2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Locality-Sensitive Hashing Based Multiobjective Memetic Algorithm for Dynamic Pickup and Delivery Problems

Fangxiao Wang, Yuan Gao, and Zexuan Zhu

***Abstract*—This paper proposes a locality-sensitive hashing based multiobjective memetic algorithm namely LSH-MOMA for solving pickup and delivery problems with dynamic requests (DPDPs for short). Particularly, LSH-MOMA is designed to find the solution route of a DPDP by optimizing objectives namely workload and route length in an evolutionary manner. In each generation of LSH-MOMA, locality-sensitive hashing based rectification and local search are imposed to repair and refine the individual candidate routes. LSH-MOMA is evaluated on three simulated DPDPs of different scales and the experimental results demonstrate the efficiency of the method.**

I. Introduction

PICKUP and delivery problem (PDP) is a vehicle routing problem ubiquitous in logistic industry. To solve a PDP is to find an optimal route, in terms of length, workload and labor cost etc., to serve a serial of nodes associated with pickup and/or delivery requests while obeying some constraints on the transportation capacity [1-6, 17]. A PDP is static when all service requests are known in advance of constructing the routes. In real-world logistic cases, many service requests occur in real-time, i.e., the input data keep changing in the course of routing. The resultant problem is called Dynamic Pickup and Delivery Problem (DPDP) [1, 14]. PDPs have been proved to be NP-hard and DPDPs represent harder cases.

Evolutionary algorithms (EAs) such as ACO [7, 8], GA [15], SA [16], and TS [16] have been widely used to solve DPDPs due to their merit of obtaining satisfactory results in tractable time cost [10]. Most of the existing methods were designed to solve DPDPs that consists of nodes raising only pickup or delivery requests, i.e., nodes raising both pickup and delivery requests are seldom considered [2, 5, 9, 11-13, 21]. Moreover, most of the methods take into account only one objective, normally the route length, when optimizing the service route. The abovementioned methods may not work well in real-world applications where the customer nodes normally request both pickup and delivery services, and multiple objectives are usually considered.

In this paper, we propose a locality-sensitive hashing based multiobjective memetic algorithm (LSH-MOMA) to solve DPDPs by optimizing the service workload and route length, simultaneously. Particularly, the one-to-many-to-one (1-M-1) DPDPs are studied. In a 1-M-1 DPDP, a vehicle starts from the depot and follows an initially scheduled route to serve static requests. The route is dynamically changed to respond to new requests raised by customer nodes, so that the two objectives, i.e., workload and route length are optimized and the transportation capacity is not violated. The vehicle should be destined to the same depot. The static requests must be served in the solution route while the dynamic ones could be selectively responded. The candidate route solutions are optimized with a multiobjective memetic algorithm, where locality-sensitive hashing (LSH) [18] based rectification operation and local search are used to repair and improve the candidate solutions in each generation. LSH is a method used to locate the nearest neighbors of a node efficiently in route rectification and refinement. The proposed LSH-MOMA is tested on three 1-M-1 DPDPs with different scale of customer nodes. The experimental results demonstrate that the method is efficient in identifying reasonable route solutions.

The rest of the paper is organized as follows. Section II describes the definition of DPDP. Section III introduces the LSH method. Section IV provides the details of the proposed LSH-MOMA. Section V presents the experimental results on simulated data. Finally, Section VI concludes this study.

II. Problem Definition

This section presents the definition of 1-M-1 DPDP. A 1-M-1 DPDP is to find an optimal route of a vehicle from a depot to go thought all customer nodes and end up at the same depot. The vehicle carries pre-requested commodities before leaving the depot, and responds to the dynamic requests while serving in the route. The optimality of the route is defined in terms of workload and distance in this study.

To formulate the problem, we assume the vehicle is located in a square region $\boldsymbol{R}=R_x \times R_y$ and the pickup and delivery requests occur stochastically. The depot, customer node, vehicle capacity constraint, service route, route length, and workload are defined as follows:

Depot: the depot D is the source and destination of the route. The vehicle must start and finish at D.

