Hindawi
Scientific Programming
Volume 2023, Article ID 7588595, 12 pages
https://doi.org/10.1155/2023/7588595

Research Article

Path Planning Algorithm for the Multiple Depot Vehicle Routing Problem Based on Parallel Clustering

Xue Han

Xuzhou Open University, Xuzhou 221000, China

Correspondence should be addressed to Xue Han; snow@xzit.edu.cn

Received 25 October 2022; Revised 16 March 2023; Accepted 31 March 2023; Published 25 April 2023

Academic Editor: Debo Cheng

It is necessary to study the problem of vehicle routing in multidistribution centers to improve the speed, time, and cost thereof. It is preferable to use as few vehicles as possible to complete the delivery of goods and minimize the total mileage. With the development of artificial intelligence technology, machine learning is usually used to solve the problem of k shortest paths in multiple distribution centers. User needs are constantly changing; the iterative convergence speed of traditional machine learning methods is low and cannot meet the requirements of path planning in a big-data environment. Aiming at difficult problems in multipath planning, the parallel characteristics of traditional machine learning algorithms are fully exploited; k-means clustering and simulated annealing algorithms are improved through the distributed computing; and the multiple depot vehicle routing problem clustering analysis and path planning under the framework of Spark distributed computing are proposed. Through 30 simulation experiments on the TSPLIB dataset, the optimal solution is obtained with a 100% accuracy rate in problem solving. Experimental comparison and analysis show that the algorithm proposed in this article can solve the problem at least twice as fast as other parallel algorithms. This finding verifies that this method can effectively solve the multipath planning problem, thus greatly improving the quality and efficiency of path planning in large-scale logistics.

1. Introduction

The transportation problem of multiple distribution centers is to find the optimal transportation path for all vehicles of a logistics company to meet the needs of a large number of customers. This is essential in an urban logistics system, where each vehicle starts from a warehouse. The vehicle routing problem (VRP) is the core problem of distribution route optimization, serving customers along the optimal path and returning to the warehouse. With the continuous expansion of logistics companies, logistics distribution is no longer limited to a single-distribution center. Research on the multiple depot vehicle routing problem (MDVRP) model has practical significance, and research on the multidistribution center vehicle routing problem is insufficient. Because of the large scale of the problem, it is difficult to design an algorithm that produces a better solution in less time.

The combination optimization problem for MDVRP has been accurately solved. Laporte and Nobert [1] proposed a branch and bound algorithm, and Laporte and Nobert [2] proposed a branch and bound algorithm for the asymmetric MDVRP. With the rise of artificial intelligence, heuristic algorithms have begun to solve the MDVRP problem. Wren and Holliday [3] proposed a heuristic-improved MDVRP algorithm based on a scanning idea; Bruce and Edward [4] used a heuristic algorithm of classification before solution to complete the construction, expanding the scale to 360 customer points; and Renaud et al. [5] designed a tabu search MDVRP algorithm with capacity and driving distance limits. The application of machine learning and deep learning in path planning has been deepened. de Oliveira et al. [6] used clustering to decompose the MDVRP into multiple single-distribution center problems; Bezerra et al. [7] used clustering to generate an initial solution and improved it by a variable neighborhood search; Bello et al. [8] used a deep reinforcement learning to directly solve combinatorial optimization problems; Wang and Chen [9] proposed a solution model based on multiagent deep reinforcement learning; and Orozco-Rosas et al. [10] proposed a machine learning path planning algorithm and a path planning solution model based on membrane evolution in known or unknown environments [11] and designed a hybrid path planning algorithm [12].

There are two deficiencies in the vehicle routing problem of multiple distribution centers as follows:

(1) Intelligent algorithms can ensure solution quality in path planning, but the time complexity and consumption are high

(2) The scale of path planning is relatively small, and the largest number of customer points is about 300, which does not conform to the reality of large logistics centers

The main contributions for solving the MDVRP problem and improving the solution quality and convergence speed are as follows:

(1) An optimized cuckoo search k-means (OCS-k-means) clustering algorithm is proposed to optimize k-means and improve the convergence speed of the clustering algorithm. For the practical requirements of large logistics centers that serve a large number of customers, the OCS-k-means algorithm is parallelized using the Spark's RDD model to further improve the convergence speed of clustering.

(2) A simulated annealing algorithm of large neighborhood search (SALNS) is proposed to improve the convergence speed of path planning within the cluster. At the same time, to adapt to the performance of path planning in a big-data environment and ensure the quality of the global optimal solution, a distributed parallel algorithm is used to accelerate the optimization of SALNS.

2. Methodology

For the MDVRP problem of large logistics centers, we must quickly realize data segmentation under multiple constraint environments, divide the entire road traffic network into k subsets [13], and form clusters of the distribution points. Clustering analysis can greatly reduce the scale of problem solving, and intelligent algorithms can easily find the optimal path on small-scale datasets. The clustering analysis of large datasets is not effective. To improve performance and accuracy, a distributed cluster environment is required for quick data segmentation. It is necessary to analyze the common k-means clustering algorithm, integrate a genetic algorithm to optimize all centroids, improve the quality of the algorithm, and implement parallelization through Spark distributed computing.

2.1. MDVRP Mathematical Model. MDVRP is an optimization problem with constraints. Assume a set of m distribution centers, $D = \{d1, d2, \ldots, dm\}$; corresponding vehicle set $V = \{V_1, V_2, V_3, \ldots, V_m\}$; and customer set $C = \{c_1, c_2, c_3, \ldots, c_n\}$.

The formula for minimizing the total transportation path length is as follows:

$$\min \sum_{i\in D\cup C} \sum_{j\in D\cup C} \sum_{k\in V} \mathrm{di\,st}_{ij} \bullet x_{ijk}, \tag{1}$$

where $\mathrm{di\,st}_{ij} = |x_i - x_j| + |y_i - y_j|$ is the Manhattan distance from point i (x_i, y_i) to j (x_j, y_j) and the decision variable is x_{ijk}, which is expressed as follows:

$$x_{ijk} = \begin{cases} 1, & \text{Vehicle } k \text{ travels from } i \text{ to } j, \\ 0, & \text{others.} \end{cases} \tag{2}$$

The distribution center constraint is that all vehicles depart from the distribution center and return after completing the task. The constraint function is given as follows:

$$\sum_{i\in D} \sum_{j\in C} \sum_{k\in V} x_{ijk} = \sum_{j\in D} \sum_{i\in C} \sum_{k\in V} x_{jik} = |D|. \tag{3}$$

The customer point service constraint is that all customer nodes have and only have a unique vehicle service, with constraint function,

$$\sum_{j\in D\cup C} \sum_{k\in V} x_{ijk} = 1, \forall i \in V. \tag{4}$$

The number of vehicles in and out of all customer nodes should be equal, i.e.,

$$\sum_{i,j\in C} x_{ijk} = \sum_{i,j\in C} x_{jik}, \forall k \in V. \tag{5}$$

2.2. K-Means Clustering. The idea of k-means clustering is to randomly assign k initial centroids as the cluster center, randomly divide all objects in the dataset into k clusters, calculate the distance between each data object and the cluster centroid, and assign these data objects to the cluster with the nearest centroid. We take the mean of all points in the cluster as the new cluster center and update this until the current and previous cluster centers are within a given threshold, to obtain the final cluster center [14]. The MDVRP solution process is defined according to the original k-means clustering algorithm, as shown in Figure 1.

