelled as the travelling officer problem by Shao et al. [45]. In the TOP, one officer is

✉ Kyle K. Qin
kai.qin2@rmit.edu.au

[1] RMIT University, 124 La Trobe St, Melbourne, VIC 3000, Australia

Published online: 17 May 2019

responsible for one region in the central business district (CBD) of Melbourne, and a relatively good way is determined for the officer to maximize the number of tickets issued to the parking vehicles in violation by using "first-come first-serve" basis (FCFS), greedy algorithm and ant colonies optimization (ACO) [45]. One essential challenge of this issue is to ensure the timeliness of reaching the targeted parking bays and issuing fines to vehicles. Based on the probabilistic model of violation duration and the walking distance between a parking slot and an officer, the greedy algorithm and ACO [13] were implemented to solve the TOP. And the results showed that these two methods work better than FCFS, which is the current method adopted by the city council. The TOP is a variant of the travelling salesman problem (TSP) or travelling salesman problem with time windows (TSPTW) which are both popular combinatorial optimization problems and classed as NP-hard [50]. The objective of TOP is to let a parking officer visit certain parking bays in violation sequentially during limited working hours while ensuring the maximum number of parking cars caught on the way.

The multiple travelling officers problem (MTOP) can be viewed as an extension of TOP or a variation of the multiple travelling salesman problem (MTSP), which involves guiding more than one officer simultaneously to patrol the entire CBD area for issuing fines on infringed vehicles. The MTOP is more complex than the TOP because it requires cooperation between officers and needs to determine which infringing parking bay should be allocated to an idle officer in real time such that the maximum number of total parking tickets is guaranteed. In contrast, the MTOP is more common in practical situations where multiple people collaborate with each other for the completion of certain tasks.

Figure 1 shows that we have ten parking cars in violation on the map and three parking officers around the area at the moment. In each update period of system, each idle officer would be assigned with a parking bay after the comparison of walking distance or the real-time leaving probability of overstayed cars to officers. After several rounds of assignments according to the suggestion of an optimal algorithm, different tours are formed for different officers eventually. The graph illustrates the final routes recommended by one greedy algorithm for three officers. The route in green colour that has three parking cars is travelled by Officer 1, and the blue and the pink routes are belonged to Officer 2 and Officer 3, respectively.

Previous studies show that TSP or MTSP is NP-hard and has been solved as combinatorial optimization problem by a variety of meta-heuristic algorithms [18, 24, 53]. With respect to the combinatorial complexity of the MTOP, meta-heuristic algorithms can be the appropriate candidates to handle it. A study has showed that meta-heuristic algorithms have the potential to resolve various types of optimization problems in a relatively effective way [17]. Basically, meta-heuristics are capable of searching a designated solution space and discovering a reasonably good solution efficiently. In general, one single solution or a population of elementary solutions is generated at the initial stage and then the algorithms update and examine the solutions iteratively with a variety of search strategies and objective functions to seek an optimal solution for the problems [41]. Some classical meta-heuristics that have been widely used for combinatorial issues are simulated annealing (SA) [50], genetic algorithm (GA) [12], particle swarm optimization algorithm (PSO) [51] and ant colonies optimization (ACO) [24]. In addition, other nature-inspired meta-heuristics that are more recently proposed are bee colonies optimization (BCO) [53], cuckoo search (CS) [54], fruit fly algorithm (FOA) [52], grey wolf optimizer (GWO) [35], dragonfly algorithm (DA) [33] and whale optimization algorithm (WOA) [34].

As a recent meta-heuristic algorithm proposed by Yang and Deb [54], CS has demonstrated its outstanding performance for a series of continuous optimization problems [16, 37]. In solving symmetric TSP, both discrete cuckoo search [41] and random-key cuckoo search [42] can outperform the other algorithms mentioned in the cited papers. GA is a classical meta-heuristic algorithm and has been applied in many studies for solving the TSP or the MTSP [11, 38, 55]. PSO is a popular swarm inspired method and has been compared with GA for coping with discrete optimization problems [28, 51, 57]. Greedy algorithms can usually improve the local optimization such that a globally optimal solution may be yielded in a reasonable amount of time. Furthermore, these algorithms have been proved to work efficiently for solving some combinatorial or scheduling problems [1, 44].

In this paper, we first formulate the model of MTOP and propose a framework based on real spatiotemporal data and the leaving probability model of parking cars to solve the above-mentioned problem. In our strategy, we first introduce a new encoding method called the leader-based random-keys encoding scheme (LERK) for presenting the population of solutions and define a fitness function for the acceptance criterion of optimization. Then, we adopt CS, GA and PSO with the new encoding scheme and fitness function for global optimization, as well as design a greedy sorting approach for local optimization. In addition, two different greedy selection methods that use either distance metric or the leaving probabilistic model are developed as the baselines. Next, we evaluate the capability of these algorithms to search for optimal patrolling tours for different numbers of officers on different days and compare the performance with another GA that uses the two-

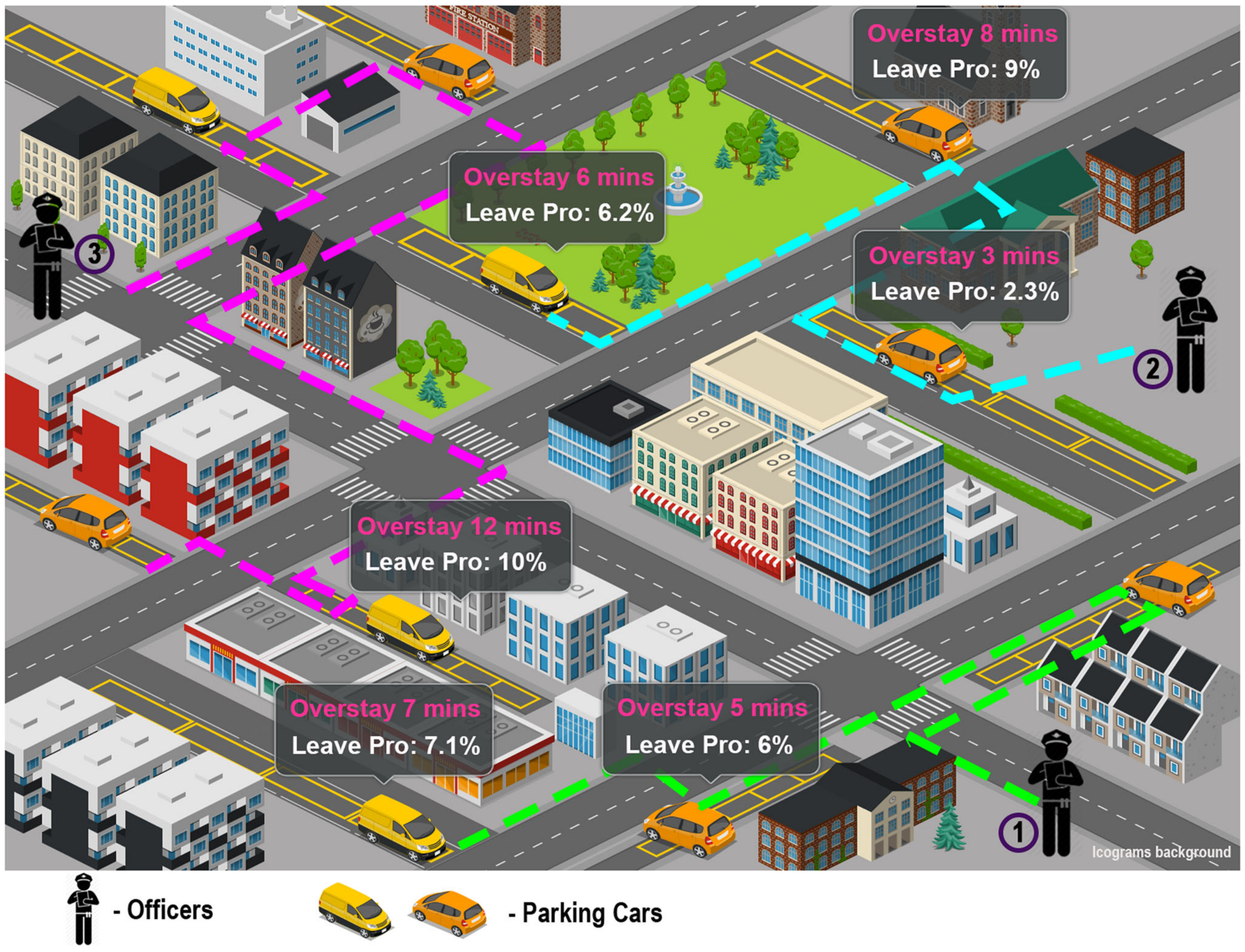


Fig. 1 Demonstration of ten parking vehicles in violation and three routes recommended by an algorithm for officers according to the distance and overstayed information

chromosome encoding technique [55] and an improved PSO using a sequence encoding method [57].

The rest of this paper is organized as follows. Section 2 reviews and summarizes the related works. Section 3 gives an introduction of the on-street parking system and the TOP and defines the model of MTOP. And the details and the new components of our framework to solve MTOP are given in Sect. 4. In Sects. 5, 6, 7 and 8, we present the implementation of CS, GA, PSO and two greedy selection approaches for solving the considered problem, respectively. Section 9 compares and presents the experimental results. Finally, Sect. 10 discusses the limitations along with future work and Sect. 11 provides a conclusion of the study.

2 Literature review

Meta-heuristic algorithms can be typically classified into two groups: single-solution-based and population-based. The former algorithms conduct optimization based on a single solution at any time, but the latter ones perform a search with multiple initial points (solutions) in a parallel style. In general, many meta-heuristic algorithms have revealed their excellent performance in solving combinatorial optimization problems [5]. In 2017, the TOP was studied with a greedy algorithm and ACO, but the MTOP has not yet been explored appropriately [45]. Thus far, a considerable effort has been devoted to seek an optimal solution for the TSP or MTSP using different meta-heuristics [7]. As a variant of the MTSP, the MTOP could be also solved by those methods.

2.1 Single-solution-based heuristics

Two single-solution-based algorithms commonly used in different fields are tabu search (TS) [19] and simulated annealing (SA). In 1989, Malek et al. [27] applied both TS and SA as search techniques to solve the TSP in a parallel computing environment. The parallel implementations help to achieve a super-linear speedup, and the experiment showed that TS outperforms SA with respect to computation time while giving comparable solutions. In 1994, Fiechter proposed another parallel TS algorithm for large TSPs which contain 500–100,000 cities [15]. Its results indicated that TS is suitable for parallel computation, and the search can work efficiently. In 1996, the TS was employed by Carlton and Barnes to solve the TSP with time windows which requires a single vehicle to visit each of a given set of customers exactly once and each visit on one location must be within a defined time frame [9]. In particular, they used a two-level tour hashing scheme within a TS process to detect optimal solutions and to promote a more diverse search.

Another well-known heuristic algorithm called variable neighbourhood search (VNS) was introduced by Hansen and Mladenovic in 1997 [36]. In 2001, to solve the asymmetric travelling salesman problem, Burke et al. [8] embedded a hybrid of HyperOpt and 3-opt within the meta-heuristic framework of VNS, which allows the algorithm to overcome local optima and form high-quality tours. In 2007, VNS was used by Carrabs et al. [10] on a variation of the TSP with pickup and delivery (TSPPD) and the issue requires the vehicle to start from a depot and return to it after all the requests have been satisfied with the additional constraint that the pickup location associated with any given request must be visited before the corresponding delivery location. In their study, three new local search operators were produced for the problem, which were then embedded within a heuristic of VNS. Then, Hemmelmayr et al. [22] modified VNS to solve both the periodic vehicle routing problem (PVRP) and the periodic travelling salesman problem in 2009. Its computational results showed that the new version of VNS is competitive and even better than the existing solutions mentioned in the literature.

2.2 Population-based heuristics

In contrast, the population-based meta-heuristics have several advantages over single-solution algorithms: information exchange between candidate solutions, superior ability of space exploration and avoidance of local optimal solutions [35]. The most prevalent one of the population-based algorithms is GA, and it has been widely used as an optimization solution in many different fields. For instance, to maintain the feasibility of solutions from parents to the offspring in GA, Bean used the random-key technique for enhancing the optimization for multiple machine scheduling and resource allocation problems in 1994 [4]. In 2006, Carter and Ragsdale proposed the two-part chromosome technique to represent the population and related operators for GA to solve the MTSP, as well as compared the computational performance of the proposed method with other approaches [11]. It showed that the two-part chromosome technique can work effectively by reducing the number of redundant solutions in the population. Thereafter, based on the two-part chromosome technique, Yuan et al. [55] introduced a new crossover operator called two-part chromosome crossover (TCX) in 2013 to remove some limitations of previous crossover operator developed by Carter and Ragsdale. The experimental result indicated that this new crossover can ensure a higher diversity of chromosome and improve the quality of solutions more effectively.

PSO is one of the earliest swarm-based algorithms and has been proven to succeed in solving continuous problems [25]. Compared with evolution-based algorithms such as GA and CS, swarm-based algorithms possess some merits by recording search space information over iterations with less operators [34]. In 2003, Wang proposed a new application of PSO by defining the swap operator and the swap sequence to solve the TSP. The results showed that the speed of convergence is rapid and little space is used by the algorithm. The probabilistic travelling salesman problem (PTSP) is one of the major stochastic routing problems that only require a random number of potential customers to be visited by a salesman. In 2010, this problem was resolved by an approach consisting of a new hybrid PSO, the greedy randomized adaptive search procedure and the expanding neighbourhood search strategy [28]. Recently, one new comparative study was conducted by Zhou et al. [57] in 2018 to solve MTSP using both improved GA and PSO with a new sequence encoding method for solutions representation. In this study, the main formulas were modified and new operators were constructed for PSO to update the velocity and the position of particles, which could be compared with our PSO using LERK.

CS is a recent bio-inspired optimization algorithm that was proposed in 2009 and has rarely been studied to solve routing problems. In discrete cuckoo search of 2014, an improved CS was developed by Ouaarab et al. [41] and either 2-opt move or double-bridge move was used for the movement of solutions in search space according to a step length determined by Levy flights. The 2-opt move and double-bridge move are responsible for the small perturbations and the large perturbations that can modify the order of visited cities on a solution. Previous researches have proven that the random-key encoding method is a robust technique to encode a solution with random numbers [4]. In 2015, Ouaarab et al. [42] applied it as solution representation scheme in solving TSP with CS. Levy flights in this study are responsible for generating new solutions by replacing the random keys in old solutions. The experimental results showed that both the discrete CS and random-key CS outperformed the other algorithms mentioned in the papers.

