Information Sciences 329 (2016) 73–89

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A multi-objective memetic algorithm based on locality-sensitive hashing for one-to-many-to-one dynamic pickup-and-delivery problem

Zexuan Zhu[a], Jun Xiao[a], Shan He[b], Zhen Ji[a], Yiwen Sun[c,*]

[a] College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
[b] School of Computer Science, University of Birmingham, B15 2TT, UK
[c] Department of Biomedical Engineering, Shenzhen University, Shenzhen 518060, China

ARTICLE INFO

Article history:
Received 31 March 2015
Revised 10 August 2015
Accepted 9 September 2015
Available online 18 September 2015

Keywords:
Memetic algorithm
Multi-objective evolutionary algorithm
Dynamic pickup and delivery problem
Locality-sensitive hashing

ABSTRACT

This paper presents an early attempt to solve one-to-many-to-one dynamic pickup-and-delivery problem (DPDP) by proposing a multi-objective memetic algorithm called LSH-MOMA, which is a synergy of multi-objective evolutionary algorithm and locality-sensitive hashing (LSH) based local search. Three objectives namely route length, response time, and workload are optimized simultaneously in an evolutionary framework. In each generation of LSH-MOMA, LSH-based rectification and local search are imposed to repair and improve the individual solutions. LSH-MOMA is evaluated on four benchmark DPDPs and the experimental results show that LSH-MOMA is efficient in obtaining optimal tradeoff solutions of the three objectives.

1. Introduction

Pickup-and-delivery problems (PDPs) [3,5,47] constitute a subset of vehicle routing problems (VRPs) [13] wherein objects or people have to be transported from an origin to a destination. They are fundamental problems of transportation and logistic distribution aiming to find an optimal vehicle route to serve customers associated with pickup and/or delivery requests under transportation capacity constraints. The route optimization in PDPs can lead to significant economic savings, e.g., the reduction of operating cost, traffic congestion, and pollution emission, and thus increase the sustainability of city development.

The majority of PDPs are optimized using a static model, i.e., all input data of the problems are known in advance and fixed during the route planning, however the increasing competitive pressures and expectations of high customer satisfaction have forced the logistic service providers to respond to dynamic requests over time. The development of communication technologies, geographic information systems and computer technologies also have greatly motivated the study of dynamic PDPs (DPDPs). In DPDPs, vehicle routes can be re-planned according to actual travel times, new requests, and unexpected events [6,8,45,51]. This kind of problem is also called online or real-time routing problem in other work [28]. PDPs have been proved to be NP-hard [5] and DPDPs represent the harder cases.

Evolutionary computation techniques such as Genetic Algorithm (GA) [26], Ant Colony Optimization (ACO) [17], and Particle Swarm Optimization (PSO) [30] have been widely used to solve DPDPs due to their ability to obtain satisfactory results in tractable

* Corresponding author. Tel.: +8675586531480.
E-mail address: ywsun@szu.edu.cn (Y. Sun).

http://dx.doi.org/10.1016/j.ins.2015.09.006
0020-0255/

time cost [40]. For instance, Cheung et al. [10] studied a mathematical model for dynamic fleet management taking into account the real-time vehicle locations, travel time, and incoming pickup and delivery requests. A GA based solution was proposed to determine a route plan through solving a static vehicle routing problem before daily operation, and then the route is re-optimized as dynamic information arrives. Haghani and Jung [25] presented a GA to solve DPDP with soft time windows where multiple vehicles with different capacities, real-time service requests, and time-dependent travel times are considered. Benyahia and Potvin [4] proposed a genetic programming to model the decision process of a human vehicle dispatcher in DPDP. In [43], a GA embedded in a rolling horizon framework was put forward to solve DPDP with time windows for long-distance freight forwarding with newly arriving requests. Cortés et al. [12] proposed a PSO-based approach to solve a DPDP formulated under a hybrid predictive control (HPC) scheme. Sáez et al. [46] extended [12] by developing more efficient solution algorithms based on GA and fuzzy clustering for the multi-vehicle DPDP. The same group in [38] further generalized their approaches based on generic evolutionary algorithms and HPC scheme to support the dispatcher of a real-time dial-a-ride service. Euchi et al. [19] presented an artificial ACO with 2-opt local search to solve dynamic VRP with pickup and delivery.