The main problem of the original k-means is that the center of the cluster may shift the distribution center in the process of continuous iteration, which may lead to there being no distribution center in the cluster. At the same time, the original k-means has a poor global search ability and easily falls into local optimal solutions.

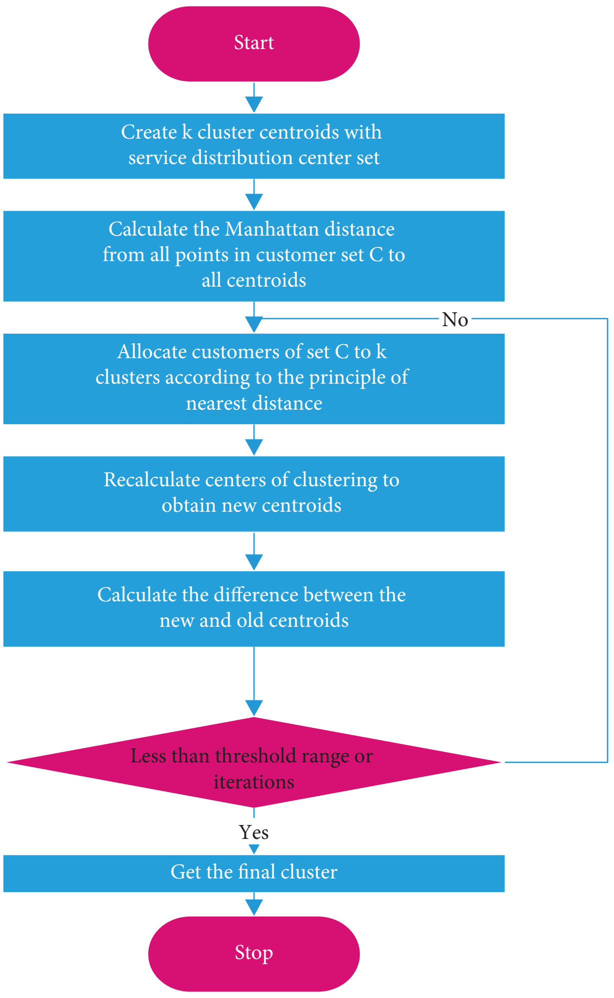
Start
Create k cluster centroids with service distribution center set
Calculate the Manhattan distance from all points in customer set C to all centroids
Allocate customers of set C to k clusters according to the principle of nearest distance
Recalculate centers of clustering to obtain new centroids
Calculate the difference between the new and old centroids
Less than threshold range or iterations
No
Yes
Get the final cluster
Stop

Figure 1: Original k-means algorithm flow in the multiple depot vehicle routing problem (MDVRP).

2.3. Optimization of K-Means Algorithm. To solve the problem of centroid shift, it is necessary to ensure that the distribution point belongs to one and only one cluster when updating, i.e., for distribution center d_i and cluster C_i, $d_i \in C_i$ is always guaranteed.

To solve the problem that k-means easily falls into local optima, it has often been proposed to combine it with a bionic algorithm, such as simulated annealing [15], the leapfrog algorithm [16], tabu search [17], particle swarm optimization [18], the ant colony algorithm [19], and cuckoo search [20]. We use an improved cuckoo search [21] to optimize k-means clustering. To introduce the cuckoo algorithm into the k-means algorithm, it is necessary to improve the cuckoo search algorithm by introducing an adaptive step size related to the optimal nest location distance, as follows:

$$A_i = \mathrm{step}_{\min} + \left(\mathrm{step}_{\max} - \mathrm{step}_{\min}\right) \times d_i, \tag{6}$$

where $\mathrm{step}_{\max}$ represents the maximum value of step size, $\mathrm{step}_{\min}$ represents the minimum value of step size, and d_i represents the distance adjustment factor, which can be calculated as follows:

$$\frac{\left(\left|\mathrm{nest}\_(i) - \mathrm{nest}\_(\mathrm{best})\right|\right)}{d\_(\max)}, \tag{7}$$

where nest_i represents the position of the ith nest, $\mathrm{nest}_{\mathrm{best}}$ represents the current optimal nest position, and $d_{\max}$ represents the maximum distance between the optimal nest and all other nests.

The improved cuckoo search optimizes the k-means clustering algorithm as follows:

Step 1: initialize the distribution center set D, customer set C, discovery probability Pa, fitness boundaries $F_{\max}$ and $F_{\min}$, discovery probability boundaries $\mathrm{Pa}_{\max}$ and $\mathrm{Pa}_{\min}$, step size boundaries $\mathrm{step}_{\max}$ and $\mathrm{step}_{\min}$, and maximum iterations T.

Step 2: set D as cluster centroids, i.e., the initial positions of the bird nest $X = D$, and initialize the objective function $f(x_0)$, where $x_0 = \{x_1^0, x_2^0, \ldots, x_m^0\}^T$.

Step 3: classify the clusters according to K-means to obtain a new cluster. Calculate the fitness value $F(i)$, and record the current optimal functional value $f\,(x_{\mathrm{i}}^t)$ and optimal nest position x_{i}^t of the population, where t is the current number of iterations and b is the number of classifications, i.e., $b = \{1, 2, 3, \ldots, m\}$.

Step 4: update new nest positions $x_i^t = \{x_1^t, x_2^t, \ldots, x_m^t\}^T$ according to equation (8), with adaptive step $a_i = \mathrm{step}_{\min} + (\mathrm{step}_{\max} - \mathrm{step}_{\min}) \times d_i, d_i = (F_{\max} - F(i))/(F_{\max} - F_{\min})$, $\mathrm{L}(\lambda) \sim \mu = t^{-\lambda}$ [22]:

$$x_i^t = x_i^{t-1} + a_i \oplus L(\lambda). \tag{8}$$

Step 5: calculate the updated objective function $f(x_i)$ of new nest positions, compare x_i^t and x_i^{t-1}, and save the best nest location $x_i = \{x_1^i, x_2^i, \ldots, x_m^i\}^T$.

Step 6: compare the adaptive discovery probability Pa with a random number $r \in [0, 1]$. If $r \leq \mathrm{Pa}$, nest positions remain unchanged; otherwise, update nest positions, where [23]

$$\mathrm{Pa} = \frac{\left((T - t) \times \left(\mathrm{Pa}_{\max} - \mathrm{Pa}_{\min}\right)\right)}{T + \mathrm{Pa}_{\min}}. \tag{9}$$

Step 7: judge whether the maximum number of iterations T is reached, or if the distance between the current and previous optimal nest positions is within the specified threshold. If the end condition is met, the algorithm ends, and the global optimal nest positions and optimal solution are output. Otherwise, return to step 3.

2.4. Optimized K-Means Algorithm Parallelization. The k-means clustering algorithm in Figure 1 performs clustering calculations and centroid updates according to the principle of the shortest Manhattan distance and allocates all elements of $D \cup C$ to $k = |D|$ clusters [24]. All sample points x_i ($i = 1, 2, 3, \ldots, m + n$) in dataset $D \cup C$ are independently calculated based on the Manhattan distance to each centroid, and the sampling points are assigned to the nearest centroid. There is no dependency between them, which is suitable for parallel processing. Similarly, the recalculated centroid positions of each cluster can only depend on the sample points in the same cluster, i.e., there is no dependency between clusters, which is also suitable for parallelization. Therefore, k-means clustering can be parallelized through the Spark RDD model.