Compared with TSP, the knapsack problem is another classic combinatorial optimization problem that has been formulated as many applications such as resources allocation and cargo loading. FOA is a relatively new global optimization algorithm that was inspired by the hunting behaviours of fruit fly swarms with excellent osphresis and vision. In 2013, a binary fruit fly optimization algorithm (bFOA) is proposed by Wang et al. [52] to solve the multidimensional knapsack problem (MKP). The bFOA uses binary string to represent the solution of the MKP and has three main search processes to perform an evolutionary search, which are smell-based search process, local vision-based search process and global vision-based search process. Then, Jiang and Yang developed a new algorithm called discrete fruit fly optimization algorithm (DFOA) to solve the TSP in 2016 [23]. The DFOA uses a crossover operator to search for the neighbours of the best-known swarm location and the edge intersection elimination (EXE) operator to enhance the exploration performance during the tasting process. The results proved that the DFOA can be an effective alternative method to solve TSP. In 2017, the same problem MKP was solved by Meng and Pan with an improved fruit fly optimization algorithm (IFFOA) [32]. In IFFOA, a parallel search is applied to balance exploitation and exploration, and a harmony search algorithm (MHS) is modified to enhance the cooperation among swarms.

Many other new approaches established in recent years may also help in dealing with our proposed problem. In 2013, Kiran et al. [26] presented an analysis of adopting the discrete version of artificial bee colony algorithm (ABC) with neighbourhood operator for TSP. It showed that the discrete ABC was enriched by 2- and 3-opt heuristic approaches to improve quality of the solutions. In 2016, Osaba et al. [40] proposed a discrete version of the bat algorithm to solve both symmetric and asymmetric TSPs. The results showed that the presented bat algorithm outperformed all the other alternatives in most of the experiments. In 2017, a discrete symbiotic organisms search (DSOS) algorithm was built by Ezugwu et al. [14] to search for an optimal solution for the TSP. The DSOS has three mutation-based local search operators to reconstruct its population, which improves its exploration and exploitation capability and the speed of convergence. In 2018, an enhanced discrete version of water cycle algorithm (DWCA) was proposed by Osaba et al. [39] for symmetric and asymmetric TSPs. Compared with other six different algorithmic counterparts, the DWCA showed a more superb performance in terms of quality, robustness, and convergence of the routes found.

As can be summarized from the literature reviewed thus far, both GA and PSO are the classical population-based meta-heuristics that have a long history of use in solving many specific optimization problems. They are all highly extensible, easy to use and reliable in most of the applications. Because the MTOP is a newly proposed problem, the use of GA and PSO as the initial solutions for this issue will be worth expecting and to the point. In addition, CS is a relatively novel algorithm that has attracted a huge amount of attention from the public after its publication in 2009. Its ease of implementation and outstanding performance in solving similar optimization problems (TSPs) render us an opportunity to investigate our specific problem with it. Therefore, GA, PSO and CS are adopted as our global optimization methods in the initial study of the MTOP. Further exploration of the ability of other new algorithms to solve the issue would be carried out in future.

3 On-street parking system and patrolling issues

3.1 On-street parking system

The City of Melbourne has deployed thousands of sensors on the ground of the street parking bays in CBD area for monitoring the availability and the violation status of parking slots in real time [29, 30]. The parking system can collect the sensors data frequently, which can indicate whether a parking vehicle has infringed the restriction on the bay.

At present, the parking sensors data set can be accessed by public members. There are around 34 million parking records around the CBD area of Melbourne for the year 2016, which include approximately 3000 parking bays with the coordinates information published online. Each parking record contains the bay ID, parking restrictions, arrival time and departure time of each vehicle. In Fig. 2, the blue bar chart shows the number of parking violations monthly in 2016 and the yellow line indicates the month-over-month data. The number fluctuates between 102,000 and 121,000 in the entire year. It is noticeable that January has the lowest number at approximately 102,000 and the figure peaks in May and July, with more than 120,000 violations for each. In addition, we can see that there was a sharp increase of about 15% in parking violations from January to February. Then, it dropped dramatically by 19% in March, which was followed by another increase (18%)

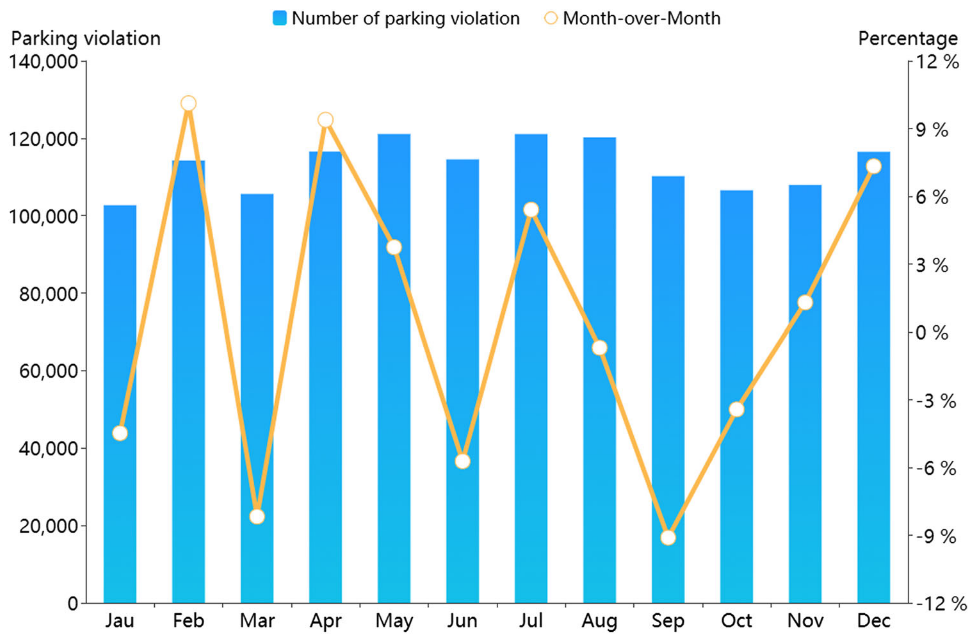


Fig. 2 Monthly number of car-parking violations that occurred in CBD area and the month-over-month data for 2016

in April. The variations between the later months tended to become less.

Figure 3 reveals the distribution of parking violations over nearly 2800 parking bays around the CBD in January 2016. As we can see, in addition to a number of parking violations that occurred in the southernmost and northernmost regions of the map, the other regions with intensive parking violations adjoin each other in the central part of the CBD. This might provide parking officers opportunities to cooperate with each other to patrol across different districts for issuing parking tickets more effectively.

3.2 Overview of TOP

In TOP, each officer is responsible for a region and must walk along the street and issue the parking tickets to the vehicles in violation [45]. The status of each parking bay is dynamically changing, and one officer needs to visit the bays with illegal parking cars sequentially and collect as many fines as possible.

Given a graph $G = (V, E)$, where V is the set of nodes representing the parking bays and E is the set of edges connecting the nodes. For every edge $e_{i,j} \in E$ between two nodes v_i and v_j, there is an associated cost $C(e_{i,j})$ that represents the travel time between two nodes. It is necessary to point out that any node in G can be visited several times at different working moments. Thus, we define that $B(v_i^t)$ denotes the parking fines obtained from node $v_i \in V$ at time t and T is the total working time for an officer.

$$B(v_i^t) = \begin{cases} 1 & \text{if the officer collects parking fines from node } v_i \text{ at time} t, \\ 0 & \text{otherwise.} \end{cases}$$

The objective is to seek an optimal route r during time T that can maximize the number of parking cars in violation caught by an officer. Assume r consists of n nodes and a set of edges E_r.

$$\text{Maximize} \quad \sum_{t=0}^{T} \sum_{i=1}^{n} B(v_i^t), \tag{1}$$

$$\text{s.t.} \quad \sum_{e_{i,j} \in E_r} C(e_{i,j}) \le T, \tag{2}$$

$$\forall v_i \in V. \tag{3}$$

In addition, let t_{overstay} represent the overstayed duration of a car, t_{dep} be departure time and t_{vio} be violation start time. Then, we have the following formula:

$$t_{\text{overstay}} = t_{\text{dep}} - t_{\text{vio}}. \tag{4}$$

A time-based probability model can be established from the exponential distribution of t_{overstay}. and it indicates that most of the parking violations have short t_{overstay} and the number of parking violations decreases exponentially with the passage of time [45]. On the other hand, the car leaving probability $P(t)$ grows correspondingly with an increase in t_{overstay}. This model is important to estimate the overstayed probability of a car in parking bay and guide officer to the bay with higher capture probability.

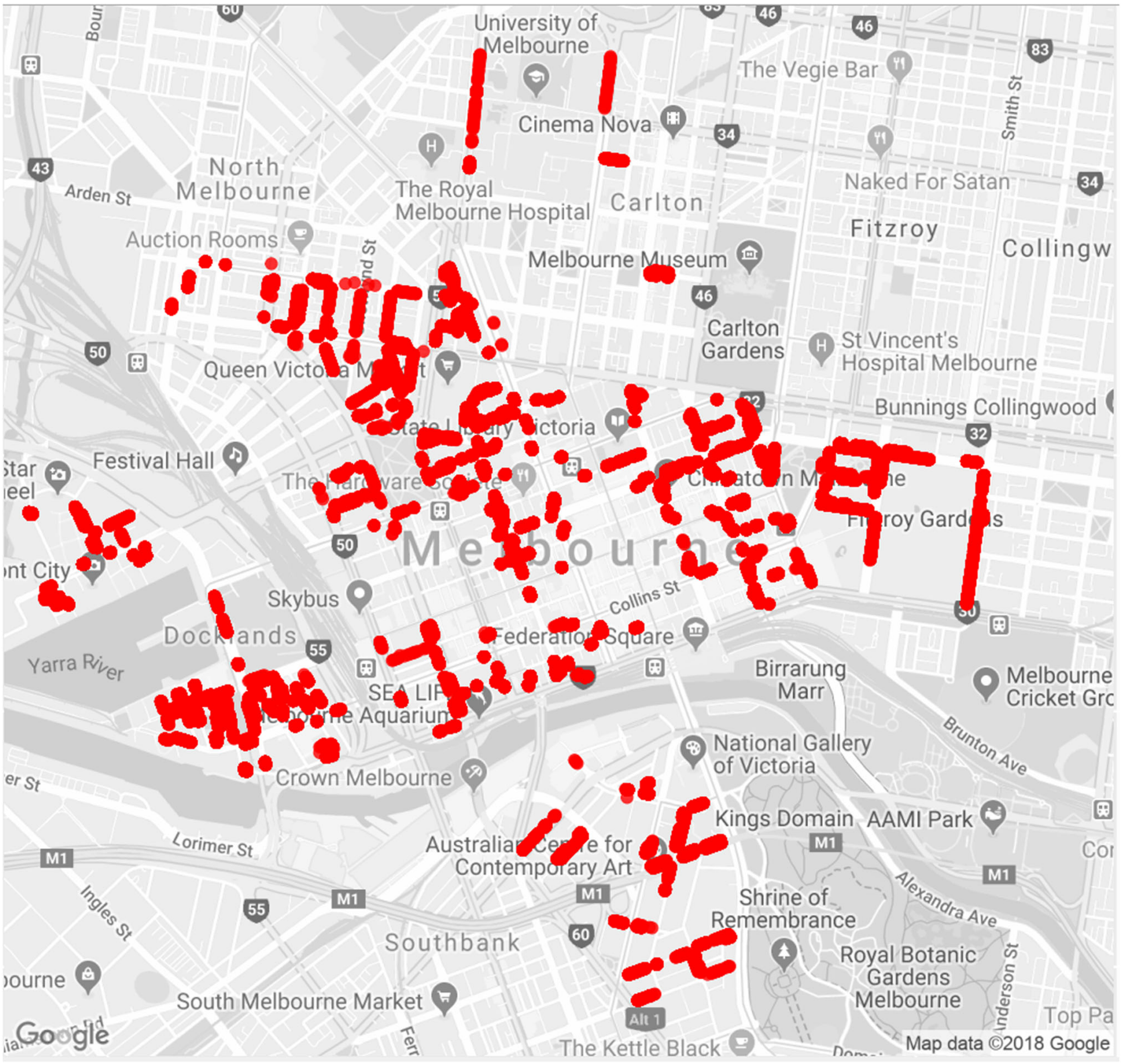


Fig. 3 Distribution of car-parking violations over about 2800 parking bays around the CBD in January 2016

3.3 Multiple travelling officers problem

Compared with the TOP, MTOP is more intricate and requires all the parking officers to travel around the whole map without any area restriction and issue parking tickets to the overstaying vehicles. From the perspective of officers, they need to work with others to patrol the area effectively in handling the illegal parking problem. From the aspect of the system, the assignment of parking bays with violation is a process to select patrolling tasks for the optimal number of working people based on the dynamically changing status and the related spatial information of both parking cars and officers.

As an extension of TOP, we define the following binary variable to record whether one node is visited at a time:

$$x_{v_i}^t = \begin{cases} 1 & \text{if node } v_i \text{ is visited by an officer at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

M is a set of officers who are responsible for all the nodes on the map and their working duration is T. R is a set that contains each route r_k for each officer $k \in M$ who has a fixed salary W. Thus, the minimization of the number of officers while ensuring the gain of maximum parking fines is a matter of great concern. The objective of MTOP is to find an optimal tour for each duty officer such that the total fine collection is maximized with a M as small as possible. And the formulation for the MTOP can be given as follows:

$$\text{maximize} \quad \sum_{t=0}^{T}\sum_{k=1}^{M}\sum_{v_i\in r_k} B(v_i^t) - |M|W, \tag{5}$$

$$\text{s.t.} \quad \sum_{e_{i,j}\in E_{r_k}} C(e_{i,j}) \le T, \quad \forall r_k \in R, \tag{6}$$

$$\sum_{t=0}^{T}\sum_{k=1}^{M}\sum_{v_i\in r_k} x_{v_i}^t = 1, \tag{7}$$

$$x_{v_i}^t \in \{0,1\}, \quad \forall r_k \in R, \quad k \in M, \tag{8}$$

where Eq. 6 denotes a constraint that keeps the total consumed time of each tour within the maximum working hours T. Equation 7 ensures that one node can only be assigned to one officer for visiting at any time during the entire working time.

4 Framework to solve MTOP

To the best of our knowledge, the sensors data can be updated frequently in the parking management system. Furthermore, we assume that the system could record the working status and the location of each officer in real time. As the framework shown in Fig. 4, idle officers and infringed parking bays are first extracted from the system at every assignment time. Then, we combine this information to generate a population of solutions via the LERK. In the following, the meta-heuristic algorithms like CS, GA or PSO are adopted to seek global optimal solutions. During the optimization procedure, a local optimization method is applied to further improve a solution. We compute the average capture probability of overstaying vehicles by officers in each solution for quality evaluation, and the distance information between bays is obtained from Google Matrix API [20]. Note that all the distance data can be pre-stored in the local system.