Most of the aforementioned approaches were proposed for single-objective problems, where the main objective namely traveling cost (in terms of time/distance) is optimized. However, the dynamic nature of DPDP introduces new conflicting objectives such as response time (or latency) and service workload. Multi-objective optimization methods such as multi-objective evolutionary algorithms (MOEAs) [11,15,23,34,44,49,53] are greatly desirable because of their capability of obtaining multiple trade-off solutions. Many MOEAs have been proposed for static VRPs/PDPs [18,20,21,24,29,52,66] and a few for general dynamic VRPs [22,55]. Yet, there is very limited work on multi-objective DPDP. The work of Núnez et al. [41] represents one of the very few attempts to use multi-objective optimization model for DPDP. In [41], the authors proposed a multiobjective-model-based predictive control framework implemented with GA to solve the dial-a-ride problem, i.e., a one-to-one DPDP, considering two opposing goals namely user and operator costs.

This paper proposes a multi-objective memetic algorithm [9,31,37,39,42] namely LSH-MOMA to solve DPDP by optimizing the route length, response time, and service workload, simultaneously. We focus on the single-vehicle one-to-many-to-one (1-M-1) DPDP, one of the most typical routing problems in city courier. In 1-M-1 DPDP, some commodities are initially available at the depot and delivered to the customers in the course of operation. At the same time, some other commodities either requested in advance or dynamically are collected from the customers and destined to the same depot. New customer pickup requests are received through a call center and forward to the vehicle during the operation. The vehicle route is dynamically changed in response to new requests. Many other types of DPDPs, especially one-to-one DPDPs, have been extensively studied and overviews can be found in [6] and [45]. However, efficient solutions for 1-M-1 DPDP are still lacking to provide practical trade-off decisions that can satisfy the multiple conflicting objectives in real-world express courier service.

This work presents an early attempt to solve the 1-M-1 DPDP by proposing LSH-MOMA as a synergy of MOEA and locality-sensitive hashing (LSH) [1] based local search. Memetic algorithms, taking advantage of both population-based global search and local search, have been shown to obtain better performance than their conventional counterparts in various complex real-world problems [2,7,27,32,33,50,57,59,60,62–64] including VRPs [35,56] and PDPs [58]. In LSH-MOMA, a vehicle is arranged to start from a depot and follow an initially scheduled Hamiltonian route to serve static requests. Afterward, the route is re-planned in response to new requests, so that the three objectives are optimized subject to transportation capacity constraint. The static requests must be served in the solution route whereas the dynamic requests could be selectively responded. A population of candidate routes is evolved using a MOEA framework where LSH-based rectification and local search are introduced to repair and improve the candidate solutions respectively in each generation. LSH-MOMA is tested on four benchmark 1-M-1 DPDPs with different scales of customer nodes. The experimental results demonstrate the superiority of LSH-MOMA to the other counterpart algorithms in identifying reasonable tradeoff solutions.

The main contributions of this study are threefold: (1) a three-objective 1-M-1 DPDP is formulated and benchmark problems based on both Euclidean plane and real map are studied; (2) an LSH-based method is proposed for fast identification of geometrically nearest neighbors, which enables route planning algorithms to handle problems in large environments; (3) a new memetic algorithm namely LSH-MOMA is put forward to solve the formulated 1-M-1 DPDP, which is expected to inspire the design of new MOEA-based approaches for DPDPs.

The rest of this paper is organized as follows. Section 2 provides the formulation of the three-objective 1-M-1 DPDP. Section 3 introduces the LSH-based nearest neighbors identification method. Section 4 presents the details of the proposed LSH-MOMA. Section 5 provides the experimental design and results on benchmark problems. Finally, Section 6 concludes this study.

2. Problem formulation

This section formulates the single-vehicle 1-M-1 DPDP. As shown in Fig. 1, to solve a 1-M-1 DPDP is to find an optimal closed-loop vehicle route that starts from a depot, then accesses the customer nodes in the course and finally ends up at the same depot. The vehicle carries delivery commodities on off-line requests away from the depot, and then serves delivery and pickup requests following a planned route. Once new dynamic requests arrive, the vehicle could choose to respond the requests by re-planning the route or just ignore them. The optimality of the route is defined in terms of three objectives namely route length, response time, and service workload in this study.

Without loss of generality, we confine the routing of a 1-M-1 DPDP in a rectangle $R = R_x \times R_y$. The definitions of customer node, depot, service route, vehicle capacity constraint, real-time load, route length, response time and workload are provided as follows:

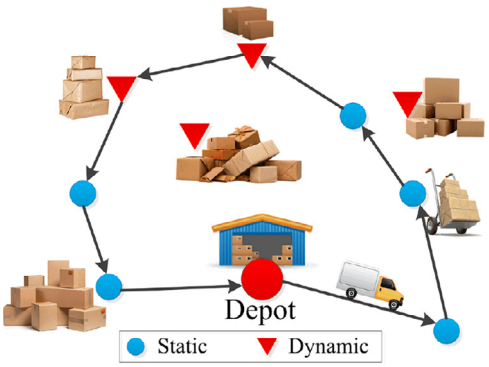


Fig. 1. An illustration of single-vehicle 1-M-1 DPDP.