Parallelization of clustering can effectively solve the problem of slow convergence in path planning. To improve the convergence speed, applications can be decomposed into many small tasks and distributed to multiple computers for processing, i.e., a superlarge dataset can be decomposed into subsets according to a standard. Each computer performs clustering analysis on a subset. The parallelization of OCS-k-means clustering based on the Spark framework is as follows:

Step 1: initialize the distribution center set D, customer set C, discovery probability Pa, fitness boundaries $F_{\max}$ and $F_{\min}$, discovery probability boundaries $\mathrm{Pa}_{\max}$ and $\mathrm{Pa}_{\min}$, step size boundaries $\mathrm{step}_{\max}$ and $\mathrm{step}_{\min}$, and maximum iteration number T.

Step 2: store the dataset $D\cup C$ in the HDFS distributed file system, and establish the RDD model of the Spark computing framework.

Step 3: take D as the cluster centroid, i.e., the initial position of the bird nest $X=D$, and initialize the objective function $f(x_0)$, where $x_0=\{x_1^0, x_2^0, \ldots, x_m^0\}^T$. At each node of the cluster, use map-related operators to format the dataset.

Step 4: use the parallel computing of the Spark RDD operator framework, use Map and Reduce to cluster the operators and partition the sample data according to the principle of the k-means minimum distance, obtain a new cluster, calculate the fitness value $F(i)$, and record the current optimal function value $f(x_\mathrm{i}^t)$ and the optimal bird nest position x_i^t of the population.

Step 5: update nest positions according to $x_i^t = x_i^{t-1} + a_i \oplus L(\lambda)$, to obtain new nest positions $x_i^t = \{x_1^t, x_2^t, \ldots, x_m^t\}^T$.

Step 6: calculate the objective function $f(x_i)$ of the updated nest location, compare x_i^t and x_i^{t-1}, and save the best nest locations $x_i=\{x_1^i, x_2^i, \ldots, x_m^i\}^T$.

Step 7: compare the adaptive discovery probability Pa with random number $r\in[0, 1]$. If $r\le \mathrm{Pa}$, the nest positions remain unchanged; otherwise, update the nest positions.

Step 8: use relevant RDD operators such as Map and Reduce to cluster the newly generated bird nest according to the principle of minimum distance from the cluster centroid, calculate the new nest location objective function, compare it with the current optimal bird nest, retain the optimal solution, and obtain the optimal nest location for this iteration.

Step 9: judge the iteration termination conditions. If the conditions are met, use the reduce aggregation operator of RDD to perform aggregation operations, output the globally optimal bird's nest position and optimal solution, and terminate the algorithm. If the conditions are not met, return to step 4.

2.5. Solving the TSP Problem. The parallel OCS-k-means clustering algorithm forms $|D|$ clusters from customer set $D\cup C$, each cluster contains one and only one service center, and each service center provides services for the customers in the cluster. In this way, the scale of the problem can be quickly reduced to $1/|D|$. All customer service logistics services in a cluster must be completed. Then, the problem becomes the traveling salesman problem (TSP), which is to find the shortest path under the constraints in Section 2.1 and solve formula (1) in each cluster, i.e., the minimum value to traverse all vertices of the optimal solution $f(x_i)$, once and only once in each cluster. The TSP is a widely studied NP problem in path planning, but the complexity of completing the optimal path by the exact solution is still very high. At present, the heuristic algorithm is a popular TSP algorithm [3–5]. Common heuristic algorithms include the genetic algorithm, ant colony algorithm, particle swarm optimization, and simulated annealing.

The SA algorithm, an improved random mountain climbing algorithm, is adopted in this article. SA can accept new points that are not as good as or are even worse than the current point with a certain route change, and it can jump out of a local optimum to realize the global optimal solution. Zhang and Qi [25] used a partial random generation of an initial path and a partial nearest insertion method to improve the genetic algorithm and reverse the evolution of the operator, but the population was set too large. He et al. [26] proposed a Metropolis criterion, which improved the fitness function and cross-mutation operator, to effectively avoid the local optima. Yu et al. [27] introduced a partial nearest neighbor method to generate the initial population for the "premature" convergence problem of a genetic algorithm, which greatly accelerated convergence and improved the search quality. Li and Su [28] used the simulated annealing intelligent optimization algorithm to solve the TSP problem and found that the selection of initial temperature is the key to the effect of the simulated annealing intelligent algorithm. Xiao-Ping and Qiu-Qiu [29] introduced an improved method of reverse operation to jump out of local optima and enter the next search space. The previous references provide a technical basis for this article.

2.6. Optimization of SA Algorithm. The simulated annealing (SA) algorithm is relatively mature as a solution for small- and medium-sized TSP problems. As the scale of problem solving increases, the difficulty of solving the problem increases exponentially, and the convergence speed is insufficient. We introduce the large-scale neighborhood search (LNS) algorithm [30], as shown in Figure 2, to optimize the simulated annealing algorithm, improve the convergence speed, and prevent local optimal solutions.

We destroy the initial solution space to remove several random nodes, e.g., $c_5, c_x, \ldots, c_6$, remove the selected points from the initial solution space, and order the remaining points according to the order of the initial solution to form a destructive solution space. We use the removed nodes to repair the failure solution space in turn by reinserting them into any position in the failure solution space and calculating

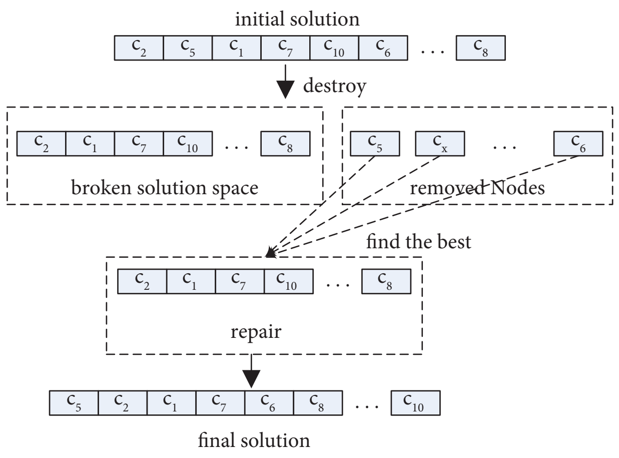

Figure 2: Working principle of LNS.

the appropriate value for inserting these different positions. We select the appropriate value effect as the insertion result until all removed nodes are inserted in the damaged solution space, and the final solution space is obtained after the repair. The SA algorithm using the LNS method (SALNS) is executed as follows:

Step 1: initialize iteration number N, exchange probability P, temperature T_0, cooling coefficient a, iteration number It, floating interval gapIter, clustering Manhattan distance matrix dist, and other parameters.

Step 2:use a greedy algorithm to generate the initial solution S_{current} and calculate the fitness as follows:

$$\text{fit}\left(S_{\text{current}}\right) = \sum_{i=1}^{n=1} \text{di st}_{i,i+1} + \text{di st}_{1,n}. \tag{10}$$

Step 3: record the optimal path $S_{\text{best}} = S_{\text{current}}$ and optimal distance fit $(S_{\text{best}}) = \text{fit}\ (S_{\text{current}})$.