4.1 Leader-based random-keys encoding scheme

One main challenge in MTOP is that the status of parking bays and the working status of officers are dynamically changing as time goes. Therefore, the idle officers and the targeted parking slots would be different at each task assignment time. We need to construct a temporary path for each currently idle officer for patrolling according to the real-time information. To encode a solution in population effectively, we introduce a new and flexible approach called the leader-based random-keys encoding scheme (LERK). First of all, the simple structure of solution created by LERK could be easily adapted to the implementation of various meta-heuristics on solving the mentioned problem. Second, one idle officer might be arranged with zero parking slot when we create a solution via encoding schemes. One reason for this is that there may be more officers than parking bays in violation at certain moments of task assignment. Another reason is that officers need to cooperate with each other to patrol a large city area and an officer has the chance to skip current assignment if all the parking bays are too far away from him/her. The LERK by nature forms a solution in which an officer can be assigned with zero or more violation bays without explicit configurations. In addition, compared with the other encoding methods mentioned in Sects. 6.2.1 and 7.2.1, all the solutions generated via LERK remained feasible during the optimization process without using specific operations of fixing.

The random-key encoding scheme is a technique that allows a solution to reach a new area by transforming the positions of keys in a continuous space [4]. It uses a vector

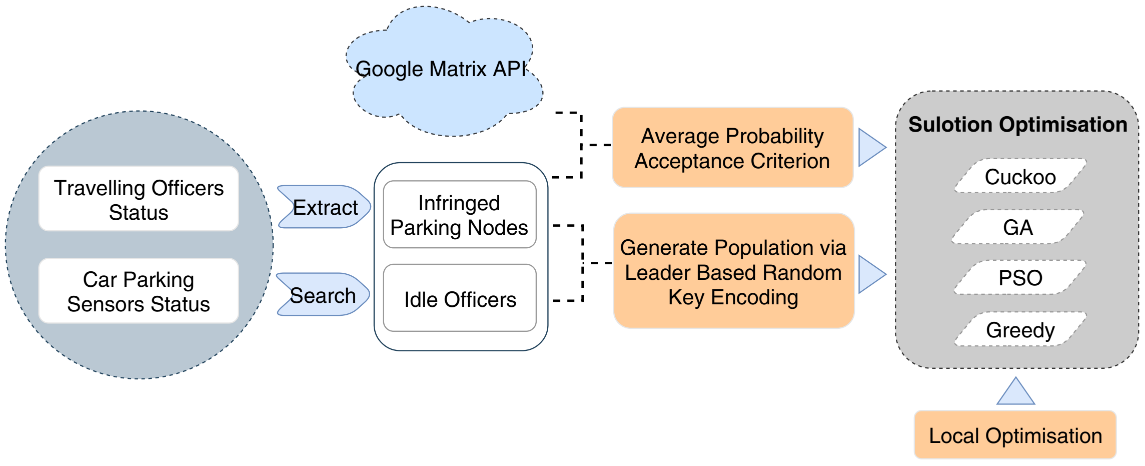

Google Matrix API
Travelling Officers Status
Car Parking Sensors Status
Extract
Search
Infringed Parking Nodes
Idle Officers
Average Probability Acceptance Criterion
Generate Population via Leader Based Random Key Encoding
Sulotion Optimisation
Cuckoo
GA
PSO
Greedy
Local Optimisation

Fig. 4 Framework for solving MTOP

of real numbers as weights to encode a solution, and a new solution can be generated by replacing the weights and reordering the nodes according to the new weights. In MTOP, each officer or parking slot has a unique ID. The IDs of idle officers and violation slots can be used to form a combinatorial link list which represents an initial solution for next task assignment. In the beginning, the LERK combines each of these IDs in the link list with a random number drawn uniformly from (0,1]. Therefore, each node in the list consists of an ID part and a fractional part. The first part contains the ID of either an idle officer or a parking slot, and the fractional part contains the random real numbers. A solution is created by sorting the nodes in link list according to the random numbers in the second part.

In a solution list, the position of officer nodes and parking slots nodes is randomly arranged by LERK. To obtain temporary sub-tours for different parking officers, we allocate every segment of the parking nodes between two officer nodes in list to the former officer as a tour. However, there is one critical problem that the first node of solution list may be a parking node, which can lead to the failure in the assignment of first tour. To cope with this issue, we need to select a leader of officers and set this node in front of the link list for each solution. A leader can be randomly selected among available people, or we can obligatorily configure everyone to be at least one leader when the number of solutions to be initialized is large. This could ensures the diversity in initial population of solutions. The position of a leader node in link list will be kept unchanged during list sorting after the replacement of random keys in non-leader nodes.

As shown in Fig. 5, there are three available officers and five parking bays with violations in the meantime. Firstly, officer $M1$ is randomly selected as leader located at the front of the list. Then, the other officers and parking nodes are appended consecutively to the list with a uniform random number (0, 1] individually. Finally, three tours are constructed for officers $M1$, $M0$, $M2$ after sorting the random keys in non-leader nodes. Note that the sub-tours of all the officers are temporary and would be substituted by new ones at the next arrangement if someone had finished the current visit at hand.

4.2 Local optimization

A local optimization method is useful to search for the optimal solution among the neighbours when a new area is reached by an algorithm. For enhancing the quality of each sub-tour in a new solution, we employ greedy sorting approach to make suitable local movements.

In each round of the greedy sorting approach, we find a parking node in each sub-tour of officer with the largest capture probability of car that was visited from current officer's position (another node v_{current}) and move this node to the front of the sub-tour as next destination v_{next}. At the commencement of the next round, v_{next} becomes the current position v_{current} of the officer and the same computation is conducted on the rest parking nodes. The execution is ended when all the nodes in each sub-tour have been hypothetically visited or a stop condition is met, such as moving only the best n nodes to the front of a sub-path for saving computation time. This sorting method could guarantee that an officer has the opportunity to visit the best parking bays first on his/her temporary tour. Furthermore, we set a probability *localRate* to decide whether this local optimization should be applied on a solution or not during optimization process.

4.3 Acceptance criterion

The acceptance criterion is used to determine whether a solution obtained in search should be accepted for the next iteration in meta-heuristics. In our implementation, the displacement of a solution in population is decided by comparing the fitness between solutions and the fitness is indicated by the average probability of issuing parking tickets on vehicles. The larger the average probability is, the better the fitness of the solution is. This method allows us to easily evaluate and compare the quality of two solutions with any different combination of parking nodes and officer nodes. To more specific, every two solutions contain the same officers and parking nodes in each iteration, but the sub-tour that is formed for an officer in these two solutions could be totally different. At the current stage of our study, the issue is trying to obtain as many benefits (including the number of parking tickets) as possible. One

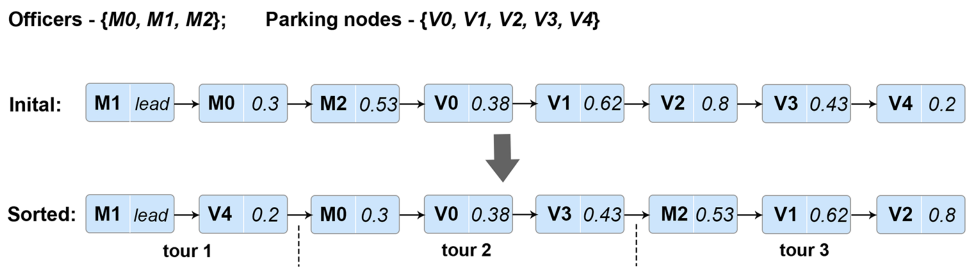

Officers - {M0, M1, M2}; Parking nodes - {V0, V1, V2, V3, V4}
Inital:
M1 lead → M0 0.3 → M2 0.53 → V0 0.38 → V1 0.62 → V2 0.8 → V3 0.43 → V4 0.2
Sorted:
M1 lead → V4 0.2 → M0 0.3 → V0 0.38 → V3 0.43 → M2 0.53 → V1 0.62 → V2 0.8
tour 1
tour 2
tour 3

Fig. 5 Generation of one initial solution via the leader-based random-keys encoding scheme

important aspect of this proposed criterion is its flexibility that enables us to easily ignore the overall detail of a solution and to only concern the number of superior parking nodes obtained by officers in terms of having a larger capture probability of car or remaining within the ideal travel distance on a temporary path. This implementation aims at avoiding the deterioration in the fitness of a solution by some extremely undesirable parking nodes.

Let us assume that a solution S contains a set of subtours R for several officers, and P_{v_i,v_j} represents the current overstay probability of the car in node v_j visited from node v_i by one officer. For only focusing on the superior parking nodes mentioned above, we can just define a probabilistic threshold Ω to exclude the nodes that have high leaving probability of car when computing the overall fitness of a solution. J_{v_i,v_j} is defined to indicate whether P_{v_i,v_j} is larger than Ω ($J_{v_i,v_j} = 1$) or not ($J_{v_i,v_j} = 0$). Therefore, the fitness function is obtained as follows:

$$\text{maximize} \quad \frac{\sum_{r\in R}\sum_{v_i\in r} P_{v_i,v_{i+1}}}{\sum_{r\in R}\sum_{v_i\in r} J_{v_i,v_{i+1}}}, \tag{9}$$

$$\text{s.t.} \quad P_{v_i,v_j} \geq \Omega, \tag{10}$$

$$J_{v_i,v_j} \in \{0, 1\}. \tag{11}$$

4.4 Main procedure

Figure 6 demonstrates the main procedure of our system that works for MTOP, and some relevant system functions and variables are stated in Table 1. After initializing the nodes of parking bays, edges and officers, their working status is scanned and updated at each update time t for finding the violation nodes and idle officers. Then, the data are utilized by LERK to generate a population of solutions that will be manipulated by different algorithms to search for the optimal ones. Next, new assignment will be made for idle officers according to relatively good solution. This process iterates until the working time t exceeds the total working time T, and tours will be formed gradually for all the officers. At the final stage, total rewards and the travelling cost can be calculated and compared for result analysis. The implementation of algorithms for global optimization is described in the following sections.

5 CS for the MTOP

Cuckoo search (CS) is a nature-inspired meta-heuristic algorithm developed in 2009 on the basis of the parasitic behaviour of some cuckoo species [54]. This algorithm is enhanced by the so-called Levy flights and has shown its impressive efficiency in dealing with symmetric TSP [41, 42]. The use of Levy flights offers the algorithm with a better exploratory ability, which is more probably to avoid being trapped in the local search [43]. The MTOP is similar to the TSP which aims at effectively finding an optimal path with the shortest distance for a travelling man. Thus, using CS to solve MTOP could be a new and rational attempt.

CS uses a global exploratory random walk which is carried out using Levy flights:

$$x_i^{t+1} = x_i^t + \alpha \otimes L(s, \lambda), \tag{12}$$

where

Fig. 6 Main procedure to solve MTOP

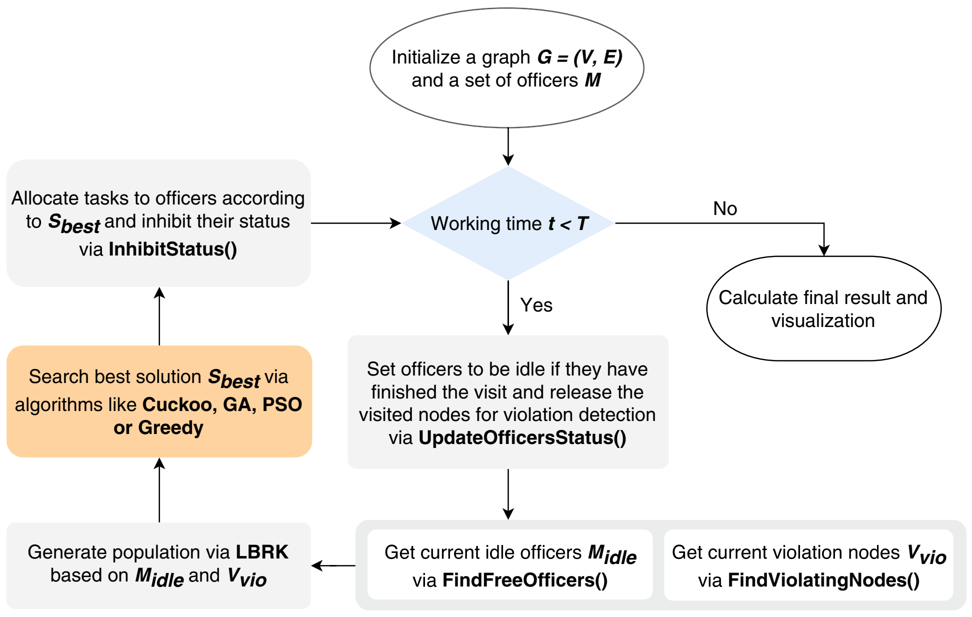

Initialize a graph *G = (V, E)* and a set of officers *M*
Working time *t < T*
No
Yes
Calculate final result and visualization
Set officers to be idle if they have finished the visit and release the visited nodes for violation detection via **UpdateOfficersStatus()**
Get current idle officers M_{idle} via **FindFreeOfficers()**
Get current violation nodes V_{vio} via **FindViolatingNodes()**
Generate population via **LBRK** based on M_{idle} and V_{vio}
Search best solution S_{best} via algorithms like **Cuckoo, GA, PSO or Greedy**
Allocate tasks to officers according to S_{best} and inhibit their status via **InhibitStatus()**

Table 1 System variables and operations

Names	Description
v.status	"booked" or "vacant"—indicates whether node $v \in V$ is allocated or not
m.status	"busy" or "idle"—indicates whether officer $m \in M$ is assigned with a node or not
UpdateOfficersStatus()	Switch the working status of each officer to "idle" if his or her current parking node is visited and release this holding node
FindViolatingNodes()	Find parking bays in violation on map
FindFreeOfficers()	Find parking officers who have finished their current patrolling tasks
GetDistance(v, m)	Get walking distance between node v and officer m
InhibitStatus(v, m)	Switch the state of node v to "booked" and the status of officer m to "busy"

$$L(s, \lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \ (s \gg 0). \tag{13}$$

Eq. 12 is the important stochastic equation for random walk. $\alpha > 0$ is the scaling factor of step size, which is related to the scales of problem [54]. And $\otimes$ indicates an entry-wise multiplication operator. Moreover, step size $L(s, \lambda)$ is drawn from the Levy distribution via Eq. 13.

Based on the original CS, leader-based random-keys cuckoo search (LERK-CS) is implemented with the proposed encoding method to solve the MTOP. It could be seen as a transformation of random-key cuckoo search (RKCS) [42] which aims at solving TSP. Algorithm 1 presents the main steps of LERK-CS in solving MTOP.

At the commencement of the algorithm, the IDs of the car-parking bays in violation and idle parking officers can be obtained in the main procedure of the system (see Fig. 6). Then, an initial population of nests (solutions) is constructed via the LERK with these unique IDs. In the next phase, optimization process of global search will be carried out on the population of solutions.