This work was supported in part by the National Natural Science Foundation of China, under grants 61171125 and 61205092, the Guangdong Foundation of Outstanding Young Teachers in Higher Education Institutions, under grant Yq2013141, Guangdong Natural Science Foundation under grant S2012010009545, the Scientific Research Foundation for the Returned Overseas Chinese Scholars, Ministry of Education of China, under grand 20111568, and Shenzhen Scientific Research and Development Funding Program under grants JCYJ20130329115450637, KQC201108300045A, and ZYC201105170243A.

F. Wang, Y. Gao, and Z. Zhu are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China. All correspondence should be addressed to Dr. Zexuan Zhu (Email: zhuzx@szu.edu.cn, Tel.: +86 755 2673 2704).

Customer Node: a customer node is denoted by $n_i=(x_i, y_i)$, where x_i and y_i together indicate the position of n_i in $\boldsymbol{R}$. The node is also associated with d_i and p_i indicating the amounts of delivery and pickup commodities, respectively. Let N be the set of all customer nodes, $N^s=\{n_0^s, n_1^s,\ldots, n_k^s\}$ be the set of nodes with static requests known before the vehicle set off, and $N^d=\{n_0^d, n_1^d,\ldots, n_m^d\}$ be the set of notes with dynamic requests raised during the course of serving, i.e., $N=N^s \bigcup N^d$. The nodes in N^s must be served whereas the ones in N^d are selectively responded.

Vehicle Capacity Constraint: the vehicle has a maximum capacity of C. This capacity limit is a hard constraint and a route is considered infeasible if this constraint is violated.

Route: a route of the vehicle is represented as $r=<n_1, n_2, \ldots, n_l>$, where $n_1=n_l=D$.

Route Length: in this study, the route length $L(r)$ is simply calculated by summing up the Euclidean distance of every two adjacent nodes in route r, i.e.,

$$L(r)=\sum_{i=1}^{l-1} d(n_i, n_{i+1})$$

where $d(n_i,n_{i+1})$ indicates the Euclidean distance between customer nodes n_i and n_{i+1}.

Workload: the workload of a scheduled route is the summarization of pickup and delivery commodities over all nodes in the route:

$$W(r)=\sum_{i=1}^{l}(p_i+d_i)$$

III. LOCALITY-SENSITIVE HASHING

To present the details of LSH-MOMA, we begin with the introduction of LSH, which is a key component in route rectification and local search.

LSH is a method of performing probabilistic dimension reduction of high-dimensional data [18]. The basic idea is to hash customer nodes according to their locations so that nearby nodes are mapped to the same value with high probability. To search the nearest neighbors of a node, what we need is to search within those nodes of the same hash values as the queried node. As such, LSH is able to quickly identify the 'approximate' nearest neighbors of a customer node by avoiding exhaustive search. Since exact nearest neighbors are not necessarily needed in this study, LSH provides great advantage in terms of time cost.

The key issue of LSH is the design of the hash function. In this study, we divide the search region $\boldsymbol{R}$ into $t\times t$ equal lattices as shown in Fig. 1. The hash function of a customer node is simply defined by mapping the node to the corresponding lattice that contains the node:

$$H(n_i)=\left(\left\lceil\frac{x_i\cdot t}{R_x}\right\rceil,\left\lceil\frac{y_i\cdot t}{R_y}\right\rceil\right)$$

where $\lceil x\rceil$ returns the smallest integer greater than x. For example, the red node in Fig. 1 is mapped to lattice (3,3) with the hash function. Based on this function, the search of the nearest neighbors of a node n_i can be limited in a small region, i.e., one or several lattices, avoiding searching the whole region. This mechanism is especially useful for dynamic requests, where the distance of the dynamic nodes to other nodes cannot be calculated off-line.

At the beginning of LSH, all customer nodes' hash values are calculated and stored. To find the K nearest neighbors of a node n_i, those nodes of hash values equal to $H(n_i)$, i.e., nodes mapped to the same lattice as n_i, are identified and their distances to n_i are calculated, thereby identifying the K nearest to n_i. If one lattice does not contain sufficient nodes, the search region is enlarged by involving the surrounding eight lattices, as shown in Fig. 1. The enlargement can be continued until all K nearest neighbors are found. Once the K nearest neighbors of n_i are identified, they can be archived for the future use. The procedure of LSH is summarized in Algorithm 1.