Step 4: according to the roulette strategy, randomly select two cities from the current path with probability Pa to exchange, insert a new city, or use the LNS method to generate a new path.

Step 5: calculate the fitness fit (S_{new}) of the new path. If fit $(S_{\text{new}}) <$ fit (S_{current}), then $S_{\text{current}} = S_{\text{new}}$; otherwise, update according to the Metropolis criterion, i.e., calculate as follows:

$$p = e^{-T\left(\text{fit}\left(S_{\text{nes}}\right) - fir\left(S_{\text{current}}\right)/fit\left(S_{\text{current}}\right)\right)}. \tag{11}$$

Moreover, generate a random number $r \in [0, 1]$. If $r < p$, then update $S_{\text{current}} = S_{\text{new}}$.

Step 6: if fit $(S_{\text{current}}) \leq$ fit (S_{best}), then $S_{\text{best}} = S_{\text{current},}$ fit $(S_{\text{best}}) =$ fit (S_{current}).

Step 7: update the temperature according to initial temperature $T_0 = T_0 * a$ and $T = T_0 * (1 + \cos(t * \text{pi/gapIter}))$, where t is the number of iterations.

Step 8: randomly select two elements, c_i and c_j. If dist$_{i}$, $j +$ dist$_{i+1}$, $j + i <$ dist$_{i,i+1}$ + dist$_{j,j+1}$, remove paths $(c_i, c_i + 1)$, (c_j, c_{j+1}), and add (c_i, c_j), (c_{i+1}, c_{j+1}).

Step 9: judge whether the end condition is reached. If not, return to step 4 to continue execution, then output result S_{best}.

2.7. SALNS Parallelization. The distributed parallel algorithm can be used to accelerate optimization. From Figure 1, it can be seen that the initial (current) solution can be broken into multiple sets of broken solutions and removed nodes. Each set can be assigned to a computing node to complete an iteration and form multiple final solutions. The optimal solution among these can be selected as the current solution of the next iteration. However, the destruction process must be executed serially, with low acceleration performance and possible path-crossing problems. Each time an optimal solution is directly transferred to all distributed computing nodes, all nodes use their current solution to cross the optimal solution [31]. As shown in Figure 3, a segment of the current optimal solution and the path of the current solution are cross reorganized (i.e.., ensuring that the position and order of the segment remain unchanged, while adding other nodes to the optimal solution according to the order of the current solution) to generate a new solution. Each computing node performs an LNS operation on the new solution to obtain a new current solution. The simple parallel SALNS (SPSALNS) process is shown in Figure 4.

3. Results and Discussion

To verify the efficiency and correctness of the algorithm proposed in this article, all the algorithms used were experimentally analyzed. The correctness and convergence speed of the OCS-k-means parallel algorithm were verified using the classical UCI machine learning dataset, and the SPSALNS simulation experiment on the TSPLIB dataset proved that the method proposed in this article can solve the MDVRP problem. To verify the correctness of the algorithm, the entire processes of the parallel OCS-k-means and SPSALNS algorithms were simulated.

3.1. Analysis of Improved Cuckoo Search Clustering Algorithm. We conducted 30 simulation experiments on four validation datasets: Iris, Wine, 4k2_far, and Balance-scale from the classic UCI machine learning dataset [32]. Table 1 shows the comparison results of six algorithms, namely, the traditional k-means clustering algorithm, firefly algorithm k-means (FA-k-means), particle swarm optimization k-means (PSO-k-means), simulated annealing clustering algorithm k-means (SACA-k-means), algorithm of bee colony k-means (ABC-k-means), and optimized cuckoo search k-means (OCS-k-means). The accuracies of the algorithms are shown in Table 1, and the number of iterations required for algorithm convergence is shown in Table 2 [33].

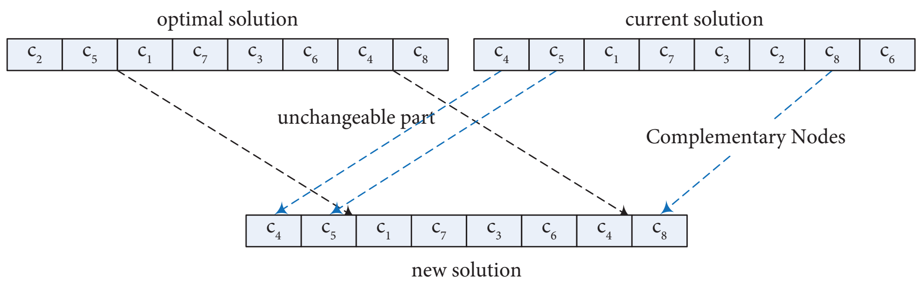

Figure 3: OX cross principle.

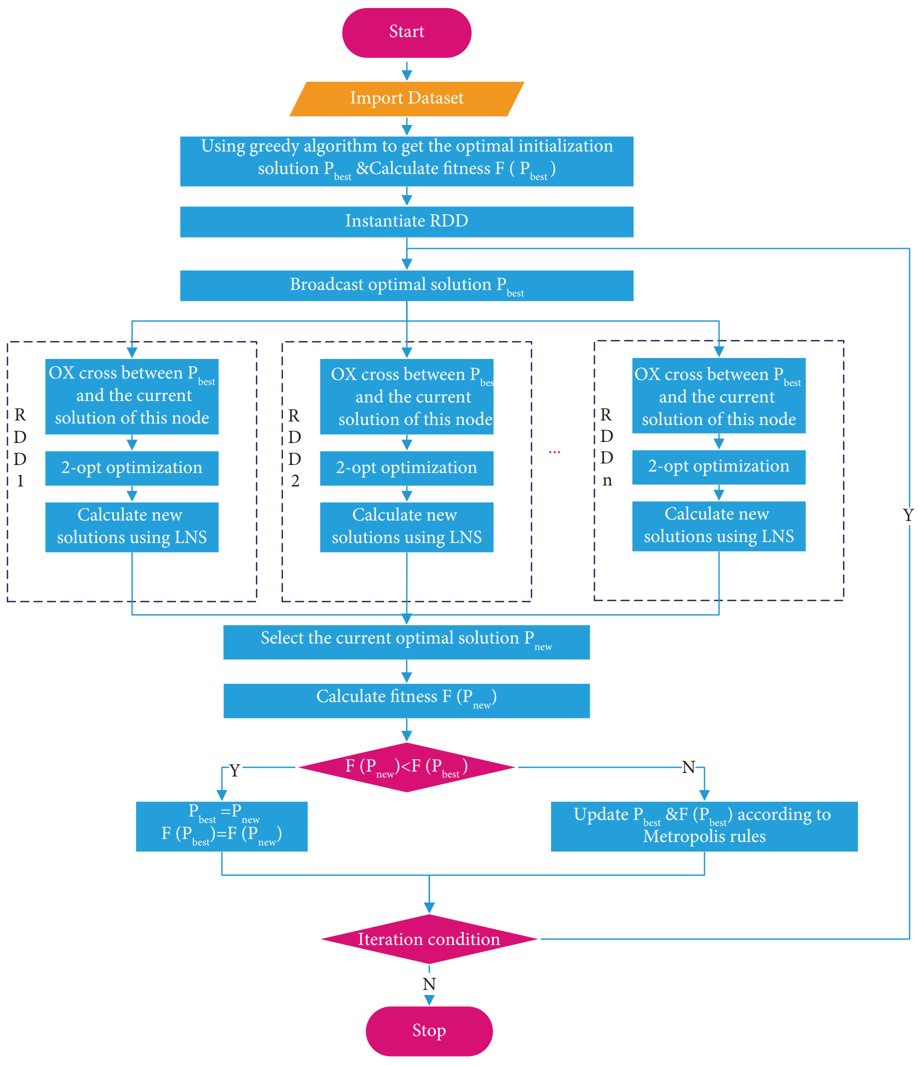
Start
Import Dataset
Using greedy algorithm to get the optimal initialization solution P best &Calculate fitness F (P best)
Instantiate RDD
Broadcast optimal solution P best
RDD1
RDD2
RDDn
OX cross between P best and the current solution of this node
2-opt optimization
Calculate new solutions using LNS
Select the current optimal solution P new
Calculate fitness F (P new)
F (P new)<F (P best)
Y
N
P best =P new F (P best)=F (P new)
Update P best &F (P best) according to Metropolis rules
Iteration condition
Stop