At the beginning of the global optimization in this CS, we introduce a new parameter called *preFly* which denotes the number of superior cuckoos that fly and seek high-quality nests in each iteration of optimization. Thus, the following step is to discover a number *preFly* of candidate nests by superior cuckoos based on the old ones from a portion p_c of the superior population via Levy flights. During this process, each candidate has a probability to conduct local optimization and then compares with one competitor that is randomly picked from original population. If the candidate has better fitness, the competitor must be replaced with it. After the search of pre-fly cuckoos, a portion p_a of worst nests are abandoned and new individuals are created instead by the movement mechanism discussed in Sect. 5.2. Finally, we rank all the nests in the order of better fitness and this whole procedure repeats until a stopping criterion is met. Furthermore, the configuration of this algorithm's main components is outlined in the following subsections.

Algorithm 1 Leader-based Random Keys Cuckoo Search

```
Input:
    A set of parking bays in violation $V_{vio}$
    A set of idle officers $M_f$
Output:
    The optimal solution $S_{best}$
 1: Initialize a population of nests $S$ via the LERK
 2: while ($l <$ maxIteration) do
 3:     for ($i <$ preFly) do
 4:         Select a nest $S_i$ from a portion $p_c$ of top $S$
 5:         Get a candidate $S_i^{'}$ based on $S_i$ via Levy flights
 6:         if ($uniform(0,1) >$ localRate) then
 7:             Apply local optimization on $S_i^{'}$
 8:         end if
 9:         Evaluate $S_i^{'}$ fitness $F_i$
10:         Select a competitor $S_j \in S$ randomly with fitness $F_j$
11:         if ($F_i > F_j$) then
12:             replace $S_j$ by $S_i^{'}$
13:         end if
14:     end for
15:     Abandon a portion $p_a$ of worse nests and build new ones
16:     Rank $S$ and keep the best individuals at top
17: end while
18: Return $S_{best}$ from $S$
```

5.1 Initialization and fitness function

We initialize the population of solutions randomly via LERK mentioned in Sect. 4.1. And all the random keys of each individual in the population are real numbers that are obtained from (0,1] at random. In addition, the fitness function mentioned in Sect. 4.3 is responsible for evaluating the quality of solutions for the substitution of worse individuals by superior ones in an iteration of optimization.

5.2 Move-generation mechanism

It is essential to maintain the diversity in the population of solutions and prevent the search ending up with a local optima. To achieve this purpose, we employ a move-generation mechanism for generating candidate solutions or replacing the worst ones with new nests in each iteration of optimization. In LERK-CS, we call the solutions in the original population as the base solutions and the candidates

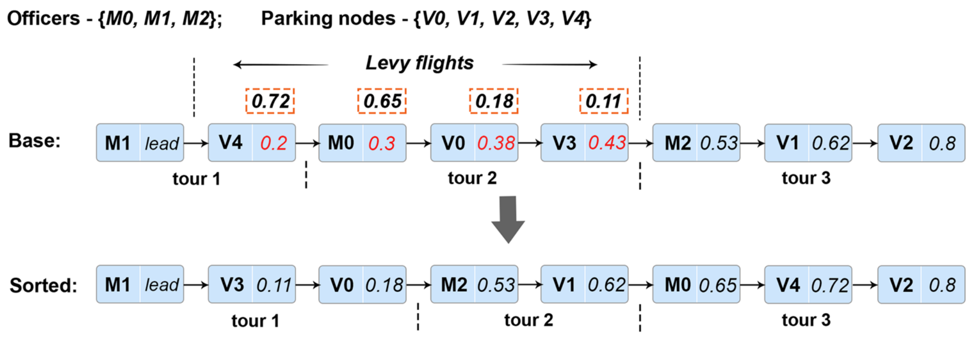


Fig. 7 LERK-CS generates new candidate solution by applying Levy flights on a random number of non-leader nodes in base solution

are generated by replacing the random keys of non-leader nodes in base individuals. Levy flights could play a significant role in letting the shift jump far away in the search space, which could enhance the search of the global optimal solution. Here, it is responsible for updating the random keys in a solution and forming a candidate after sorting the new keys. During the updating stages, all the base solutions are randomly chosen from a portion p_c of the best population. Then, the candidate solutions will be derived from them, which can ensure the inheritance from advantageous individuals among different iterations. Figure 7 illustrates the procedure of generating a candidate solution via the movement mechanism. It shows that four non-leader nodes $V4$, $M0$, $V0$, $V3$ in the base solution are randomly picked and conduct Levy flights for their keys individually. Thereafter, a solution with new routes for officers $M1$, $M2$, $M0$ is born while sorting the list with new keys.

5.3 Configure local optimization

In our CS, before the possible displacement by a new candidate solution that is built via Levy flights for a competitive purpose, we could further improve the solution using the local optimization approach mentioned in Sect. 4.2. Note that a probability *localRate* is used to decide whether a local optimization should be applied or not.

6 GA for the MTOP

Many studies have been conducted to find out how well GA can perform to evolve an optimal solution for the optimization problems. Different types of GA operators were also developed to enhance the search efficiency, such as crossover methods and mutate methods. To evaluate the effectiveness of GA with respect to solving MTOP, we first introduce a new GA called leader-based random-keys genetic algorithm (LERK-GA) as global optimization method. On the other hand, we also adopt one GA using the two-part chromosome crossover (TCX-GA) published by Yuan et al. in 2013 [55] for a comparative analysis. In LERK-GA, the LERK is used to present chromosomes and a new crossover is developed for solution evolution. However, TCX-GA uses the two-part chromosome technique for solutions encoding.

6.1 Establishment of LERK-GA

LERK-GA uses the LERK for population initialization and a new crossover to generate offspring for GA in solving MTOP. We also improve the optimization process via the operators like the reproduction and immigration mentioned in the random-key genetic algorithm that is proposed by Snyder for TSP [49]. As the classical GA, each individual in the population is defined as a chromosome and the nodes inside are viewed as genes.

In the beginning of each evolving time, we rank the chromosomes in population by their fitness and then separate the population into three portions, namely the elitist, the crossover part and the mutate part (see Fig. 8). Then, individuals in the elitist part are copied directly to the next iteration by the reproduction operator and one new crossover operator is developed to spawn offspring which will replace the individuals in the crossover part of the

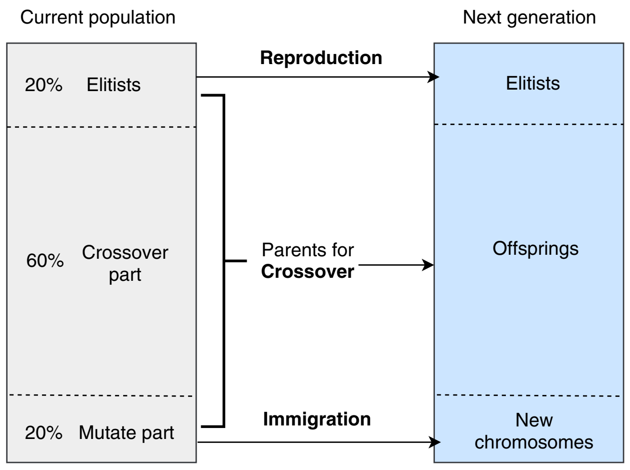


Fig. 8 LERK-GA divides the population into three parts to execute reproduction, crossover and immigration operations for the next generation

population. Last, the immigration operator is applied on the mutate portion of original population. The details of these operators are described in the following subsections, and the steps of the proposed GA are given in Algorithm 2.

Algorithm 2 Leader-based Random Keys Genetic Algorithm

```
Input:
    A set of parking bays in violation $V_{vio}$
    A set of idle officers $M_f$
Output:
    The optimal solution $S_{best}$
 1: Initialize a population of chromosomes $S$ via the LERK
 2: while ($l <$ maxIteration) do
 3:     Rank $S$ and keep the best ones at top
 4:     $S_{new} \leftarrow S$
 5:     Remain a portion $elitistRate$ of best $S_{new}$ unchanged
 6:     for ($i <$ crossoverSize) do
 7:         Select $S_{Mom}$, $S_{Dad}$ from $S$ at random
 8:         Get $S_{child}$ by new crossover on $S_{Mom}$, $S_{Dad}$
 9:         if ($uniform(0, 1) >$ localRate) then
10:             Apply local optimization on $S_{child}$
11:         end if
12:         Replace $S_i \in S_{new}$ by $S_{child}$
13:     end for
14:     Do the immigration on a portion $mutateRate$ of worst $S_{new}$
15:     $S \leftarrow S_{new}$
16: end while
17: Return $S_{best}$ from $S$
```

6.1.1 Initialization and fitness function

While idle officers and parking bays in violation are found by the system, we use LERK to generate the initial population of chromosomes for the problem. And the fitness function mentioned in Sect. 5.1 is used to evaluate the quality of chromosomes for comparison or displacement.

6.1.2 A new crossover

At the beginning of our new crossover, two parents are randomly selected from the former population and then sorted by the ID part of their genes. To obtain a new child with maintaining the diversity in the population, we inherit the genes from both mom and dad chromosomes alternatively with a probability. As an example shown in Fig. 9, either Mom's leader node $M1$ or Dad's ($M2$) is first set as the leader node for the child chromosome (mom is chosen). Next, each of other officer nodes $M0$, $M2$ and parking nodes $V0$, $V1$, $V2$, $V3$, $V4$ is sequentially inherited from either mom or dad with a probability. It is worth noting that if the leader node of dad $M2$ is also picked as child's gene, a random key is required for it. Finally, the child is sorted by the random keys in the genes and new sub-tours are formed for three officers.

6.1.3 Reproduction and immigration

The use of reproduction operator could prevent the degradation of the good solutions among different generations. It is a simple strategy that duplicates a portion of best individuals from original population and delivers them to the next generation.

Immigration helps in introducing new chromosomes to the population, which could ensure the diversity during evolution. In each generation, a portion of worse individuals in former population are replaced by new chromosomes which are created in the same way as in the initialization stage.

6.1.4 Configure local optimization

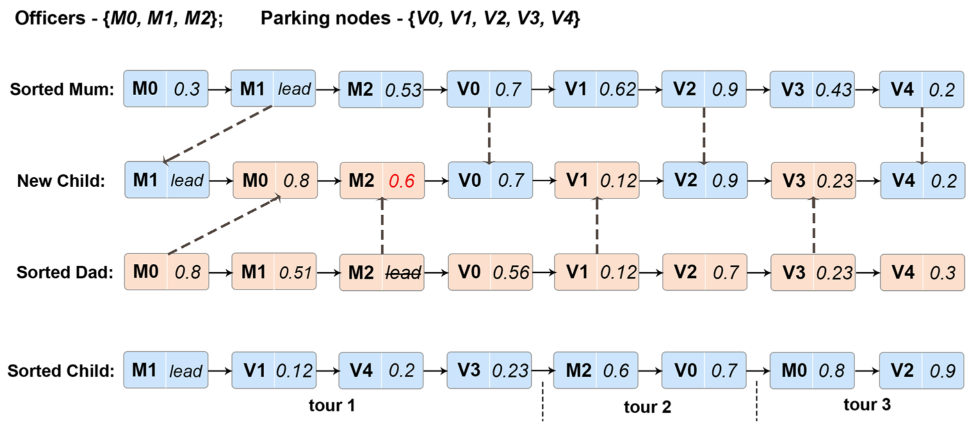
In this GA, the local optimization mentioned in Sect. 4.2 is applied to improve a new child solution created by the new crossover. It is similar to LERK-CS, and there is a probability *localRate* to decide whether the local optimization shall be executed on a new child.

6.2 Implementation of TCX-GA

For comparison, we adopt a recent GA proposed by Yuan et al. in 2013 [55] to solve our problem. In that paper, the two-part chromosome crossover (TCX) was introduced to generate offspring in GA. It is proven that overall quality of solutions could be improved when solving MTSP. In addition, Yuan et al. utilized the two-part chromosome representation to encode solutions which was proposed by Carter and Ragsdale [11]. Noticeably, to solve MTOP and make a comparison with LERK-GA, we need to use the same fitness function and local optimization as those mentioned in our paper for this algorithm (TCX-GA).

6.2.1 Initialization

According to Yuan's approach, we use the two-part chromosome technique to generate the initial population [11]. In MTOP, the first part of each chromosome is a random permutation of N parking slots' IDs and the second part contains the number of parking nodes assigned to each officer. The sum of integers in the second part of a chromosome must be equal to N, and those integers range from 0 to N. Figure 10 shows an example of using the two-part chromosome to represent a solution with seven parking bays and three officers. The first part of genes are filled with all parking nodes in random order, and the second part of the genes indicate the number of parking nodes assigned to officers $M0$, $M1$, $M2$.



Officers - {M0, M1, M2}; Parking nodes - {V0, V1, V2, V3, V4}
Sorted Mum: M0 0.3 → M1 lead → M2 0.53 → V0 0.7 → V1 0.62 → V2 0.9 → V3 0.43 → V4 0.2
New Child: M1 lead → M0 0.8 → M2 0.6 → V0 0.7 → V1 0.12 → V2 0.9 → V3 0.23 → V4 0.2
Sorted Dad: M0 0.8 → M1 0.51 → M2 ~~lead~~ → V0 0.56 → V1 0.12 → V2 0.7 → V3 0.23 → V4 0.3
Sorted Child: M1 lead → V1 0.12 → V4 0.2 → V3 0.23 → M2 0.6 → V0 0.7 → M0 0.8 → V2 0.9
tour 1
tour 2
tour 3

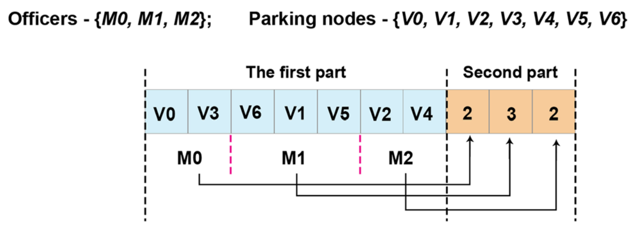
Fig. 9 LERK-GA obtains child chromosome via our new crossover operator based on two parents

6.2.2 Two-part chromosome crossover

We apply the TCX crossover operator to enhance the search performance of TCX-GA for MTOP. It is the same as the original approach described in Yuan's study, TCX handles each officer individually in the first part of the chromosome, which makes the offspring highly inherit the genes from the parental chromosomes [55]. Firstly, two chromosomes of mom and dad are randomly selected from population as the base. Then, genes segment with a random size in each officer's tour is extracted from the first part of mom to the child. Next, Dad's genes with the same parking slots' IDs as the remaining genes in mom are drawn out. In the last step, a random number of those genes drawn from dad will be added to each officer's tour in child chromosome sequentially and these random numbers range from 0 to the current size of the unscheduled genes. Meanwhile, the total length of each officer's tour can be calculated and stored in the second part of the child.

6.2.3 Other operations

Similarly, the roulette wheel selection is used for choosing parents for conducting the TCX crossover or the swap mutation on the first part and the second part of the chromosomes at each iteration of evolution [55]. In roulette wheel selection, each chromosome in population is assigned with a portion of a virtual roulette wheel according to its rank of fitness. Chromosomes which have a higher fitness value are allocated a larger segment of the roulette wheel. The roulette wheel rotates at a random speed, and the individual that the pointer is pointing at is picked when it stops. Furthermore, the replacement strategy to preserve a number of parents or offspring for the next generation is the steady-state GA [55].