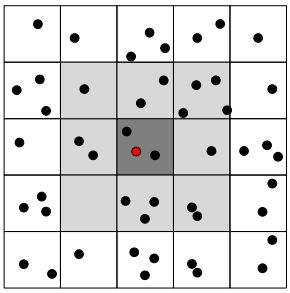


Fig 1. LSH based on grating

Algorithm 1: Procedure of LSH

INPUT: hash values of all customer nodes (calculated off-line), a queried node n_i, and K;
OUTPUT: the K nearest neighbors of n_i;

```
BEGIN
1   Set (a,b) = H(n_i), S={}, and c=0;
2   While |S| < K do
3      S_tmp = {};
4      For i=max(a-c,1) to min(a+c,t)
5         For j=max(b-c,1) to min(b+c,t)
6               Add all nodes within lattice (i, j) to S_tmp;
7         End For
8      End For
9      If | S_tmp | < K then
10        S=S⋃ S_tmp;
11     Else
12        Calculate the distances of n_i to all nodes in S_tmp;
13        Identify the K-|S| nearest nodes to n_i and add them to S;
14        Return S;
15     End If
16     c=c+1;
17  End While
END
```

IV. LSH-MOMA FOR 1-M-1 DPDPS

The workflow of LSH-MOMA for 1-M-1 DPDPs is shown in Fig 2. The service requests consist of requests from both static and dynamic nodes. The static nodes known in advance must be served whereas the dynamic ones received in the course are selectively responded.

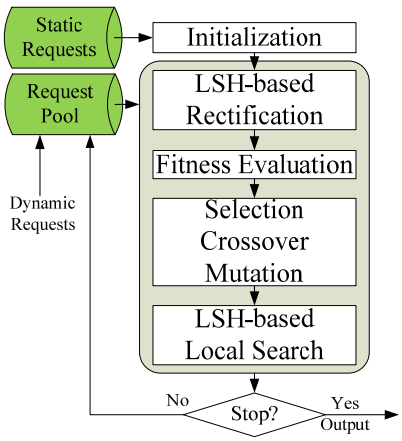


Fig 2. The workflow of LSH-MOMA

Before the vehicle leaving the depot, a route is scheduled with the static requests. Here, we assume that the commodities carried by the vehicle do not exceed 50% of the vehicle's capacity when leaving the depot, so that there has space left for serving dynamic requests. During the course of serving, the dynamic requests are received and buffered in a request pool. The pool is checked by LSH-MOMA in a fixed time window say Γ generations, and a part dynamic nodes are added to the routes within the mutation operator. In each time window, a customer node in the route is served, i.e., the served node becomes unchangeable in the route. The candidate routes are evolved with evolutionary operators including selection, crossover, and mutation. In each generation of LSH-MOMA, LSH-based rectification and local search are applied to the individuals to repair and refine the candidate solutions. The outline of LSH-MOMA is provided in Algorithm 2. Details of the components of LSH-MOMA are provided in the following subsections.

Algorithm 2: LSH-MOMA for DPDP

```
BEGIN
1   i=0;
2   Initialize the population P_i using Algorithm 3;
3   While stop criteria are not satisfied do
4     If i mod Γ == 0 do
5       Serve a customer node on each route;
6       If dynamic requests occur do
7         Update the request pool;
8       End If
9     End If
10    Generate an offspring population Q_i with selection,
      crossover, and mutation;
11    Apply LSH-based rectification on each offspring;
12    Evaluate the fitness of each offspring;
13    Apply LSH-based local search on each offspring;
14    Generate new population P_{i+1} from P_i ∪ Q_i based on
      Pareto optimality;
15    i= i + 1
16  End While
End
```

A. Chromosome Encoding Scheme

In LSH-MOMA, a variable length chromosome is used to encode a candidate route, as shown in Fig. 3. The chromosome starts and ends at the depot D, and the intermediate is a sequence of customer nodes served in the route.