Figure 4: SPSALNS algorithm.

Table 1: Accuracy statistics of clustering algorithms.

Algorithm	Iris (%)	Wine (%)	4k2_far (%)	Balance-scale (%)
k-means	70.00	56.67	46.67	63.33
FA-k-means	76.67	76.67	53.33	76.67
PSO-k-means	73.33	66.67	60.00	80.00
SACA-k-means	83.33	70.00	56.67	83.33
ABC-k-means	83.33	73.33	53.33	76.67
OCS-k-means	86.67	83.33	56.67	90.00

Table 2: Statistics of iterations of the different clustering algorithms.

Algorithm	Iris	Wine	4k2_far	Balance-scale
k-means	19.8	41.2	34.7	39.7
FA-k-means	9.7	31.8	28.9	34.7
PSO-k-means	46.9	21.2	32.6	36.4
SACA-k-means	13.1	29.6	25.7	31.2
ABC-k-means	12.7	27.6	28.6	27.7
OCS-k-means	8.9	19.5	22.3	25.3

Through comparison, it is found that OCS-k-means is superior to the traditional k-means algorithm and other commonly used improved k-means algorithms in terms of accuracy and iteration time.

The number of iterations is easily affected by the number of features, as well as by the hyperparameter k (the number of clusters). With the increase in k, the number of iterations will continue to be smaller, but this does not imply that the model effect will continue to improve.

For this reason, a nonparametric statistical analysis is carried out using silhouette coefficients. This method is suitable for the cluster analysis of data without real labels, and the method is used to evaluate the impact of different algorithms or different operation modes of algorithms on clustering results based on the original data. The silhouette coefficient of a single sample is calculated as follows:

$$\begin{aligned} s(i) &= \frac{a(i) - b(i)}{\max(a(i), b(i))}, \\ s &= \frac{\sum_{i=1}^{k} s(i)}{k}, \end{aligned} \tag{12}$$

where $a(i)$ is the similarity within the measurement group and $b(i)$ is the similarity between the measurement groups. $s(i)$ ranges from −1 to 1. The larger the value, the higher the intragroup coincidence and the farther the intergroup distance; that is, the larger the silhouette coefficient value, the better the clustering effect.

According to the silhouette coefficient equation, the statistics of various clustering algorithms are obtained, and the resulting silhouette coefficients are shown in Table 3. It can be seen that OCS-k-means is the best in Iris, Wine, and Balance-scale, and that 4k2_far is the second best.

Table 3: Statistical analysis results of silhouette coefficients of different clustering algorithms.

Algorithm	Iris	Wine	4k2_far	Balance-scale
k-means	0.55	0.48	0.42	0.51
FA-k-means	0.58	0.59	0.46	0.59
PSO-k-means	0.57	0.53	0.49	0.61
SACA-k-means	0.62	0.55	0.48	0.62
ABC-k-means	0.62	0.57	0.46	0.59
OCS-k-means	0.64	0.62	0.48	0.66

3.2. Efficiency Analysis of Parallel OCS-K-Means. To verify the clustering efficiency of the parallel OCS-k-means clustering algorithm, Dataset 1–Dataset 4 were designed, with respective sample sizes of 10^5, 10^6, 10^7, and 10^8, and data dimensions of 3, 5, 7, and 20. The original and parallel k-means clustering algorithms and the parallel OCS-k-means clustering algorithm were each simulated 30 times, and the running time (in seconds) was compared and analyzed.

The average convergence times of the original k-means, parallel k-means, and parallel OCS-k-means on Dataset 1 were 13.57 s, 33.71 s, and 39.11 s, and the average convergence times on Dataset 2 were 21.06 s, 57.95 s, and 60.11 s, respectively. Because the number of data samples in Dataset 1 and Dataset 2 test sets was small and because of the long execution time of Map and Reduce, the convergence performance of the parallel k-means and parallel OCS-k-means was worse than that of the original k-means. When the amount of sample data were large, the execution time of Map and Reduce in the parallel algorithm accounted for a small proportion of the total running time. At this time, the average convergence time of the parallel clustering algorithm was better than the serial algorithm. To verify the efficiency improvement of the parallel OCS-k-means algorithm with large-scale data, a statistical test was carried out on the 30 experimental results of the three algorithms in Dataset 3 and Dataset 4 with a confidence level of 95.0%. The original k-means and parallel k-means algorithms were tested using an F-test to obtain the two-sample analysis of variance. The F-test showed that P value ≥ 0.05, so the sample has an equal variance. The two-sample ANOVA t-test can be used for the difference test. The original k-means and parallel k-means test results are shown in Table 4.

Similarly, an F-test and t-test were performed on the parallel k-means and parallel OCS-k-means algorithms, and the test results are shown in Table 5.

As shown in Tables 4 and 5, the t-test exhibits $P < 0.01$, indicating that the efficiency of the parallel k-means algorithm was significantly higher than that of the original k-means algorithm in large datasets, while the efficiency of the OCS-k-means algorithm was approximately twice than that of the parallel k-means algorithm. Thus, the parallel OCS-k-means clustering algorithm is more suitable for processing large-scale datasets.

3.3. Convergence and Parallelism Analysis of Improved Simulated Annealing Algorithm. Figures 5 and 6 show the comparison of convergence time between SALNS and PSO, GA, and SA on the datasets Att48 and Kro A100, indicating that SALNS convergence time was shorter.

Ten simulation experiments were carried out on the TSPLIB datasets Oliver30, Att48, Eil51, Eil76, KroA100, and Ch130. Table 6 compares the $F(P_{\text{best}})$ mean values solved by

Table 4: Test results of original k-means and parallel k-means.

Parameters	Dataset 3		Dataset 4	
	Original k-means	Parallel k-means	Original k-means	Parallel k-means
Average	419.516	246.452	8487.354	4103.102
Variance	25.579	18.610	8846.707	6702.702
F	1.37446754		1.319872	
$P\ (F \leq f)$ one-tailed	0.19841217		0.229742	
t-Stat	142.596496		192.5747	
$P\ (T \leq t)$ one-tailed	$7.6409E-76$		$2.14E-83$	
t one-tailed critical	1.67155276		1.671553	

Table 5: Test results of parallel k-means and parallel OCS-k-means.