Officers - {M0, M1, M2}; Parking nodes - {V0, V1, V2, V3, V4, V5, V6}
The first part
Second part
V0 V3 V6 V1 V5 V2 V4 2 3 2
M0
M1
M2

Fig. 10 Example of the two-part chromosome representation for seven parking bays with three officers

7 PSO for the MTOP

Because of the simple structure of the solution generated by LERK which could transfer a continuous optimization to combinatorial optimization problem, it is also interesting to evaluate the capability of PSO in handling the optimization for MTOP. To gain more insights about the ability of our proposed components, we develop a PSO using LERK (LERK-PSO) and a state-of-the-art PSO published by Zhou et al. in 2018 [57] on current optimization problem. LERK-PSO employs the LERK to encode the initial solutions and a new way to update particles' speed, but Zhou's PSO uses a new sequence encoding method with the Modify Breaks operation to initialize population and three more complex formulas to update the velocity and the position of particles [57]. Here, let us temporarily denominate the latter algorithm to the Breaks particle swarm optimization (Breaks-PSO) in this paper for the purpose of comparison.

7.1 Establishment of LERK-PSO

As the members of population-based meta-heuristics, GA, CS and PSO can be further classified into evolutionary algorithms or swarm-based algorithms, and PSO is one of the most popular swarm-based algorithms for solving optimization problems. PSO considers each particle as a solution in the population, the particles need to move through an n-dimensional search space and the best positions that have been reached are recorded separately for the next comparison [51]. Furthermore, a global best solution is selected from the whole swarm. In each iteration, every particle updates its velocity and position according to the following formulas:

$$V_i' = \omega \times V_i + c_1 \times r_1 \times (S_{pb} - S_i) + c_2 \times r_2 \times (S_{gb} - S_i), \tag{14}$$

$$S_i' = S_i + V_i', \tag{15}$$

where ω is the inertia factor; S_i and S_i' represent current solution and the new solution of a particle, respectively; S_{pb} denotes the best solution this particle has been found and S_{gb} is the global best solution among all the particles; V_i' is the new velocity of the particle and V_i is the old one; c_1 and c_2 are learning factors; and r_1 and r_2 are random uniform numbers between 0 and 1. The following subsections describe our modification on standard PSO for solving the represented problem, and the pseudocode of LERK-PSO is described in Algorithm 3.

7.1.1 Initialization

Each individual in population includes an initial solution S_i generated by the LERK, the current best particle solution S_{pb} (a copy of S_i) and a velocity vector V_i which has the same length as S_i, and the velocity values are random uniform numbers drawn from (0,1]. In the end, there is the best global solution S_{gb} that is chosen among the whole population.

7.1.2 Update particles speed and position

In each iteration, all the particles are required to update the velocity and move to new position. Here, we only have to update the velocity vector V_i and obtain new solution S_i' by recalculating the random keys of current solution S_i in each particle. More specifically, we let key_{pb} to be the random keys vector of particle's best solution S_{pb}, key_{gb} to represent the random-keys vector of the global best solution S_{gb} and key_i to be the random-keys vector of the current solution S_i. Then, we can slightly replace Eqs. 14 and 15 with the following formulas:

$$V_i' = \omega \times V_i + c_1 \times r_1 \times (\mathrm{key_{pb}} - \mathrm{key_i}) + c_2 \times r_2 \times (\mathrm{key_{gb}} - \mathrm{key_i}), \tag{16}$$

$$\mathrm{key_i'} = \mathrm{key_i} + \mathrm{V_i'}. \tag{17}$$

We use Eq. 16 to update the velocity vector in a particle and Eq. 17 to generate a vector of new random keys $\mathrm{key_i'}$ for solution S_i and get a new solution S_i' through sorting new keys in S_i. It is worth mentioning that we only update a key of node in S_i randomly with a probability. To further enhance the search efficiency, we adopt a fuzzy adaptive ω that was introduced by Shi and Eberhart—a random version setting ω to $0.5 + random(0,1)/2$ for dynamic system optimization [48].

Algorithm 3 Leader-based Random Keys Particle Swarm Optimization

```
Input:
    A set of parking bays in violation $V_{vio}$
    A set of idle officers $M_f$
Output:
    The optimal solution $S_{best}$
 1: Initialize a swarm of particles $Y$ via the LERK
 2: while ($l <$ maxIteration) do
 3:     for ($i <$ particleSize) do
 4:         Update velocity $V_i$ of particle $Y_i$ via Eq. 16
 5:         Calculate new keys and get new solution $S_i'$ for $Y_i$ via Eq. 17
 6:         if ($uniform(0,1) >$ localRate) then
 7:             Apply local optimization on $S_i'$
 8:         end if
 9:         Evaluate the fitness $F_i$ of $S_i'$
10:         if ($F_i > F_{pb}$) then
11:             Replace local best $S_{pb}$ in $Y_i$ by $S_i'$
12:         end if
13:         if ($F_i > F_{gb}$) then
14:             Replace global best $S_{gb}$ of swarm by $S_i'$
15:         end if
16:     end for
17: end while
18: $S_{best} \leftarrow S_{gb}$
19: Return $S_{best}$
```

7.2 Implementation of Breaks-PSO

In contrast, we adopt an improved PSO proposed by Zhou et al. [57] to solve MTOP. The main components mentioned in that paper such as the solution encoding method and the operators to update the position and the velocity are also applied to our problem. Noticeably, fitness function and local optimization method described in Sect. 4 are required by Breaks-PSO for either further optimization or fitness assessment on solutions.

7.2.1 Initialization

Zhou et al. [57] introduced a new sequence encoding method to express the actual population by dividing it into the route population and the breakpoint population. It is similar to the two-part chromosome representation in GA, and an individual in the population consists of two parts (a route part and a breaks part) [57]. In this study, the route part is an array that contains the IDs of infringed parking nodes in a random order and the second part is an integer array with each value representing the sum of current nodes in route part that have been assigned to officers. For instance, Fig. 11 gives one example that we have two arrays in one solution: route = ($V0$, $V3$, $V6$, $V1$, $V5$, $V2$, $V4$) and breaks = (2, 5, 7). Generally, the last integer of the breaks can be ignored because it has to be the length of the route array. Moreover, the velocity in each particle is an exchange sequence that is randomly generated and all the numbers in the exchange sequence are random integers from 1 to N (the length of the route array).

7.2.2 Compute particles velocity and position

The mechanism to update the particles' position and velocity is another main distinction from LERK-PSO. It is more intricate in Breaks-PSO due to the need for reconstructing the original formulas and defining new operators for calculating the new velocity and position of the route part and breaks part. The newly defined formulas are as follows:

$$V_i' = \lceil \omega \times V_i \rceil \odot \{\lceil c_1 \times r_1 \times (S_{pb}^{\text{route}} \ominus S_i^{\text{route}})\rceil \odot \lceil c_2 \times r_2 \times (S_{gb}^{\text{route}} \ominus S_i^{\text{route}})\rceil\}, \quad (18)$$

$$S_i^{\text{route}'} = S_i^{\text{route}} \oplus V_i', \quad (19)$$

$$S_i^{\text{break}'} = \lceil r_3 \times (S_{pb}^{\text{break}} \odot S_i^{\text{break}})\rceil \odot \lceil r_4 \times (S_{gb}^{\text{break}} \odot S_i^{\text{break}})\rceil. \quad (20)$$

Here, S_i^{route} and $S_i^{\text{route}'}$ denote current position and new position of particle's route, respectively; S_{pb}^{route} and S_{gb}^{route} are the best route position owned by particle and the global best route position, respectively; S_i^{break} and $S_i^{\text{break}'}$ represent current position and new position of particle's breakpoint, respectively; S_{pb}^{break} and S_{gb}^{break} are particle's own best breakpoint position and the global best breakpoint position, respectively; and r_3 and r_4 are random uniform numbers between 0 and 1.

Equations 18 and 19 allow a particle to update the velocity and the route part by learning from its current best position and the global best position. Equation 20 updates the particle's breakpoint position without velocity. To avoid directly adding or subtracting two exchange sequences by the standard PSO formulas, exchange sequence operators $\ominus$ and $\oplus$ are used to compute new exchange sequences and then obtain new route array. Moreover, similarity calculation $\odot$ is applied to get the intermediate value between two sequences [57]. More explanation of the operators is as follows:

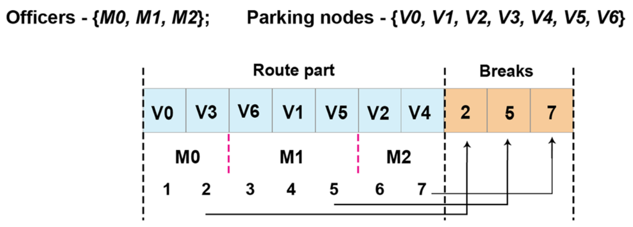


Fig. 11 Example of a solution representation by route-break encoding method for seven parking bays with three officers

- $\oplus$ $L = \{(i,j)\}$ represents an exchange sequence that is used to swap the parking node at positions i and j in the route part of a solution. Assuming one route array is $A = (V0, V3, V2, V1)$ and an exchange sequence is $L = \{(1,2)\}$, we will obtain the new route $B = (V3, V0, V2, V1)$ after $A \oplus L$.
- $\ominus$ Obtain an exchange sequence. If $A = (V0, V3, V2, V1)$ and $B = (V2, V0, V3, V1)$ are two route arrays, exchange sequence $L = A \ominus B$, so $L = \{(1,2),(1,3)\}$.
- $\odot$ Similarity calculation. If $L_1 = \{(1,3),(1,4)\}$ and $L_2 = \{(1,5),(1,3)\}$, we apply the equation $L_1 \odot L_2 = L_1 + (L_2 - L_1) \div 2$, so $L_1 \odot L_2 = \{(1,4),(1,4)\}$.

To ensure the feasibility of the solutions, we must fix the infeasibility of particle route velocity and position. During the computation of an exchange sequence, change the value of the number into 1 if the number in the exchange sequence is smaller than 1, and directly change the value of the number into N if the number in the exchange sequence is greater than N. If the same issue happens after calculating Eq. 20, recalculation is conducted until a feasible solution is found.

8 Greedy selection approaches

Greedy algorithm is simple but has been proven to solve some combinatorial or scheduling problems effectively [1, 44]. We further develop two greedy selection algorithms for the assignment of parking bays to officers using either distance information or the leaving probabilistic model. As is known, the "first-come first-serve" method is currently used by the city council to arrange a valid parking bay with the earliest violation time for an in-charge officer each time in TOP [45]. Thus, we also abide by the rule of "first-come first-serve" while applying greedy algorithms to handle the competition between multiple officers in the bays assignment in the MTOP. And two greedy selection approaches are implemented as the baselines in this paper.

In greedy selection approaches, when one parking violation is detected by system, idle officers are required to compete for the assignment of the targeted bay based on the walking distance or the capture probability of car from each officer. The main difference between two greedy selection methods is that the former one will allocate the detected parking bay to the officer who is closest to it and the latter one will first consider the parking officer who has the maximum likelihood to issue parking ticket on the vehicle. The details of two approaches are described in Algorithms 4 and 5.

Algorithm 4 Greedy Selection by Distance

```
Input:
    A set of parking bays sorted by violation time $V_{vio}$
    A set of idle officers $M_f$
 1: for each node $v \in V_{vio}$ do
 2:     $dis_{nearest} \leftarrow$ null
 3:     $m_{best} \leftarrow$ null
 4:     for each officer $m \in M_f$ do
 5:         $dis \leftarrow$ GetDistance($v, m$)
 6:         if ($dis < dis_{nearest}$) then
 7:             $dis_{nearest} \leftarrow dis$
 8:             $m_{best} \leftarrow m$
 9:         end if
10:     end for
11:     Assign $v$ to $m_{best}$
12:     InhibitStatus($v$, $m_{best}$)
13: end for
14: Return to MTOP main procedure
```

Algorithm 5 Greedy Selection by Probability

```
Input:
    A set of parking bays sorted by violation time $V_{vio}$
    A set of idle officers $M_f$
 1: for each node $v \in V_{vio}$ do
 2:     $p_{max} \leftarrow$ null
 3:     $m_{best} \leftarrow$ null
 4:     for each officer $m \in M_f$ do
 5:         $p \leftarrow$ CalProbability($v, m$)
 6:         if ($p > p_{max}$) then
 7:             $p_{max} \leftarrow p$
 8:             $m_{best} \leftarrow m$
 9:         end if
10:     end for
11:     Assign $v$ to $m_{best}$
12:     InhibitStatus($v$, $m_{best}$)
13: end for
14: Return to MTOP main procedure
```

9 Experimental evaluation

Seven optimization algorithms mentioned in this paper are evaluated with the car-parking sensors data in 2016 which can be fetched from the Open Data Platform [30]. The car-parking system records the arrival time and departure time of the car on each parking bay. The dataset also contains the coordinates and parking rules of the bays, which allows us to compute the violation time and the distance between two bays. And the distance information between parking bays is retrieved from Google Distance Matrix API in walking mode [20].

Intuitively, the use of a central point on the map as the starting location for parking officers could contribute to a relatively fair distribution of people to every direction of the map. Therefore, the initial position of the officers is the City Central Station which is located at the centre of the map. It has been found that most of the parking violations happened during daily working hours. We require all the parking officers to work from 8 am to 6 pm each day to collect parking fines around the CBD in Melbourne (see Fig. 3). When it is closed to the end of working time, we will stop the assignment for an officer if his/her next patrol time to a bay exceeds 6 pm. If a parking bay is visited by an officer, the status of both officer and parking node will be hidden from the following optimization process until the visiting is finished. We assume that the street walking speed of officers is approximately 1.2 m/s and the update frequency of the system to detect violation nodes and available officers is 1 min. The probability *localRate* to conduct the local optimization for a solution is 0.5, and only the best three nodes with maximum capture probability are manipulated by the greedy sorting method for saving computation time. In acceptance criterion, the capture probability threshold Ω to accept a parking node for fitness calculation is 0.21. Moreover, the population size and the number of iterations for all the implemented meta-heuristics are 100 and 300, respectively. The other parametric settings of individual algorithm are given in Tables 2, 3, 4, 5 and 6. Finally, the system was implemented in Python 3.6 and executed on a desktop with Intel i9-9900k CPU 8 Cores 3.6 GHz and Memory 32 GB DDR3. The final experimental results are discussed in the following subsections.