$n_1=D$	n_2	n_3	...	n_i	...	$n_l=D$

Fig 3. Chromosome encoding scheme

B. Population Initialization

At the beginning of LSH-MOMA, a population of chromosomes is randomly generated to encode candidate routes consist of all static nodes. A LSH-based route rectification (introduced in Section IV.C) is applied to the chromosomes to ensure their validity. The population is evolved using a traditional multiobjective evolutionary algorithm (MOEA) for ten generations. Afterward, the population is fed to LSH-MOMA where dynamic nodes are considered. The procedure of population initialization is provided in Algorithm 3.

Algorithm 3: Population Initialization

```
BEGIN
1  Set i=0;
2  Randomly initialize the population P_i with static nodes;
3  Apply LSH-based route rectification on each individual;
4  While stop criteria are not satisfied do
5     Evaluate the fitness of each individual;
6     Generate an offspring population Q_i with selection,
      crossover, and mutation;
7     Generate new population P_{i+1} from P_i ∪ Q_i based on
      Pareto optimality;
8    i = i + 1;
9  End While
END
```

C. LSH-based Rectification

After a new route is generated by random initialization or evolutionary operation, the validity of the route should be guaranteed. The route must avoid violations of the capacity constraint. Accordingly, a LSH-based rectification is proposed here to detect and repair capacity constraint violations occurring in a route. The procedure of LSH-based rectification is provided in Algorithm 4.

D. Multiobjective Fitness Evaluation

In each generation of LSH-MOMA, the fitness of each candidate route is evaluated in terms of route length $L(r)$ and workload $W(r)$. Particularly, the algorithm simultaneously minimizes $L(r)$ and maximizes $W(r)$. It is noted that the two

Algorithm 4: LSH-based Rectification

INPUT: a candidate route $r=<n_1, n_2, \ldots, n_l>$;

BEGIN

While $W(r) > C$ **do**
  Go along r from n_1 and find out the first node n_f where the first violation occurs;
  Identify the K nearest neighbors of n_f based on LSH method;
  Randomly select a nearest neighbor node n_p of n_f;
  If $n_p \in r$ **do**
    Relocate n_f before or after n_p;
  Else
    Relocate n_f to any other position of the route;
  End if
End While

END

objectives tend to conflict with each other. A long route likely leads to larger workload. There is no single solution can optimize the two objectives at the same time. The idea of Pareto-optimality [19] is applied here to solve this multiobjective optimization problem so that multiple trade-off solutions can be obtained in a single run, from which appropriate solutions can be selected on demand.

E. Evolutionary operators

After fitness evaluation, traditional evolutionary operators including selection, crossover, and mutation are used to evolve the population. The selection is performed following NSGA-II [19] where both nondomination rank and crowding distance are considered as selection criteria. Moreover, elitism is used to retain the best individuals at each generation. After selection, the order-based crossover [16] is applied to generate offspring from the selected individuals. The new offspring individuals undergo mutation by randomly swapping two nodes or reversing a small part of the route.

To respond to dynamic requests, we also enable the mutation operation to randomly add/delete dynamic nodes with a certain probability in a fixed time window Γ, which is specified as a few generations. In other words, LSH-MOMA checks the request pool once for every Γ generations, and adds some dynamic nodes to the candidate routes in mutation, or reversely removes dynamic nodes in the route to the pool.

After crossover and mutation, the offspring routes should undergo LSH-based rectification to ensure their validity.

F. LSH-based Local Search

To improve the quality of candidate routes, an LSH-based local search is introduced to fine-tune the offspring generated by evolutionary operators. Particularly, given a route, the LSH-based local search randomly selects a node in the route, and then locally changes the position of the node or adds a nearest neighbor of the node to the route. If the resultant new route is better than the old one, it replaces the old one. The details of LSH-based local search are given in Algorithm 5.

Algorithm 5: LSH-based Local Search

INPUT: a candidate route $r=<n_1, n_2, \ldots, n_l>$;
BEGIN
  Randomly select a node n_i in r;
  Identified the K nearest neighbors of n_r based on LSH method;
  Randomly select a nearest neighbor node n_p of n_r;
  If $n_p \in r$ **do**
    Relocate n_p before or after n_i;
  Else
    Insert n_p to r before or after n_i;
  End If
  Evaluate the fitness of the new route;
  If the new route dominates the old one **do**
    Replace the old one with the new one;
  End If
END