Customer node: a customer is represented as a node in the environment R. Let $O = \{1, 2, \ldots\}$ index all customer nodes located in R, $S(\subseteq O)$ index the set of **static nodes** of which the requests are known before the vehicle set off, and $D(\subseteq O)$ index the set of **dynamic nodes** whose requests arise during the execution of the transportation, i.e., $O = S \bigcup D$. The sizes of O and D increase as new requests arrive. A customer node i is associated with four attributes (x_i, y_i, u_i, v_i), where x_i and y_i are the coordinates of the node in R, while u_i and v_i indicate the amounts of pickup and delivery commodities requested at this node, respectively. Dynamic nodes make merely pickup requests. i.e., $v_i = 0$ if $i \in D$. All static nodes must be served whereas the dynamic ones are selectively responded subject to the vehicle capacity constraint. Those unserved dynamic nodes will automatically become static nodes in next round of route planning.

Depot: the depot is a special customer node indexed with 0. It is the source and destination of a service route. The delivery commodities of all static nodes must be loaded to the vehicle at the depot before departure, and then distributed to the corresponding customers in the serving course. Meanwhile, all commodities picked up in the route should be carried back to the same depot in the end.

Service route: a cyclic service route is represented as a sequence of customer nodes accessed by the vehicle. Particularly, a route is denoted as $P = \{p_0, p_1, \ldots, p_l\}$, where $p_0 = p_l = 0$, $p_i \in \{1, 2, \ldots, |O|\}$ $(0 < i < l)$, $p_i \neq p_j$ $(i \neq j)$, and $S \subseteq P \subseteq O$. A node accessed in a route is also referred to as a **route node**.

Vehicle capacity constraint: the vehicle is subject to a maximum capacity C. A feasible route contains no violation of the constraint at any route node.

Real-time load: given a candidate route P, the real-time load of the vehicle at node p_i is calculated as follows:

$$\Omega(p_i) = \sum_{j=0}^{i} \left(u_{p_j} - v_{p_j}\right) \tag{1}$$

where $\Omega(p_i)$ accumulates the load changes as the vehicle goes along the route from the depot, i.e., p_0, to node p_i. A capacity constraint violation occurs if $\Omega(p_i) > C$ is detected at any route node. In the following text, we assume that neither the total pickup nor delivery commodities of all static nodes exceeds the vehicle capacity, i.e., $\max\{\sum_{i\in S} u_i, \sum_{i\in S} v_i\} < C$, as such feasible routes to serve all static nodes are guaranteed to exist. Any dynamic node with a pickup request greater than $C - \sum_{i\in S} u_i$ must be rejected.

Route length: the length of a route P, denoted as $L(P)$, is calculated by summing up the distance of every two adjacent nodes in P, i.e.,

$$L(P) = \sum_{i=0}^{l-1} d(p_i, p_{i+1}) \tag{2}$$

where $d(p_i, p_{i+1})$ indicates the distance between customer nodes p_i and p_{i+1}. Similarly, the length of a route segment from node p_m to node p_n $(0 \leq m < n \leq l)$ can be defined as follows:

$$L(p_m, p_n) = \sum_{i=m}^{n-1} d(p_i, p_{i+1}) \tag{3}$$

Response time: the response time of a route P is the total amount of time all customer nodes wait to get served:

$$T(P) = \sum_{i=1}^{|O|} t_i \tag{4}$$

$$t_i = \begin{cases} L(p_0, p_j)/V & \text{if} \quad \exists p_j \in P : p_j == i \quad \text{and} \quad i \in S; \\ L\left(q(o_i), p_j\right)/V & \text{if} \quad \exists p_j \in P : p_j == i \quad \text{and} \quad i \in D; \\ L\left(q(o_i), p_l\right)/V & \text{otherwise} \quad \nexists p_j \in P : p_j == i. \end{cases} \tag{5}$$

where t_i indicates the waiting time of customer node i, V is the average speed of the vehicle (set to 1 for simplicity), and $q(o_i)(\in P)$ denotes the route node the vehicle is accessing when node i submits its request. If node i is static, it must be included in the route, say $\exists p_j \in P : p_j == i$, so t_i is the traveling time of the vehicle from the depot to node p_j. If node i is dynamic and included in the route, t_i is the traveling time of the vehicle from node $q(o_i)$ to node i. Otherwise, t_i is the traveling time of the vehicle from node $q(o_i)$ to the depot, i.e., the customer has to wait till the whole journey ends.