9.1 Evaluation metrics

For investigating the performance of different algorithms with the MTOP model, we first examine the convergence ability of fitness by the meta-heuristics along with the computational time. Then, we apply five metrics to measure the capability of all the algorithms on the fulfilment of patrolling tasks: total number of parking tickets, average walking distance per ticket, average assignment rate, average capture rate and average success rate. For the first metric, we will calculate the total number of parking tickets issued by the officers in a day, and the second one indicates the average travelling meters per ticket in the same day of work. The last three criteria denote the rate of the infringed bays allocated to the officers, the percentage of parking cars in violation caught by officers and the success rate of the attempts to catch infringed cars. The bigger number the fine is or the smaller the total travel distance is, the better is the performance of an algorithm.

We verify the efficiency of different algorithms in gaining parking fines through metrics such as average assignment rate Q_{assign}, average capture rate Q_{capture} and average success rate Q_{success}. The formulas to compute them are as follows:

Table 2 Parametric configuration for LERK-CS

Parameter	Value	Description
p_a	0.3	Portion of worse nests
p_c	0.6	Portion of superior nests
α	0.05	Scaling factor of step size
λ	1	Exponent of a power-law distribution
preFly	0.3	Portion of superior cuckoos that fly and seek high-quality nests

Table 3 Parametric configuration for LERK-GA

Parameter	Value	Description
elitistRate	0.2	Proportion of population for the reproduction operation
crossRate	0.3	Proportion of population for the crossover operation
mutateRate	0.2	Proportion of population for the immigration operation

Table 4 Parametric configuration for TCX-GA

Parameter	Value	Description
crossRate	0.9	Crossover probability rate in GA
mutateRate	0.3	Mutation probability rate in GA
replacePercent	40%	Replacement percentage of steady-state GA

Table 5 Parametric configuration for LERK-PSO

Parameter	Value	Description
ω	1	The inertia weight
c_1	0.1	Learning factor
c_2	0.2	Learning factor
r	0.5	The probability to update a key in one solution

Table 6 Parametric configuration for Breaks-PSO

Parameter	Value	Description
ω	1.1	The inertia weight
c_1	2	Learning factor
c_2	2	Learning factor

$$Q_{\text{assign}} = \frac{1}{D}\sum_{i=1}^{D}\frac{\text{vio}^{\text{i}}_{\text{assign}}}{\text{vio}^{\text{i}}_{\text{total}}}, \tag{21}$$

$$Q_{\text{capture}} = \frac{1}{D}\sum_{i=1}^{D}\frac{\text{vio}^{\text{i}}_{\text{capture}}}{\text{vio}^{\text{i}}_{\text{total}}}, \tag{22}$$

$$Q_{\text{success}} = \frac{1}{D}\sum_{i=1}^{D}\frac{\text{vio}^{\text{i}}_{\text{capture}}}{\text{vio}^{\text{i}}_{\text{assign}}}. \tag{23}$$

Here, D represents the working days; $\text{vio}^{\text{i}}_{\text{total}}$ is the total quantity of parking violation on day i; $\text{vio}^{\text{i}}_{\text{assign}}$ denotes the number of parking bays that had been assigned to the officers during day i; and $\text{vio}^{\text{i}}_{\text{capture}}$ represents the number of infringed cars caught by the officers.

9.2 Performance comparison

Four main experiments were conducted for the evaluation of different algorithms on MTOP. After data extraction and cleaning, we use the parking violation information in January 2016 to build the leaving probability model of cars and picked the data of first week in February for system testing. The parking violation information is shown in Table 7.

9.2.1 Convergence ability of fitness and running time

First of all, we test the ability of five meta-heuristic algorithms on the convergence of fitness along with their computational time. Two greedy selection approaches are excluded from this assessment because they follow the discipline of "first-come first-serve" to arrange officers

Table 7 Parking violation number in a week of February 2016

Day	Total number of parking violations
Monday	1657
Tuesday	1970
Wednesday	1972
Tuesday	1976
Friday	1940
Saturday	1279
Sunday	1643

without the use of the proposed fitness function. To demonstrate the changing history of the best fitness found by each algorithm in MTOP, the search for optimal solution needs to be carried out at one moment of working time period when exist a number of idle officers or parking bays with violation. As mentioned earlier, the task assignment is updated every minute in our simulation system and the parking data at exactly 9 am on Monday are selected for this experiment. In all, 60 parking violations occurred at that time and we assume that the number of idle officers is seven, which is an approximate number found frequently at the system updating time. One setting is that each officer has a chance to skip the patrolling mission during the current assignment suggested by a solution.

Figure 12 illustrates the convergence curves of fitness for five meta-heuristic algorithms with seven officers and 60 parking bays in violation. The experiment was tested under 600 iterations for each algorithm, but we only use 300 iterations for other experiments in the daily patrol of officers. It is found that the optimal solution obtained by LERK-GA has the largest average capture probability (around 0.82) and it could reach 0.80 after 220 iterations. Obviously, LERK-CS has a relatively low optimal fitness (approximately 0.76) but can converge faster than the former algorithm. In contrast, the other three algorithms all began to converge to relatively good solutions after 300 iterations. In this comparison, LERK-PSO has the smallest fitness for the solutions. Moreover, the figure of LERK-CS does not fluctuate as significantly as that of LERK-GA, TCX-GA and LERK-PSO. LERK-PSO fluctuates more intensely than Break-PSO. All of the above suggest that LERK-CS can converge rapidly and smoothly, and LERK-GA tends to obtain a solution with better fitness with the same number of iterations.

The time consumption of the algorithms under the condition mentioned above is given in Table 8. The value before ± is the average running time coming from 30 tests, and the value after ± is the relevant standard deviation. After 600 iterations, LERK-CS consumed the least time to finish one test with about 4.6 min. LERK-GA comes second in this comparison with average of 8.8 min/run. The figure of TCX-GA is twice the time of LERK-GA, and the time of LERK-PSO is slightly higher than that of TCX-GA. At last, Break-PSO has the largest computation time for one test. One main reason why PSO algorithms take a relatively long time is that they need to update the velocity and position for every particle in each iteration. Particularly, Break-PSO involves onerous operations to fix the infeasibility of solutions created during a updating stage.

Another finding to be mentioned is that through the observation of the details of optimal solutions, those with a particularly high fitness tend to create an imbalanced assignment of parking nodes to idle officers. In other words, high capture probability of cars leads to that most of

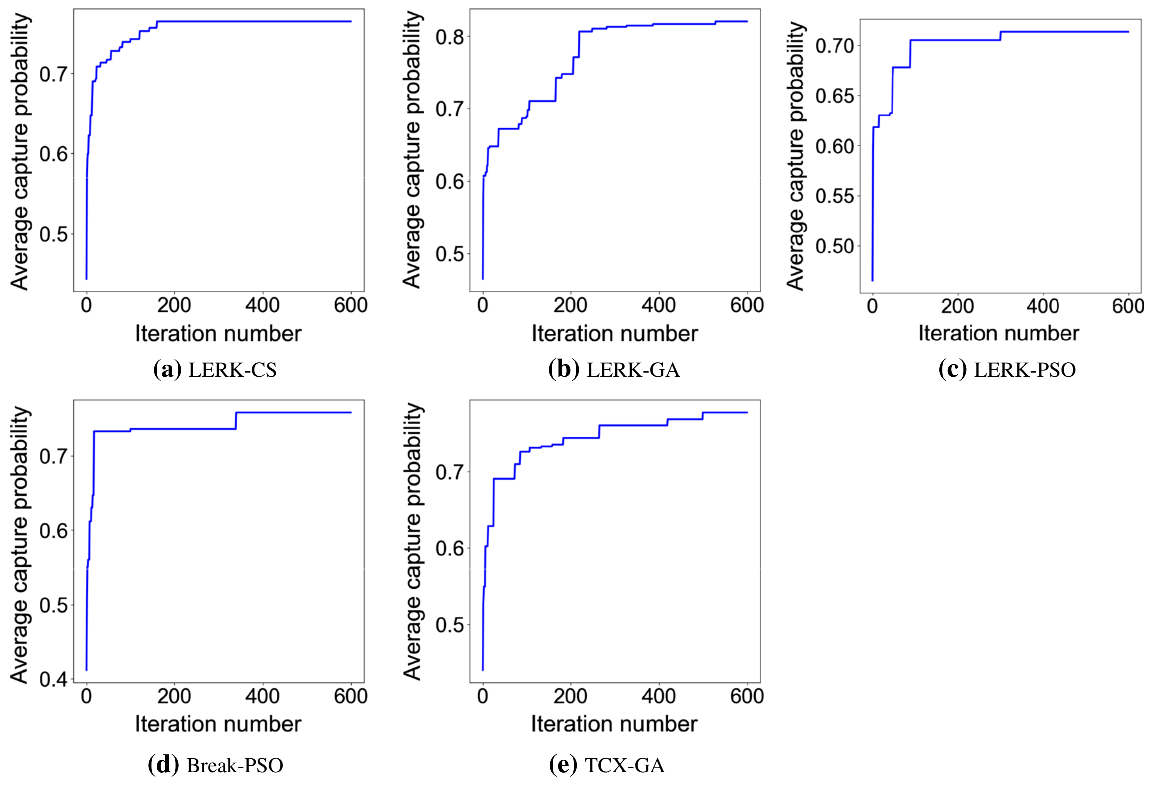


Fig. 12 Convergence ability curve of all the meta-heuristic algorithms using seven officers on 60 parking bays in violation at 9:00 am

Table 8 Computational time for the convergence of fitness (in minute)

LERK-CS	LERK-GA	TCX-GA	LERK-PSO	Break-PSO
4.67 ± 0.22	8.80 ± 0.40	15.76 ± 0.35	18.26 ± 0.60	21.66 ± 1.62

the parking nodes are allocated to a couple of optimal officers and other people may have no task in one arrangement. However, this is a phenomenon commonly existed in the real world, and the officers without tasks will be included in the next arrangement which is executed every minute in our system. More insights into the performance of algorithms are revealed in the following experiments which are conducted on a daily basis.

9.2.2 Daily benefit obtained by distinct number of officers

In this experiment, we compare the ability of the algorithms using different numbers of parking officers in issuing parking tickets during one-day work from 8 am to 6 pm and the corresponding average walking distance required for one ticket. Because the performance of the implemented algorithms is similar on either weekdays or weekends, we choose the results on Monday for a demonstration. We had carried out ten trials for each algorithm with distinct number of officers and the parking data on Monday. Table 9 displays the experimental results about the quantity of parking tickets issued by 15–50 parking officers using different algorithms, and Table 10 shows the average walking distance required for issuing one ticket.

As per the information shown in Table 9, more parking fines are collected with an increased number of officers by all the algorithms. And the figure of all the algorithms tends to converge faster after using more than 40 parking officers. It is noticeable that LERK-CS and LERK-GA collect more parking fines than other algorithms using 15–50 officers for patrolling on Monday. LERK-GA can obtain a few more tickets than LERK-CS when the number of officers is less than 30, but the latter surpasses the former one while we use more than 30 officers. Moreover, two PSO methods show a medium level of fulfilment in this experiment. The figure of TCX-GA is lower than two PSOs with 20 or less officers, but it can converge rapidly when the number of officers is increased.

To further confirm the performance of algorithms in guiding officers for collecting parking fines, we employ the two-sample pooled t test with a significance level at 0.05 (95% confidence) for statistical analysis. Table 11 presents the statistic differences in performance among the algorithms. Each comparison in the table involves two algorithms μ_1 and μ_2, and the null hypothesis can be stated as μ_1 and μ_2 have a similar accomplishment in issuing parking tickets when applied to ten trials. And the alternative hypothesis is that algorithm μ_1 can work better than μ_2. The statistical hypothesis tests in this paper use two-tailed critical regions to indicate whether algorithm μ_1 is significantly better than μ_2 ($t > 2.10$) on fines collection or vice versa ($t < -2.10$), with the rejection of the null hypothesis. Obviously, GA, CS and PSO that are incorporated with LERK show a more superb performance in this experiment than the other algorithms. However, there is no sufficient evidence to support whether LERK-CS or LERK-GA is more advantageous than each other.

Table 10 illustrates that the average walking distance for issuing one parking ticket decreases gradually when more parking officers are involved simultaneously. The data reveal an equivalent outcome to that given in Table 9, LERK-CS and LERK-GA could contribute to the least walking meters for one parking ticket when the number of officers ranges from 15 to 50. Furthermore, the figure in the tables also indicates that LERK-PSO has a tendency to obtain more parking tickets than Breaks-PSO, but Breaks-PSO needs less travelling distance to get one ticket. The

Table 9 Results for issuing parking tickets on Monday with multiple parking officers

Algorithms	$M = 15$		$M = 20$		$M = 30$		$M = 40$		$M = 50$	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
LERK-GA	966	20	1155	27	1380	14	1504	18	1561	7
LERK-CS	953	20	1121	18	1404	22	1516	11	1560	6
LERK-PSO	906	18	1048	18	1229	18	1334	21	1420	13
Break-PSO	843	18	940	13	1072	15	1191	24	1336	16
TCX-GA	409	47	800	42	1175	21	1370	20	1505	14
Greedy-Dis	249	0	359	0	496	0	682	0	998	0
Greedy-Pro	249	0	360	0	490	0	687	0	989	0

M is the number of officers, and mean is the average number of parking tickets gained by each algorithm in ten trials

Table 10 Results for the distance walked by parking officers to issue one parking ticket

Algorithms	$M = 15$		$M = 20$		$M = 30$		$M = 40$		$M = 50$	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
LERK-GA	500	16	516	27	466	19	372	28	313	10
LERK-CS	538	15	581	17	462	27	376	16	330	7
LERK-PSO	599	16	683	15	770	23	761	30	666	19
Break-PSO	577	18	604	20	593	18	587	31	524	20
TCX-GA	1530	201	1005	51	960	28	753	48	482	29
Greedy-Dis	2514	0	2324	0	2453	0	2258	0	1612	0
Greedy-Pro	2514	0	2324	0	2469	0	2216	0	1656	0

M is the number of officers, and mean is the average meters travelled for issuing a parking ticket in ten trials

Table 11 Statistical differences about issuing parking tickets between the algorithms, which is indicated by two-sample *t* test with a significance level at 0.05

μ_1	μ_2	$M = 15$		$M = 20$		$M = 30$		$M = 40$		$M = 50$	
		t^*	p^*	t^*	p^*	t^*	p^*	t^*	p^*	t^*	p^*
LERK-GA versus	Break-PSO	14.45	< 0.001	22.69	< 0.001	47.46	< 0.001	32.99	< 0.001	40.71	< 0.001
	TCX-GA	34.48	< 0.001	22.48	< 0.001	25.68	< 0.001	15.75	< 0.001	11.31	< 0.001
	Greedy-Dis	113.37	< 0.001	93.23	< 0.001	199.67	< 0.001	144.41	< 0.001	254.34	< 0.001
	Greedy-Pro	113.37	< 0.001	93.23	< 0.001	201.03	< 0.001	143.53	< 0.001	258.40	< 0.001
LERK-CS versus	Break-PSO	12.93	< 0.001	25.78	< 0.001	39.43	< 0.001	38.93	< 0.001	41.45	< 0.001
	TCX-GA	33.68	< 0.001	22.21	< 0.001	23.81	< 0.001	22.23	< 0.001	11.42	< 0.001
	Greedy-Dis	111.31	< 0.001	133.86	< 0.001	130.51	< 0.001	239.76	< 0.001	296.20	< 0.001
	Greedy-Pro	111.31	< 0.001	133.89	< 0.001	131.37	< 0.001	238.32	< 0.001	300.94	< 0.001
LERK-PSO versus	Break-PSO	7.83	< 0.001	15.38	< 0.001	21.19	< 0.001	14.18	< 0.001	12.88	< 0.001
	TCX-GA	31.23	< 0.001	17.16	< 0.001	6.17	< 0.001	− 3.92	0.001	− 14.07	< 0.001
	Greedy-Dis	115.42	< 0.001	121.04	< 0.001	128.77	< 0.001	98.18	< 0.001	102.65	< 0.001
	Greedy-Pro	115.42	< 0.001	120.87	< 0.001	129.83	< 0.001	97.43	< 0.001	104.84	< 0.001
LERK-GA versus	LERK-CS	1.45	0.16	3.31	0.004	− 2.91	0.009	− 1.79	0.09	0.34	0.74
	LERK-PSO	7.05	< 0.001	10.42	< 0.001	20.94	< 0.001	19.43	< 0.001	30.20	< 0.001

In the table, t^* and p^* denote *t* values and *p* values

greedy algorithms require the largest walking distance per ticket, but their figures dropped and converge to other advantageous methods with more officers. We observed that all the implemented meta-heuristics could work more efficiently than the baseline approaches in gaining parking fines with less travel distance.