V. Simulation Results and Analysis

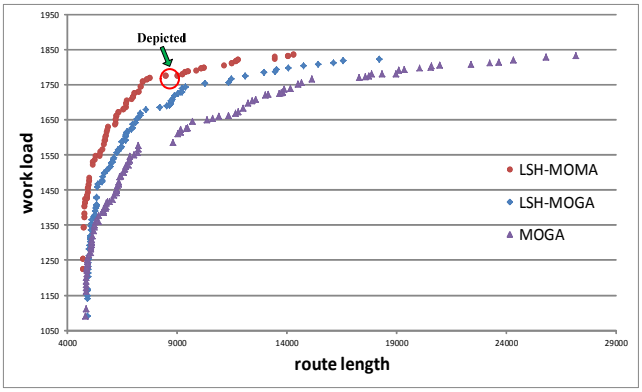
LSH-MOMA is implemented in C++ and run on a PC with Intel Pentium 4 2.4 GHz. To test the performance of LSH-MOMA, three simulated 1-M-1 DPDPs of different scales are generated with $|N^d| = 50$ and $|N^s| = 30$, 50, and 70, respectively. One new dynamic request is generated and pooled in each ten generations.

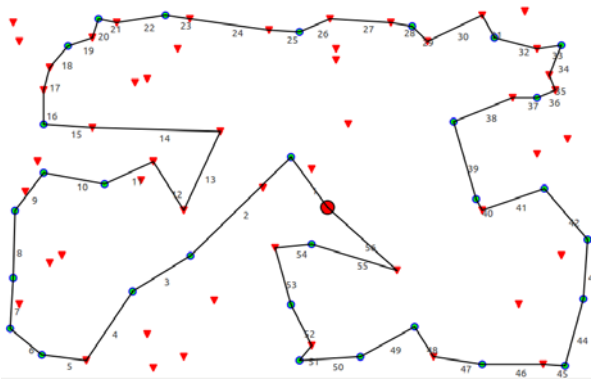
LSH-MOGA, i.e., LSH-MOMA without using the LSH-based local search, and MOGA, i.e., LSH-MOGA without using LSH are also considered in the comparison study to test the effect of LSH and LSH-based local search. LSH-MOMA, LSH-MOGA, and MOGA are run with the same parameter setting of population size = 200, crossover probability = 0.6, mutation rate = 0.09, and time window Γ= 10 generations. LSH-MOMA is terminated when the generation number exceeds 500. To ensure a fair comparison among LSH-MOMA, LSH-MOGA and MOGA on each problem, LSH-MOGA and MOGA are configured to terminate when the computational effort incurred exceeds that of the LSH-MOMA. The three algorithms are independently run for 25 times on each DPDP, and the average results are reported.

The mean values of the Pareto optimal set obtained by the three algorithms are summarized in Table 1 in terms of route length, workload, and dynamic response rate. Here, the

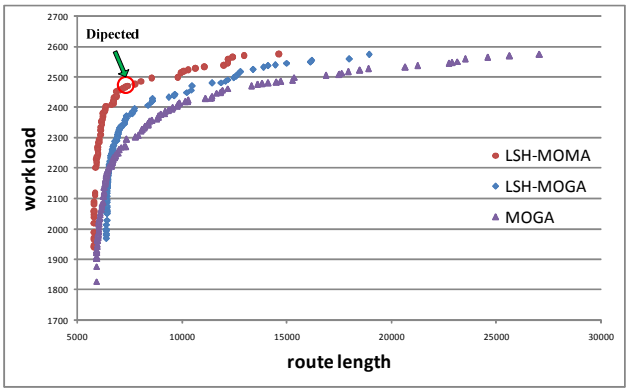
TABLE1 AVERAGE RESULTS OF THE PARETO OPTIMAL SETS OBTAINED BY LSH-MOMA, LSH-MOGA, AND MOGA IN TERMS OF ROUTE LENGTH, WORKLOAD, AND DYNAMIC RESPONSE RATE.