Workload: the service workload of a route P is calculated by summing up the pickup and delivery commodities over all route nodes, i.e.,

$$W(P) = \sum_{i=1}^{l-1} \left(u_{p_i} + v_{p_i}\right) \tag{6}$$

where u_{p_i} and v_{p_i} indicate the amount of pickup and delivery commodities at node p_i, respectively.

To solve the above 1-M-1 DPDP is to find an optimal cyclic route that simultaneously minimizes $L(P)$ and $T(P)$ but maximizes $W(P)$. The objective function $F(P)$ of a route can be formulated as follows:

$$\begin{aligned} &\min F(P) = \left(L(P), T(P), -W(P)\right) \\ &\text{subject to } \forall p_i \in P : \Omega(p_i) \le C \end{aligned} \tag{7}$$

Note that minimizing $-W(P)$, i.e., maximizing $W(P)$, demands for serving more dynamic nodes, which inevitably increases the path length $L(P)$. Conversely, fewer route nodes should be involved to minimize $L(P)$, which reduces the workload $W(P)$, i.e., increases $-W(P)$. So the minimizations of $L(P)$ and $-W(P)$ conflict with each other. The response time $T(P)$ correlates with both $L(P)$ and $W(P)$ to some extent. On one hand, lower $L(P)$, i.e., shorter route, means lower response time of the static nodes. On the other hand, larger workload $W(P)$, achieved by serving more dynamic nodes, results in lower response time of the dynamic nodes. Due to the conflicting nature of the three objectives, there is no single solution can minimize them simultaneously without tradeoff.

The idea of Pareto-optimality [16] is applied here to solve this three-objective optimization problem such that multiple trade-off solutions can be obtained in a single run, from which appropriate ones can be selected according to different preferences in real-world practices. Given two feasible routes $P1$ and $P2$, $F(P1)$ is said to dominate $F(P2)$ if and only if $L(P1) \le L(P2)$, $T(P1) \le T(P2)$, $-W(P1) \le -W(P2)$, and in at least one objective $P1$ is smaller than $P2$. A feasible route P^* is Pareto optimal to Eq. (7) if there is no other solution P such that $F(P)$ dominates $F(P^*)$. $F(P^*)$ is called a Pareto optimal vector. All Pareto optimal routes form the Pareto optimal set and the set of corresponding Pareto optimal vectors is referred to as the Pareto front [36].

3. LSH based nearest neighbors identification

Before introducing LSH-MOMA, an LSH [1] based nearest neighbor identification method, which is a key component in route rectification and local search, is proposed in this section. The basic idea of LSH is to hash customer nodes according to their locations so that nearby nodes are mapped to the same hash value with a high probability. The problem of searching the nearest neighbors of a node is then transformed as a problem of identifying the nodes of the same hash values. Therefore, using LSH is able to quickly identify the approximated nearest neighbors of a customer node without the time-consuming pairwise distance calculation. This is particularly useful for distance calculation on a real map where the distance of two nodes is subject to the availability of physical connections.

The key issue of implementing LSH is the design of the hash function. In this study, the search region R is divided into $n \times n$ equal lattices as shown in Fig. 2. The hash function of a customer node i is defined by mapping the node to the corresponding lattice that contains the node:

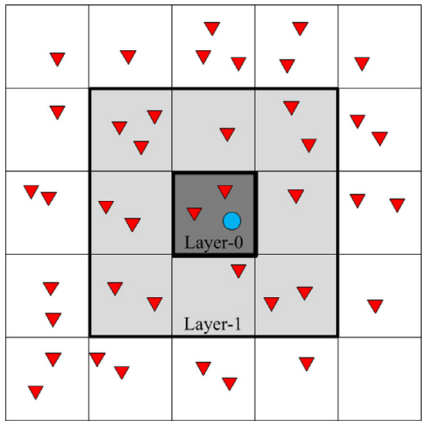


Fig. 2. LSH based nearest neighbors identification.

$$H(i) = \left(\left\lceil \frac{nx_i}{R_x} \right\rceil, \left\lceil \frac{ny_i}{R_y} \right\rceil \right) \tag{8}$$

where the ceiling function $\lceil \cdot \rceil$ returns the smallest integer greater than the input value. Based on this hash function, the search of the nearest neighbors of a node i can be confined in a small region, i.e., one or a handful of lattices, thus avoiding searching the whole region R. We define S_i^k as the set of layer-k nearest neighbors of node i:

$$S_i^k = \left\{ j \,\middle|\, \left| \left\lceil \frac{nx_j}{R_x} \right\rceil - \left\lceil \frac{nx_i}{R_x} \right\rceil \right| \le k, \left| \left\lceil \frac{ny_j}{R_y} \right\rceil - \left\lceil \frac{ny_i}{R_y} \right\rceil \right| \le k \right\} \tag{9}$$