Finally, we briefly discuss and compare the CPU computational time of the algorithms for completing one-day work which lasts for 10 hours. As mentioned before, the simulation system starts an detection of current parking violations and idle officers and updates the routes every minute. Without the iterations of optimization process, most tests of both greedy algorithms can finish a one-day simulation in 15 min, which is significantly less than that of the meta-heuristic algorithms. Among those meta-heuristics, LERK-CS consumed the smallest time for computation and two PSOs have the largest computational time. For instance, when using 30 parking officers for patrolling, LERK-CS averagely spent approximately 200 min on one-day test and LERK-GA averagely needed one-third more time than the former one. TCX-GA and LERK-PSO are twice and thrice the running time of LERK-GA, respectively. However, Break-PSO consumed five times more execution time than LERK-GA for one trial.

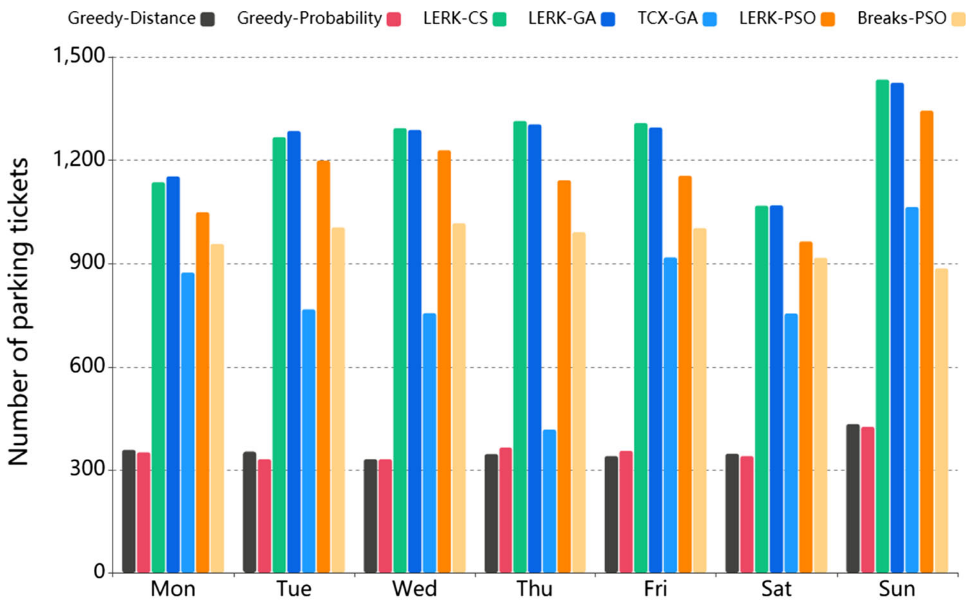
9.2.3 Number of parking tickets issued in a week

From the previous comparison, we know that there is a major disparity between the capability of different algorithms when using 20–40 people for patrolling in one day. To examine the stability of the algorithms in solving

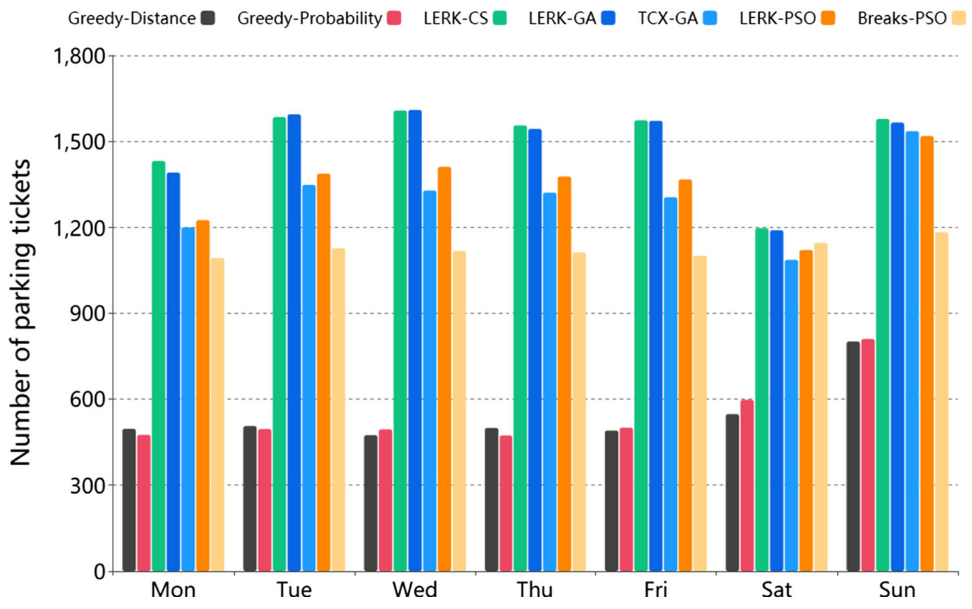
MTOP in different days, we carried out an experiment for each algorithm on one-week data with a group of 20 and 30 officers.

Figure 13a shows that LERK-GA and LERK-CS have the tendency to collect the largest number of parking fines using 20 officers, with more than 1100 tickets for every day except for Saturday. In this comparison, LERK-PSO takes up the third position, with more than 1000 parking tickets per day. Breaks-PSO has a similar performance in most of the days, with about 900–1000 tickets each day. The number of TCX-GA is lower than that of other meta-heuristics, but higher than that of two greedy selection approaches. Compared with that in case of the group of 20 people, the total number of tickets issued increases dramatically when the number of officers is added to 30. Figure 13b shows that both LERK-CS and LERK-GA have a similarly good figure, with the tickets number ranges from around 1350 to 1550 on weekdays and Sunday. There is a huge improvement in the performance of TCX-GA which is comparable to LERK-PSO in all the considered days, but all their numbers are smaller than those of LERK-CS and LERK-GA. As the baselines, two greedy selection

Fig. 13 Total number of parking tickets issued by a group of 20 and 30 officers in one week



(a) Number of parking tickets issued by 20 officers daily.



(b) Number of parking tickets issued by 30 officers daily.

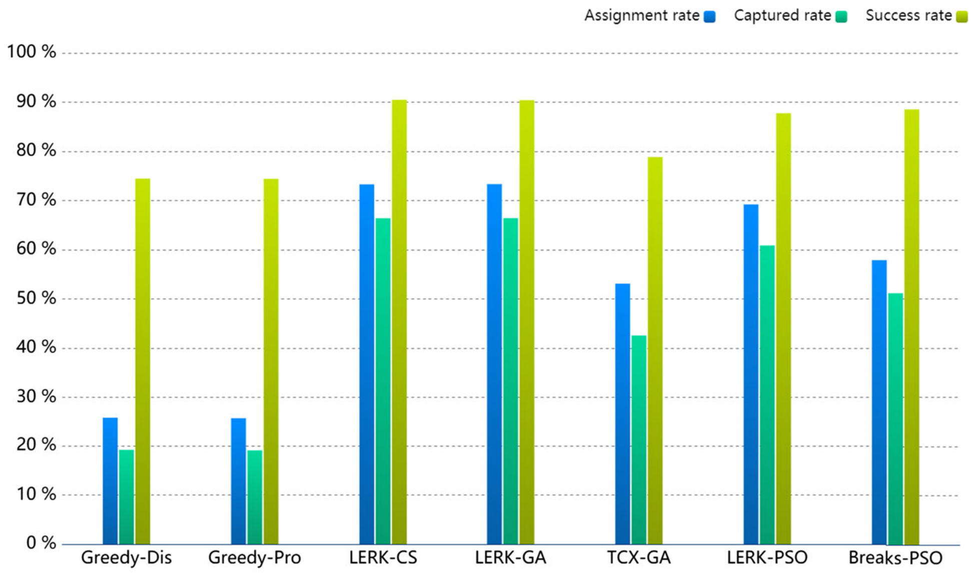
algorithms exhibit an increase from less than 400 tickets issued by 20 officers to more than 450 when using 30 officers. As a result, the graphs point out that the algorithms mentioned show a steady ability to solve MTOP on different days with a fixed number of officers.

9.2.4 Effectiveness of officer patrol

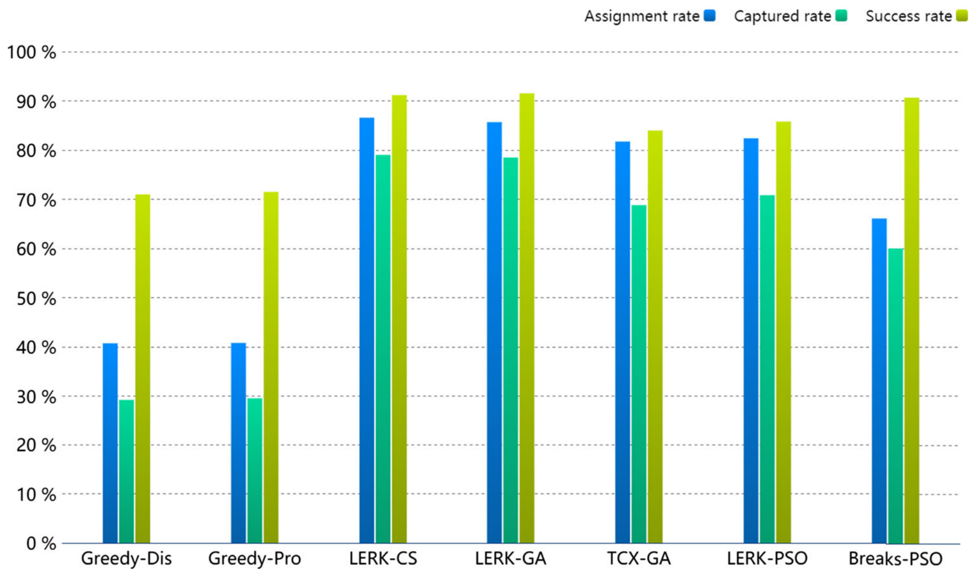
In this section, we evaluate the effectiveness of different algorithms in leading officers to issue parking tickets using the average assignment rate (Eq. 21), average capture rate (Eq. 22) and average success rate (Eq. 23), where D is seven days in the week mentioned in Table 7. The assignment rate, capture rate and success rate shown in Fig. 14 are the average daily results of different algorithms for a week.

Figure 14a illustrates that two greedy selection algorithms have similar but the lowest ratios for the assignment of parking bays and the capture of infringed cars by 20 officers, with around 25% for the average assignment rate and 18% for average capture rate. In contrast, LERK-GA and LERK-CS have an average of around 70% of the violation bays assigned to officers in one week and approximately average 67% of vehicles in violation had been issued with tickets. In addition, the average assignment rate and capture rate of TCX-GA are only

Fig. 14 Compare the average assignment rate, average capture rate and average success rate between algorithms in one week with different number of officers



(a) Average assignment rate, capture rate and success rate for 20 officers during a week.



(b) Average assignment rate, capture rate and success rate for 30 officers during a week.

approximately 54% and 43%, respectively. Two PSOs have a superior ratio for these two metrics compared with TCX-GA. As for the average success rate, the figures of LERK-CS and LERK-GA are the highest (90%) among the methods mentioned. Greedy selection algorithms and TCX-GA can successfully catch about 75–78% of the vehicles at visited parking bays. The average success rate of both PSOs is a little lower than that of LERK-CS and LERK-GA.

As shown in Fig. 14b, there is an apparent increase in the average assignment rate and capture rate for all the algorithms when using 30 officers for patrolling, particularly LERK-CS and LERK-GA. Although the average assignment rate and capture rate have been risen, the average success rate changes very little for all the methods compared with Fig. 14a. Only TCX-GA and Break-PSO can have a big improvement in the success rate of cars capture when using 30 officers. Note that LERK-CS and LERK-GA have an almost similarly good performance in this experiment whenever the number of officers is 20 or 30.

9.3 Analysis of algorithms and results

Different meta-heuristics and their variants apply different strategies for solution selection, updating or substitution, and these operations can potentially affect the performance for solving specific optimization problems. Traditionally, GA imitates the biological evolution procedure of chromosomes using selection, crossover and mutation operators that could influence the convergence of the algorithm. TCX-GA uses roulette wheel selection to choose parents from the population for crossover and mutate operation. The size of every part of the roulette represents the fitness value of a solution, which implies that chromosomes with better fitness are more likely to be selected than those with lower fitness. Meanwhile, steady-state GA that is applied as replacement policy requires a proportion of parents and their offspring to compete for survival. In TCX-GA, the TCX crossover ensures the inheritance of highly fit subtours from the parental chromosomes and the diversity in the second part of chromosomes. However, good new or old chromosomes may be discarded during the process of replacement and the total number of superior chromosomes is not guaranteed. Compared with other meta-heuristics, TCX-GA shows a relatively inferior capability in all the experiments when the number of officers is less than 30, but the gap reduces considerably when more officers are used for the patrolling.

In contrast, as the most advantageous algorithm in experimental evaluation, LERK-GA uses the reproduction operator to replicate the best individuals from one generation to the next. This operation can improve the elitist solutions monotonically among different generations and ensure a constant supply of good individuals for mating. Next, a new crossover method (Fig. 9) is designed for the creation of child chromosomes encoded by LERK. The crossover operator provides new chromosomes with a chance to be better than the replaced ones. In other words, the search is oriented towards the subregions of the search space in this process, where an optimal solution is supposed to exist. Furthermore, a small portion of the worst population conduct the immigration operation which could enhance the diversity in the population and accelerate the convergence speed.