Parameters	Dataset 3		Dataset 4	
	Parallel k-means	Parallel OCS-k-means	Parallel k-means	Parallel OCS-k-means
Average	246.452	173.093	4103.102	2861.74
Variance	18.610	13.134	6702.702	5110.535
F	1.416947		1.311546	
$P\ (F \leq f)$ one-tailed	0.176702		0.234876	
t-Stat	71.31483		62.5568	
$P\ (T \leq t)$ one-tailed	$1.71E-58$		$3.09E-55$	
t one-tailed critical	1.671553		1.671553	

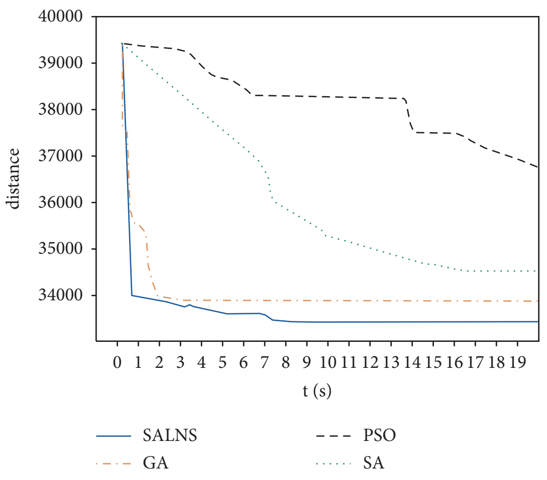

Figure 5: Comparison of Att48 data convergence curves.

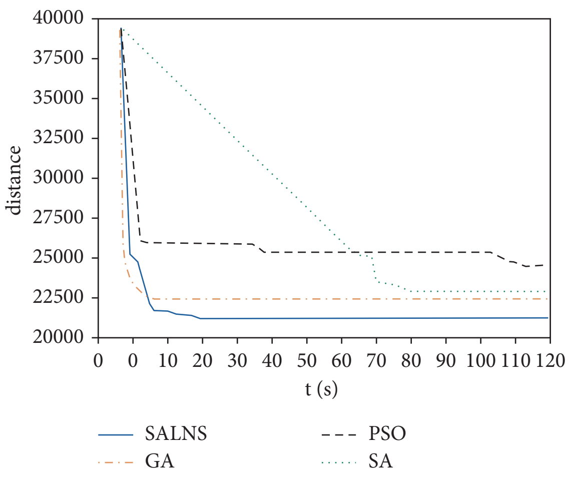

Figure 6: Comparison of Kro A100 data convergence curves.

SPSALNS, SQACS [34], and SSA [35], from which it can be seen that SPSALNS can find the real optimal solution for all situations in the dataset.

In SPSALNS, each computing node only accepts the current optimal solution broadcast by the system and intersects with the current solution of the node to form a new solution. At this time, each computing node is fully parallel, which greatly improve the parallelism. The results of 30 experiments using SSA, SQACS, and SPSALNS on the KroA100 dataset were statistically tested. The P value of the F-test was less than 0.05, and the two-sample heteroscedasticity hypothesis in the t-test was used. The statistical analysis results are shown in Table 7.

Through statistical analysis, we found the P value from the t-test to be less than 0.01; thus, the difference is statistically significant, HO is rejected, and H1 is accepted. It can

Table 6: Comparison of $F(P_{\text{best}})$ mean values.

Dataset	SPSALNS	SQACS	SSA	Real solution
Oliver30	420	421	427	420
Att48	33522	33583	34103	33522
Eil51	426	427	451	426
Eil76	538	543	582	538
KroA100	21282	21540	22695	21282
Ch130	6100	6370	6568	6100

be seen that the SPSALNS operation time (33.564 ± 4.110 s) is less than the SQACS operation time (93.673 ± 10.832 s) and SSA operation time (109.788 ± 15.523 s). Therefore, its computation time is significantly better than those of the two algorithms SQACS [34] and SSA [35]. The average running time of these three algorithms in the TSPLIB dataset, i.e.,

Table 7: Statistical analysis results of SPSALNS, SSA, and SQACS.

Parameters	Comparison between SPSALNS and SSA		Comparison between SPSALNS and SQACS	
	SSA	SPSALNS	SQACS	SPSALNS
Average	109.788	33.564	93.673	33.564
Variance	240.984	16.893	117.353	16.893
Standard deviation	15.523	4.110	10.832	4.110
F	14.26465		6.946536	
$P\ (F \leq f)$ one-tailed	$1.15E-10$		$6.48E-07$	
t-Stat	25.99848		28.41517	
$P\ (T \leq t)$ one-tailed	$7.39E-24$		$4.94E-27$	
t one-tailed critical	1.69236		1.687094	

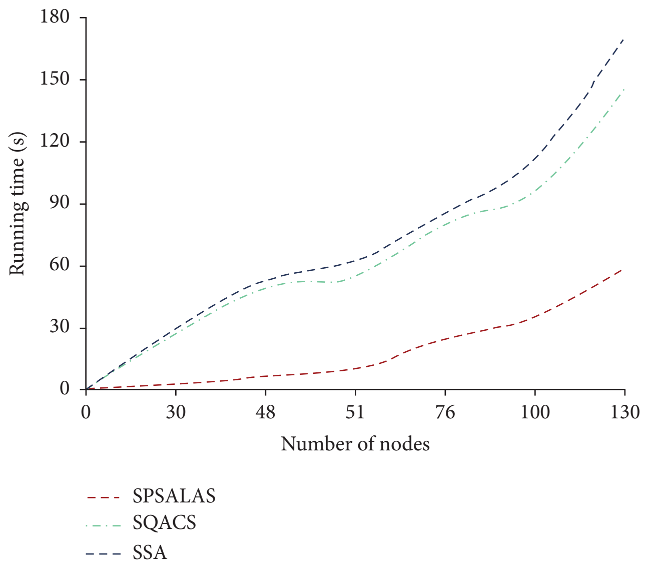

Figure 7: Comparison of convergence rates.

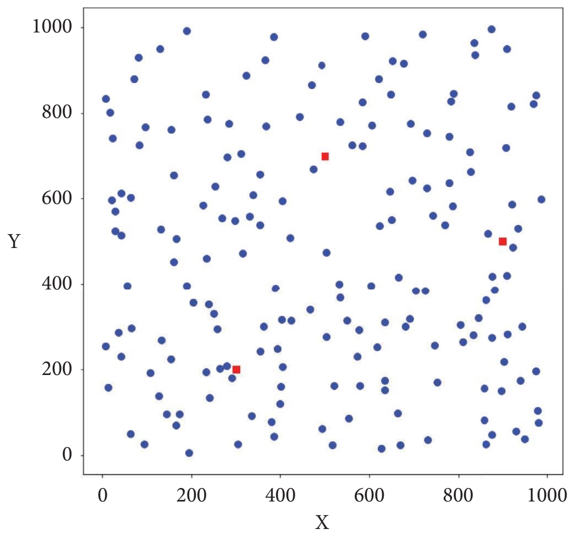

Figure 8: Distribution of service center and customer point $D \cup C$ set: red squares indicate service centers and blue dots represent customers.