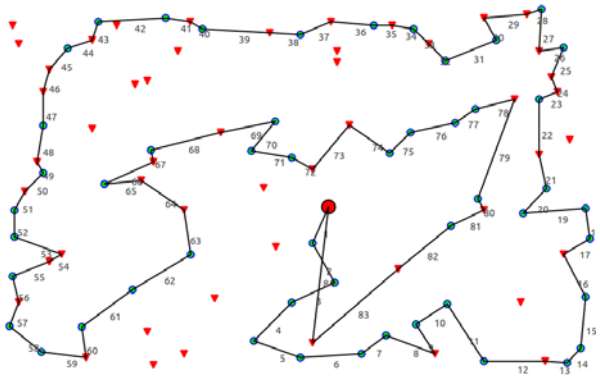
	Mean Route Length			Mean Workload			Mean Dynamic Response Rate		
	DPDP1	DPDP2	DPDP3	DPDP1	DPDP2	DPDP3	DPDP1	DPDP2	DPDP3
LSH-MOMA	**6683.29**	**6855.02**	**7419.12**	**1604.59**	2275.3	**2862.79**	**0.5616**	0.4622	**0.4888**
LSH-MOGA	7532.31	8197.51	8992.4	1540.09	**2283.98**	2805.31	0.4819	**0.4843**	0.4399
MOGA	9748.8	10100.16	11323.86	1516.865	2275.31	2825.29	0.4666	0.4627	0.4547



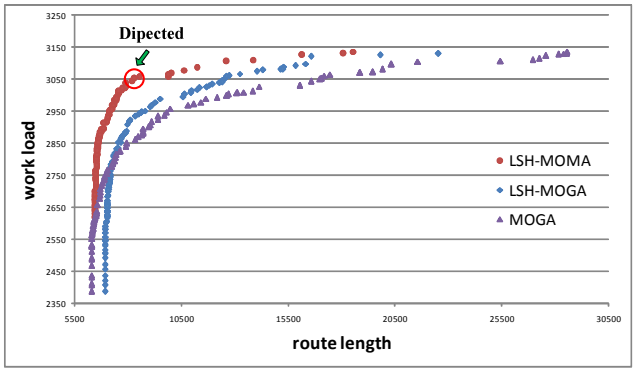


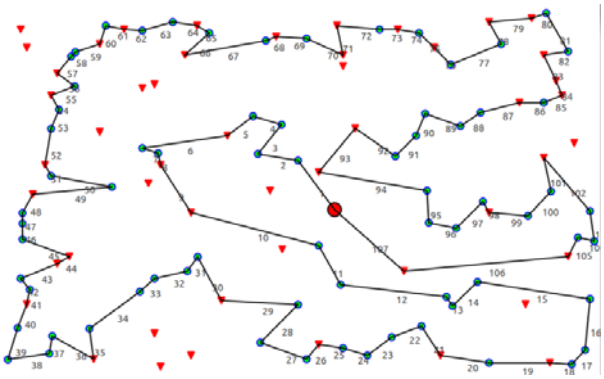
(a) DPDP 1 (30 static, 50 dynamic)





(b) DPDP 2 (50 static, 50 dynamic)





(c) DPDP 3 (70 static, 50 dynamic)

Fig 4. Performance of LSH-MOMA, LSH-MOGA, and MOGA on three simulated DPDPs.

dynamic response rate indicates how many percent of the dynamic requests are served. It is shown that LSH-MOMA obtained better performance than the other two algorithms, which suggests LSH-MOMA does benefit from the LSH method and LSH-based local search.

The Pareto optimal fronts obtained by the three algorithms in one randomly selected run are plotted in Fig 4 (left part). LSH-MOMA is observed to attain Pareto optimal front dominating that of LSH-MOGA, which reveals the effect of LSH-based local search. Both LSH-MOMA and LSH-MOGA dominate MOGA, suggesting LSH method is capable of boosting the search performance of the algorithms.

A solution on the Pareto optimal front of LSH-MOMA is selected to be depicted for each simulated DPDP in the right part of Fig. 4, where the big red spot denotes the depot, the small blue spots represent the static nodes, and the red triangle are dynamic nodes. The routes shown in Fig. 4 look reasonable as all static nodes and a major part of dynamic nodes are served. Those unserved dynamic nodes are either far away from the route or of few pickup and delivery commodities, so they could be ignored for the sake of route length or workload.