In the comparison of the overall performance among the implemented algorithms, LERK-GA and LERK-CS outperform others in terms of the speed of fitness convergence, the number of daily parking tickets collected, the efficiency in the assignment of violation bays and the quality of task completion. LERK-CS first arranges a number of cuckoos to explore the areas around the current best nests via Levy flights, which could help new candidates to jump out of local optimum. The new candidates found by the cuckoos then compete with old nests at random for survival in the population. Like LERK-GA, LERK-CS also discards a portion of worst solutions in the population and introduces new ones instead at the last stage of each iteration. But a main difference in this process is that LERK-CS generates new solutions based on superior ones via our movement mechanism which is empowered by LERK and Levy flights. The movements made on superior nests could maximize the probability of the detection of an optimal nest.

Generally speaking, one significant merit of PSO is that it preserves search space information over subsequent iterations while evolution-based algorithms (CS and GA) may drop previous information while a new population is formed. Furthermore, PSO usually uses less operators than the evolutionary approaches (selection, crossover, mutation, elitism, etc.) and thus is easier to implement [34]. In this paper, both PSOs exhibit a competitive performance on solving MTOP. The particles in PSO can share their information about the search space, and in each iteration, every particle computes its new velocity and position by considering its current best solution as well as the global best solution in the swarm. However, PSO may face up to the slow convergence rate, parameter selection problem and easily get trapped in a local optimum because of its poor exploration when solving complex multimodal problems [5]. A particle sometimes falls into a local optimum, and its velocity and position cannot change effectively. To largely avoid this problem, LERK-PSO uses random keys to encode the solutions and generates new solutions by updating the keys directly with the standard equations of PSO. Furthermore, a random setting ω used in LERK-PSO can also help the algorithm to leap out the local search space and speed up the convergence.

In addition, an essential aspect is worth mentioning: the random-keys encoding method has the ability to switch between the continuous and the combinatorial search space, which enables GA, CS or PSO to easily adopt specific search strategies to ensure a good balance between intensification and diversification. With little effort, we can devise an effective movement mechanism based on the encoding method for each meta-heuristic to handle this type of balance via making a small motion in a local region or a big exploration to a remote region. In MTOP, the combination of the random keys and optimization algorithms not only dramatically prevents the specific repair for the infeasibility of solutions, but also can assure sufficient randomness in generating keys for solutions and project the search space of solutions from keys space.

9.4 Summary of experiments

We first test the convergence ability of fitness for five meta-heuristics and then study the daily benefits gained by different algorithms with the parking data of one week. By increasing the number of officers from 15 to 50, all the approaches can progressively improve their result of getting parking fines. Two-sample *t* test had been conducted to further evaluate the statistical differences of algorithms in issuing parking tickets on overstaying cars. The result indicates that CS and GA using LERK for solution representation outperform the baselines and other heuristic approaches in terms of collecting parking fines and the travelling distance per fine. In other words, LERK-CS and LERK-GA have a much better convergence ability than others. Compared with two greedy algorithms, the meta-heuristics mentioned in this paper could provide a more effective search for optimal solutions to allocate violation nodes to idle officers.

Next, we examine the stability of the implemented algorithms in collecting parking fines on different days of a week with 20 or 30 officers for patrolling. The figures show that LERK-GA and LERK-CS have the best result on all the days with either 20 or 30 officers. LERK-PSO can work better than Breaks-PSO and TCX-GA in this comparison, but its performance becomes worse when the number of people is declined from 30 to 20. Two PSOs are not as good as LERK-CS and LERK-GA because of some drawbacks. The first one is that a superior solution for each particle is probably found at the beginning of searching. Second, the differences between neighbour individuals between the generations are small, which causes the algorithm to get trapped in the local optimum.

Finally, we analyse and compare the effectiveness of different algorithms in assisting officers to manage parking violations. Compared with the other methods, LERK-CS and LERK-GA have the best capability to assign violation bays to officers and catch the infringed cars. In addition, TCX-GA and two PSOs can work more efficiently than the baseline approaches. However, there is a little change in the average success rate of the attempts to issue tickets, although the average assignment rate and capture rate are increased proportionally with an increasing number of officers.

10 Discussion and limitation

In this section, we will discuss the limitations of the current model and the solutions for the MTOP, as well as outline the future plans for improvement.

Aiming at having a straightforward and fair comparison, we assumed the walking speed of all the officers to be the same in the simulation system, which is difficult to achieve in the real world. Therefore, we plan to study the effects on the overall performance of the algorithms by considering different walking speeds for different officers.

Secondly, the balance in the tasks assignment for officers is not taken into account in this study. Imbalanced workloads for parking officers might affect the completion of patrolling missions and deteriorate the final outcome of parking management. This can also be investigated in our future research.

Thirdly, there is no constraint on the travel time consumed by an officer to visit the next parking bay in violation. When the parking area for patrolling is huge, a long travelling distance for officers might be undesirable. This might provide us with an opportunity to learn by dividing the entire CBD area into smaller portions and arranging multiple officers in each portion of area, as well as apply an constraint on the maximum travel time when assigning a parking bay to an officer.

11 Conclusions

In this paper, we develop the model of MTOP and apply population-based meta-heuristics to solve it, with a new solution encoding method (LERK), new fitness function and local optimization approach. Based on the newly proposed components, CS, GA and PSO are deployed to solve the issue and compare them with another state-of-the-art GA and PSO using different encoding methods and search operations briefed in the literature. To verify the model and the optimization algorithms, we built a simulation system and test it with real-world car-parking data recorded from on-street parking slots. In addition, several evaluation metrics are employed to examine the ability and effectiveness of different algorithms in guiding officers to issue parking tickets on infringed vehicles. The experimental results show that the proposed LERK-CS and LERK-GA have superior performance over the other algorithms when using 15–50 officers for patrolling.

In future, we intend to utilize clustering techniques to separate a large parking area into relatively small regions, and each has multiple officers working towards parking management. Meanwhile, more advantageous algorithms could be adopted to recommend optimal paths for offices, with considering the balance of task assignments. In addition, as the algorithms using the LERK in this study are feasible and effective in solving the MTOP, we might also consider using them for other similar stochastic problems, such as the multiple postman problem and the vehicle scheduling issue.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Akçay Y, Li H, Xu SH (2007) Greedy algorithm for the general multidimensional knapsack problem. Ann Oper Res 150(1):17
2. Arain QA, Memon H, Memon I, Memon MH, Shaikh RA, Mangi FA (2017) Intelligent travel information platform based on location base services to predict user travel behavior from user-generated gps traces. Int J Comput Appl 39(3):155–168
3. Ayala D, Wolfson O, Dasgupta B, Lin J, Xu B (2018) Spatio-temporal matching for urban transportation applications. ACM Trans Spat Algorithms Syst (TSAS) 3(4):11
4. Bean JC (1994) Genetic algorithms and random keys for sequencing and optimization. ORSA J Comput 6(2):154–160
5. Beheshti Z, Shamsuddin SMH (2013) A review of population-based meta-heuristic algorithms. Int J Adv Soft Comput Appl 5(1):1–35
6. Benson JP, O'Donovan T, O'Sullivan P, Roedig U, Sreenan C, Barton J, Murphy A, O'Flynn B (2006) Car-park management using wireless sensor networks. In: Proceedings 2006 31st IEEE conference on local computer networks. IEEE, pp 588–595
7. Bonyadi MR, Azghadi MR, Shah-Hosseini H (2008) Population-based optimization algorithms for solving the travelling salesman problem. In: Traveling salesman problem. IntechOpen
8. Burke EK, Cowling PI, Keuthen R (2001) Effective local and guided variable neighbourhood search methods for the asymmetric travelling salesman problem. In: Workshops on applications of evolutionary computation. Springer, Berlin, pp 203–212
9. Carlton WB, Barnes JW (1996) Solving the traveling-salesman problem with time windows using tabu search. IIE Trans 28(8):617–629
10. Carrabs F, Cordeau JF, Laporte G (2007) Variable neighborhood search for the pickup and delivery traveling salesman problem with lifo loading. INFORMS J Comput 19(4):618–632
11. Carter AE, Ragsdale CT (2006) A new approach to solving the multiple traveling salesperson problem using genetic algorithms. Eur J Oper Res 175(1):246–257
12. Chatterjee S, Carrera C, Lynch LA (1996) Genetic algorithms and traveling salesman problems. Eur J Oper Res 93(3):490–510
13. Dorigo M, Birattari M (2011) Ant colony optimization. In: Encyclopedia of machine learning. Springer, Berlin, pp 36–39
14. Ezugwu AES, Adewumi AO (2017) Discrete symbiotic organisms search algorithm for travelling salesman problem. Exp Syst Appl 87:70–78
15. Fiechter CN (1994) A parallel tabu search algorithm for large traveling salesman problems. Discrete Appl Math 51(3):243–267
16. Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35
17. Gendreau M, Potvin JY (2005) Metaheuristics in combinatorial optimization. Ann Oper Res 140(1):189–213
18. Geng X, Chen Z, Yang W, Shi D, Zhao K (2011) Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. Appl Soft Comput 11(4):3680–3689
19. Glover F, Laguna M (1998) Tabu search. In: Handbook of combinatorial optimization. Springer, Berlin, pp 2093–2229
20. Google: Google matrix api. https://developers.google.com/maps/documentation/distance-matrix/intro. Accessed 11 Aug 2018
21. Guo Q, Wolfson O (2018) Probabilistic spatio-temporal resource search. GeoInformatica 22(1):75–103
22. Hemmelmayr VC, Doerner KF, Hartl RF (2009) A variable neighborhood search heuristic for periodic routing problems. Eur J Oper Res 195(3):791–802
23. Jiang Zb, Yang Q (2016) A discrete fruit fly optimization algorithm for the traveling salesman problem. PLoS ONE 11(11):e0165804
24. Junjie P, Dingwei W (2006) An ant colony optimization algorithm for multiple travelling salesman problem. In: First international conference on innovative computing, information and control, 2006. ICICIC'06, vol 1. IEEE, pp 210–213
25. Kennedy J (2010) Particle swarm optimization. Encyclopedia of machine learning. pp 760–766
26. Kıran MS, İşcan H, Gündüz M (2013) The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem. Neural Comput Appl 23(1):9–21
27. Malek M, Guruswamy M, Pandya M, Owens H (1989) Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. Ann Oper Res 21(1):59–84
28. Marinakis Y, Marinaki M (2010) A hybrid multi-swarm particle swarm optimization algorithm for the probabilistic traveling salesman problem. Comput Oper Res 37(3):432–442
29. Melbourne CC. Car parking data. https://data.melbourne.vic.gov.au/Transport-Movement/On-street-Car-Parking-Sensor-Data-2016/dj7e-rdx9. Accessed 11 Aug 2018
30. Melbourne CC. On-street parking data. https://www.melbourne.vic.gov.au/about-council/governance-transparency/open-data/Pages/on-street-parking-data.aspx. Accessed 11 Aug 2018
31. Memon I, Chen L, Majid A, Lv M, Hussain I, Chen G (2015) Travel recommendation using geo-tagged photos in social media for tourist. Wireless Pers Commun 80(4):1347–1362
32. Meng T, Pan QK (2017) An improved fruit fly optimization algorithm for solving the multidimensional knapsack problem. Appl Soft Comput 50:79–93
33. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27(4):1053–1073
34. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67

35. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
36. Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100
37. Mohamad A, Zain AM, Bazin NEN, Udin A (2013) Cuckoo search algorithm for optimization problems-a literature review. Appl Mech Mater 421:502–506
38. Moon C, Kim J, Choi G, Seo Y (2002) An efficient genetic algorithm for the traveling salesman problem with precedence constraints. Eur J Oper Res 140(3):606–617
39. Osaba E, Del Ser J, Sadollah A, Bilbao MN, Camacho D (2018) A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem. Appl Soft Comput 71:277–290
40. Osaba E, Yang XS, Diaz F, Lopez-Garcia P, Carballedo R (2016) An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. Eng Appl Artif Intell 48:59–71
41. Ouaarab A, Ahiod B, Yang XS (2014) Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput Appl 24(7–8):1659–1669
42. Ouaarab A, Ahiod B, Yang XS (2015) Random-key cuckoo search for the travelling salesman problem. Soft Comput 19(4):1099–1106
43. Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. J Comput Phys 226(2):1830–1844
44. Ribas I, Companys R, Tort-Martorell X (2011) An iterated greedy algorithm for the flowshop scheduling problem with blocking. Omega 39(3):293–301
45. Shao W, Salim FD, Gu T, Dinh NT, Chan J (2018) Traveling officer problem: Managing car parking violations efficiently using sensor data. IEEE Internet Things J 5(2):802–810
46. Shao W, Salim FD, Song A, Bouguettaya A (2016) Clustering big spatiotemporal-interval data. IEEE Trans Big Data 2(3):190–203
47. Shao W, Zhang Y, Guo B, Qin K, Chan J, Salim FD (2018) Parking availability prediction with long short term memory model. In: International conference on green, pervasive, and cloud computing. Springer, Berlin, pp 124–137
48. Shi Y, Eberhart RC (2001) Fuzzy adaptive particle swarm optimization. In: Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546), vol 1. IEEE, pp 101–106
49. Snyder LV, Daskin MS (2006) A random-key genetic algorithm for the generalized traveling salesman problem. Eur J Oper Res 174(1):38–53
50. Song CH, Lee K, Lee WD (2003) Extended simulated annealing for augmented TSP and multi-salesmen TSP. In: Proceedings of the international joint conference on neural networks, 2003, vol 3. IEEE, pp 2340–2343
51. Wang KP, Huang L, Zhou CG, Pang W (2003) Particle swarm optimization for traveling salesman problem. In: International conference on machine learning and cybernetics, 2003, vol 3. IEEE, pp 1583–1585
52. Wang L, Zheng Xl, Wang Sy (2013) A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem. Knowl-Based Syst 48:17–23
53. Wong LP, Low MYH, Chong CS (2008) A bee colony optimization algorithm for traveling salesman problem. In: Second Asia international conference on modeling & simulation, 2008. AICMS 08. IEEE, pp 818–823
54. Yang XS, Deb S (2009) Cuckoo search via lévy flights. In: World congress on nature & biologically inspired computing, NaBIC 2009. IEEE, pp 210–214
55. Yuan S, Skinner B, Huang S, Liu D (2013) A new crossover approach for solving the multiple travelling salesmen problem using genetic algorithms. Eur J Oper Res 228(1):72–82
56. Zanella A, Bui N, Castellani A, Vangelista L, Zorzi M (2014) Internet of things for smart cities. IEEE Internet Things J 1(1):22–32
57. Zhou H, Song M, Pedrycz W (2018) A comparative study of improved ga and pso in solving multiple traveling salesmen problem. Appl Soft Comput 64:564–580