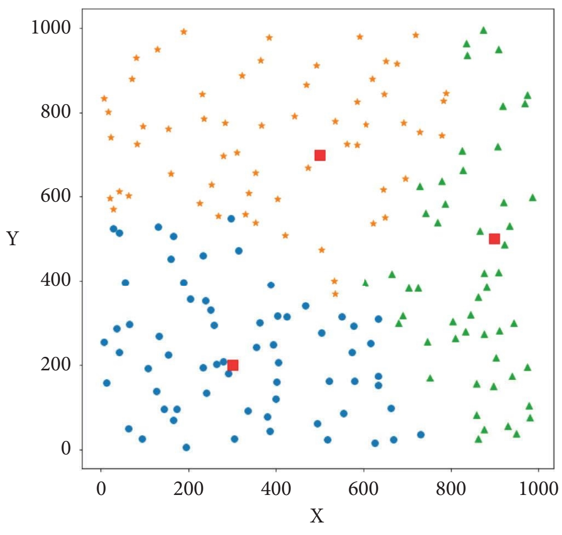

Figure 9: Three clusters formed by the OCS-k-means algorithm.

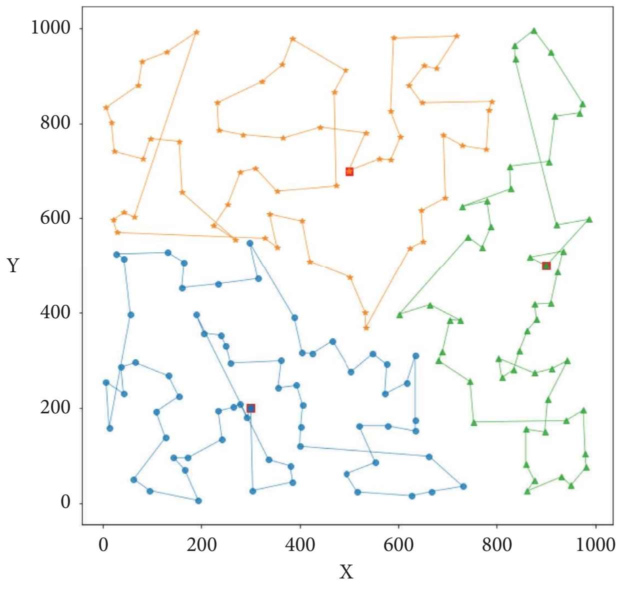

Figure 10: Path planning using the SPSALNS algorithm.

algorithm efficiency, is shown in Figure 7, which indicates that the efficiency is improved by at least a factor of two.

3.4. Experimental Simulation Results. For the MDVRP problem, we constructed the dataset shown in Figure 8, including the service center set D (rectangles) and service customer set C (dots). The OCS-k-means algorithm was used to decompose $|D|$ clusters according to the constraint formula in Section 2.1 to form one cluster for each service center, which greatly reduces the scale of problem solving, as shown in Figure 9.

For each cluster, as shown in Figure 9, the SPSALNS algorithm is used internally to solve for the optimal service path, with calculation results as shown in Figure 10.

3.5. Analysis. The distributed parallel OCS-k-means clustering algorithm can quickly cluster and partition a large number of service nodes in the MDVRP problem. The more data are processed, the more obvious the advantages. It can quickly divide a large amount of data into smaller problem sizes. In each cluster, the Spark computing framework based on an improved simulated annealing algorithm is used for parallel computing, which can increase the computing speed and optimize the quality of the solution. For the same data, the strategy adopted in this article was compared with the original PSO solution, and the convergence speed of the parallel PSO solution is shown in Figure 11. It can be found that the solution quality is effectively improved, and convergence is faster.

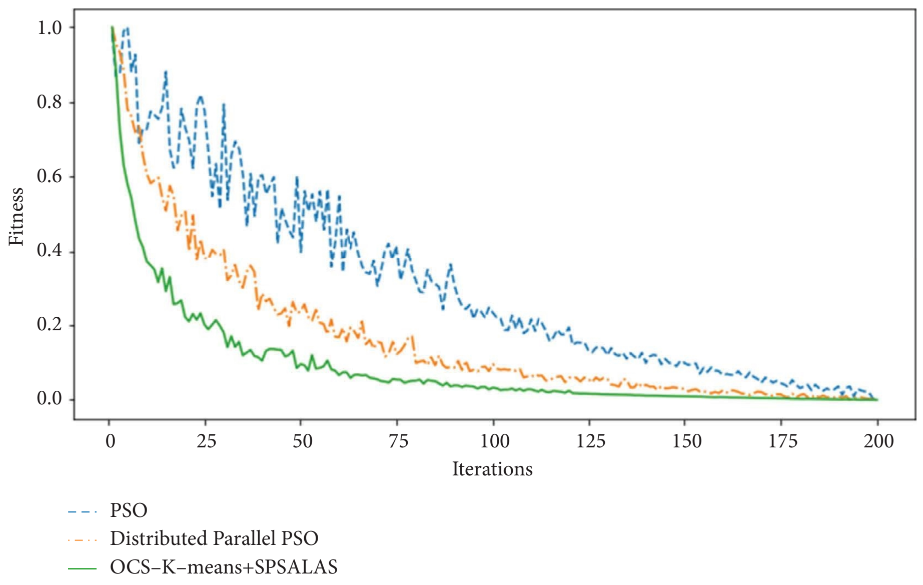

Figure 11: Comparison of fitness between the OCS-k-means + SPSALLAS algorithm and other algorithms.

4. Conclusion

Large and superlarge logistics centers cannot avoid the NP-hard problem of MDVRP. Traditional solution algorithms are inefficient and perform poorly. Intelligent algorithms such as heuristic and deep learning methods converge slowly when large amounts of data are involved; hence, they cannot adapt to vehicle path planning, and it is difficult to meet the needs of the rapid development of logistics. To improve the efficiency of path planning, the frameworks of the k-means and simulated annealing algorithms were analyzed, their parallel and serial computing parts were sorted and optimized, and the speed was improved through the Spark distributed computing. It was found that the convergence rate of the distributed parallel algorithm was at least twice than that of the traditional intelligent algorithm when the data volume was large; the greater the data volume, the better the convergence performance.

This method is designed for the actual needs of large logistics centers; however, the following limitations exist in the actual scenario:

(1) For applications that serve a small number of customers, parallel processing cannot effectively improve the convergence speed, but the cost of parallel processing decreases

(2) In the case of frequent changes in customer demand, frequent clustering and optimal path solving are required, which is inefficient

MDVRP has several possible directions for future work. User requirements may change, so we might consider the real-time dynamic path planning problem of vehicle-borne multiagents in complex dynamic environment. In addition, reinforcement learning, deep learning, and neural networks can be used to solve complex path planning problems with more constraints. Finally, the three-dimensional path planning problem in the scenario of multiple types of vehicles (e.g., UAVs, cars, and ships) can be applied such as to disaster relief, exploration, and medical aid.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author declares that there are no conflicts of interest.

Acknowledgments

This study was sponsored by Xuzhou Open University. The author thanks LetPub (https://www.letpub.com) for its linguistic assistance during the preparation of this article.