VI. CONCLUSIONS

In this paper, we propose a locality-sensitive hashing based multiobjective memetic algorithm (LSH-MOMA) for solving dynamic pickup and delivery problems (DPDPs). LSH-MOMA simultaneously minimizes the route length and maximizes the workload using the ideal of Pareto optimality. Novel LSH-based rectification and local search are proposed to repair and refine the candidate solutions, so that the algorithm converges faster to better Pareto optimal set. Experimental results on three simulated DPDPs demonstrate the efficiency of LSH-MOMA.

REFERENCES

[1]. G. Berbeglia, J. F. Cordeau, and G. Laporte. "Dynamic pickup and delivery problems," *European Journal of Operational Research*, vol. 202, no. 1, pp. 8-15, 2010.

[2]. M. R. Swihart, and J. D. Papastavrou. "A stochastic and dynamic model for the single-vehicle pick-up and delivery problem," *European Journal of Operational Research*, vol. 114, no. 3, pp. 447-464, 1999.

[3]. S. Salhi, and G. Nagy. "A cluster insertion heuristic for single and multiple depot vehicle routing problems with backhauling" *Journal of the Operational Research Society*, vol. 50, no. 10, pp. 1034-1042, 1999.

[4]. C. H. Li, and S. X. Yang. "A generalized approach to construct benchmark problems for dynamic optimization," *Lecture Notes in Computer Science*, *Simulated Evolution and Learning,* vol. 5361, pp. 391-400, 2008.

[5]. J. Renaud, F. F. Boctor, and J. Ouennicls. "A heuristic for the pickup and delivery traveling salesman problem," *Computers & Operations Research*, vol. 27, no. 9, pp. 905-916, 2000.

[6]. K. C. Tan, Y. H. Chew, and L. H. Lee. "A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows," *Computational Optimization and Applications*, vol. 34**,** no. 1, pp. 115-151, 2006.

[7]. M. Mavrovouniotis, and S. X. Yang. "Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors," *Applied Soft Computing*, vol. 13, no. 10, pp. 4023-4037, 2013.

[8]. M. Mavrovouniotis, and S. X. Yang. "Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem," *2012 IEEE Congress on Evolutionary Computation*, pp.1-8, 2012

[9]. P. Sombuntham."Benchmark problem instances for generalized multi-depot vehicle routing problems with pickup and delivery requests," *Proceedings of the 2012 Asia Pacific Industrial Engineering & Management Systems Conference*, pp. 290-297, 2012.

[10]. T. T. Nguyen, S. X. Yang, and J. Branke. "Evolutionary dynamic optimization-A survey the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1-24, 2012.

[11]. I. Gribkovskaia, Halskau, G. Lapore, and M. Vicek. "General solutions to the single vehicle routing problem with pickups and deliveries," *European Journal of Operational Research*, vol. 180, no. 2, pp. 568-584, 2007.

[12]. H. Hernández-Pérez, and J. J. Salazar-González "Heuristics for the one-commodity pickup-and-delivery traveling salesman problem," *Transportation Science*, vol. 38, no. 2, pp. 245-255, 2004.

[13]. M. Gendreau, G. Laporte, and D. Vigo. "Heuristics for the traveling salesman problem with pickup and delivery," *Computers & Operations Research*, vol. 26, pp. 699-714, 1999.

[14]. K. C. Tan, and Y. H. Chew. "A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows," *Computational Optimization and Applications*, vol. 34, pp. 115-151, 2006.

[15]. D. Saez, C. E. Cortes, and A. Nunez. "Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithm and fuzzy clustering," *Computers & Operations Research*, vol. 35, pp. 3412-3428, 2008.

[16]. K. C. Tan, L. H. Lee, Q. L. Zhu, and K.Ou. "Heuristic methods for vehicle routing problem with time windows," *Artificial Intelligence in Engineering*, vol. 15, no. 3, pp. 281-295, 2001.

[17]. K. C. Tan, C. Y. Cheong, and C. K. Goh. "Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation," *European Journal of Operational Research*, vol. 177, no. 2, pp. 813-938, 2007.

[18]. A. Andoni, and P. Indyk. "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Communications of the ACM*, vol. 51, no. 1, pp. 117-122, 2008.

[19]. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation,* vol. 6, no. 2, pp.182–197, 2002.