References

[1] G. Laporte and Y. Nobert, “Optimal solutions to capacitated multidepot vehicle routing problem,” *Congressus Numerantium*, vol. 44, no. 3, pp. 283–292, 1984.

[2] G. Laporte and Y. Nobert, “Solving a family of mufti-depot vehicle routing andlocation-routing problems,” *Transportation Science*, vol. 22, no. 3, pp. 161–172, 1988.

[3] A. Wren and A. Holliday, “Computer scheduling of vehicles from one or more depots to a number of delivery points,” *Journal of the Operational Research Society*, vol. 23, no. 3, pp. 333–344, 1972.

[4] L. Bruce and W. Edward, “A new heuristic for the mufti-depot vehicle routing problem that improves upon best-known solutions[J],” *Math Manage Sci*, vol. 13, no. 3-4, pp. 371–406, 1993.

[5] J. Renaud, G. Laporte, and F. F. Boctor, "A tabu search heuristic for the multi-depot vehicle routing problem," *Computers and Operations Research*, vol. 23, no. 3, pp. 229–235, 1996.

[6] F. B. de Oliveira, R. Enayatifar, H. J. Sadaei, F. G. Guimarães, and J. Y. Potvin, "A cooperative coevolutionary algorithm for the multi-depot vehicle routing problem," *Expert Systems with Applications*, vol. 43, pp. 117–130, 2016.

[7] S. N. Bezerra, S. R. de Souza, and M. J. F. Souza, "A GVNS algorithm for solving the multi-depot vehicle routing problem," *Electronic Notes in Discrete Mathematics*, vol. 66, pp. 167–174, 2018.

[8] I. Bello, H. Pham, and Q. V. Le, "Neural combinatorial optimization with reinforcement learning," 2016, https://arxiv.org/abs/1611.09940.

[9] W. Wang and H. Chen, "Deep reinforcement learning for multi-depot vehicle routing problem," *Control and Decision*, vol. 37, no. 8, pp. 2101–2109, 2022.

[10] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, and A. Cuesta-Infante, "Mobile robot path planning using a QAPF learning algorithm for known and unknown environments," *IEEE Access*, vol. 10, no. 8, pp. 84648–84663, 2022.

[11] U. Orozco-Rosas, O. Montiel, and R. Sep´ulveda, "Mobile robot path planning using membrane evolutionary artificial potential fifield," *Applied Soft Computing*, vol. 1, 2019.

[12] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots," *IEEE Access*, vol. 7, no. 11, pp. 156787–156803, 2019.

[13] Y. Liu, R. Song, R. Bucknall, and X. Zhang, "Intelligent multi-task allocation and planning for multiple unmanned surface vehicles (USVs) using self-organising maps and fast marching method," *Information Sciences*, vol. 496, pp. 180–197, 2019.

[14] R. Jia, Y. Guan, and L. I. Ya-long, "Parallel k-means clustering algorithm based on MapReduce model," *Computer Engineeringand Design*, vol. 35, no. 2, pp. 657–660, 2014.

[15] L. I. Jiang-wei and X. U. Lun-hui, "Research on path planning based on simulated annealing algorithm and neu -ral network algorithm," *Automation and Instrumentation*, vol. 32, no. 11, pp. 6–9, 2017.

[16] P. Zhang, L. I. Ting, H. Zhang, and T. Yan, "Modeling research for EHV power transformer based on hybrid artificial fish swarm and shuffled frog leaping algorithm," *Insulators and Surge Arresters*, vol. 6, no. 12, pp. 7–19, 2018.

[17] H. Qunru and S. Pan, "Vehicle routing optimization of logistics system based on tabu search algorithm," *Science Technology and Engineering*, vol. 19, no. 34, pp. 401–407, 2019.

[18] C. Niu and B. Wu, "Research on text clustering algorithm based on improved particle swarm optimization and K-means," *Journal of Lanzhou University of Arts and Science (Natural Sciences)*, vol. 33, no. 4, pp. 44–47, 2019.

[19] L. Zhang, L. He, and D. Wu, "Research on improved ant colony algorithm in path planning," *Manufacturing Automation*, vol. 42, no. 2, pp. 55–59, 2020.

[20] X. Yu, *Improved Parallel K-means Clustering Algorithm based on Cuckoo Search*, Chongqing University, Chongqing, China, 2017.

[21] X. Yang and S. Deb, "Recent advances and applications," *Neural Computing and Applications*, vol. 24, no. 1, pp. 672–679, 2014.

[22] B. Wang and X. Yu, "Parallel K-means clustering algorithm based on adaptive cuckoo search," *Application Research of Computers*, vol. 35, no. 3, pp. 675–679, 2018.

[23] R. Wang, X. Cui, and Y. Li, "Self-adaptive adjustment of cuckoo search K-means clustering algorithm," *Application Research of Computers*, vol. 35, no. 12, pp. 3593–3597, 2018.

[24] J. Du, L. Duan, W. Duan, and Q. Bu, "Parallel implementation of improved K-means algorithm based on Spark," *Application Research of Computers*, vol. 37, no. 2, pp. 434–436, 2020.

[25] Y. Zhang and Y. Qi, "An improved genetic simulated annealing algorithm to solve TSP," *Intelligent Computer and Applications*, vol. 7, no. 3, pp. 52–54, 2017.

[26] Q. He, Y. Wu, and X. U. Tong-Wei, "Application of improved genetic simulated annealing algorithm in TSP optimization," *Control and Decision*, vol. 33, no. 2, pp. 220–225, 2018.

[27] Z. Yu, J. Zhuang, and X. Zhai, "Path planning research based on improved TSP model and simulated annealing algorithm," *Art Technology*, vol. 31, no. 2, pp. 56-57, 2018.

[28] Y. Li and Y. Su, "Application of simulated annealing intelligent algorithm in tsp," *Industrial and Science Tribune*, vol. 16, no. 3, pp. 68–70, 2017.

[29] X. U. Xiao-Ping and Z. H. U. Qiu-Qiu, "Improved simulated annealing algorithm forof solving tsp," *Computer Systems and Applications*, vol. 24, no. 12, pp. 152–156, 2015.

[30] L. Nan, Y. Chen, and Z. Zhang, "Improved adaptive large neighborhood search algorithm for mixed fleet routing problem of dynamic demands," *Application Research of Computers*, vol. 38, no. 10, pp. 2926–2934, 2021.

[31] S. Liu, *Spark-based Parallel Simulated Annealing Algorithm and its Application in TSP Problem*, North Minzu University, Yinchuan, China, 2022.

[32] K. S. Tushar, C. P. Sushree, and M. R. Kabat, "An adaptive cuckoo search based algorithm for placement of relay nodes in wireless body area network," *Journal of King Saud University Computer and Information Sciences*, vol. 34, pp. 1–12, 2019.

[33] Q. Fu, *Optimization and Application of K-Means Clustering Algorithm Based on Spark Framework*, Chengdu University of Technology, Chengdu, China, 2020.

[34] X. Xu, Y. Tang, and F. Wang, "Artificial cooperative search algorithm for solving traveling salesman problems," *Journal of Computer Applications*, vol. 46, no. 7, pp. 1838–1843, 2000.

[35] J. Xue and B. Shen, "A novel swarm intelligence optimization approach: sparrow search algorithm," *Systems Science and Control Engineering*, vol. 8, no. 1, pp. 22–34, 2020.