The detailed procedure of LSH-based nearest neighbors identification is provided in Algorithm 1. Given a queried node i and the layer k, the algorithm first calculates the hashed values (A, B) of node i based on Eq. (8). And then the algorithm scans from lattice row $\max(A-k, 1)$ (i.e., takes 1 if $A-k<1$) to row $\min(A+k, n)$ (i.e., takes n if $A+k>n$), and meanwhile from column $\max(B-k, 1)$ (i.e., takes 1 if $B-k<1$) to column $\min(B+k, n)$ (i.e., takes n if $B+k>n$), to find out all nodes within the scope to form the set of layer-k nearest neighbors S_i^k of node i. For example, a target node denoted as the blue dot in Fig. 2 is mapped to lattice (3,3) by the hash function. The layer-0 nearest neighbors of this node are located in the same lattice, while the layer-1 nearest neighbors spread to involve the nodes in the outer surrounding eight lattices.

Algorithm 1 The procedure of LSH based nearest neighbors identification.

INPUT: the hash values of all customer nodes (calculated off-line), a queried node i, and layer k;
OUTPUT: the set of layer-k nearest neighbors of node i, i.e., S_i^k;
BEGIN
1: $(A, B) = H(i)$; {//A and B are the coordinates of the mapped lattice of node i}
2: $S_i^k = \{\}$;
3: **For** $a = \max(A-k, 1)$ to $\min(A+k, n)$ **do** {//Scan from row $A-k$ (or 1 if $A-k<1$) to row $A+k$ (or n if $A+k>n$)}
4: **For** $b = \max(B-k, 1)$ to $\min(B+k, n)$ **do** {//Scan from column $B-k$ (or 1 if $B-k<1$) to column $B+k$ (or n if $B+k>n$)}
5: Identify $S_{\text{tmp}} = \{j \mid H(j) == (a, b)\}$;
6: $S_i^k = S_i^k \bigcup S_{\text{tmp}}$;
7: **End For**
8: **End For**
END

4. LSH-MOMA for 1-M-1 DPDP

This section presents the details of the proposed LSH-MOMA. The flowchart of the algorithm is shown in Fig. 3. Before the vehicle leaves the depot, a population of candidate routes (or called individuals) are initially scheduled for all static nodes only. In the course of serving, dynamic requests are received and buffered in a request pool. The pool is checked by LSH-MOMA in a fixed time slice (a.k.a decision epoch) say Γ generations, and dynamic nodes are inserted into the candidate routes using the mutation operator and local search. In each time slice, the vehicle also completes service of the first unserved node in each candidate route, i.e., one node is fixed in each candidate route through the ongoing evolution. In other words, the changeable part in each route is becoming shorter as the population evolves. 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 repair and refine each individual. The outline of LSH-MOMA is provided in Algorithm 2. Details of the key components of LSH-MOMA are provided in the following subsections.

4.1. Chromosome encoding scheme and initialization

In LSH-MOMA, a variable-length chromosome, represented as an integer string, is used to encode a candidate route. As shown in Fig. 4, the chromosome starts and ends at the same depot, i.e., $p_0 = p_l = 0$, and the intermediate part is a sequence of customer node indexes accessed in the route. Because p_0 and p_l are always fixed, they can be omitted in the encoding for the sake of simplicity. At the beginning of LSH-MOMA, a population of chromosomes is randomly generated to encode candidate routes consisting of all static nodes, i.e., each chromosome is a random permutation of all static nodes' indexes.

4.2. LSH-based rectification

Once a new route is generated by initialization or evolutionary operation, the feasibility of the route should be verified and capacity constraint violations are forbidden. An LSH-based rectification is proposed to detect and repair capacity constraint violations occurring in a route. The procedure of LSH-based rectification is provided in Algorithm 3.

Given a candidate path $P = \{p_0, p_1, \ldots, p_l\}$, the rectification sequentially checks the real-time load $\Omega(p_i)$ at each unserved (unfixed) node p_i. If $\Omega(p_i) > C$ holds, a violation is detected and then LSH-based rectification is executed. More specifically, a descendant route node p_j $(j > i)$ that falls within $S_{p_i}^k$ (i.e., the set of layer-k nearest neighbors of p_i) and possesses the largest

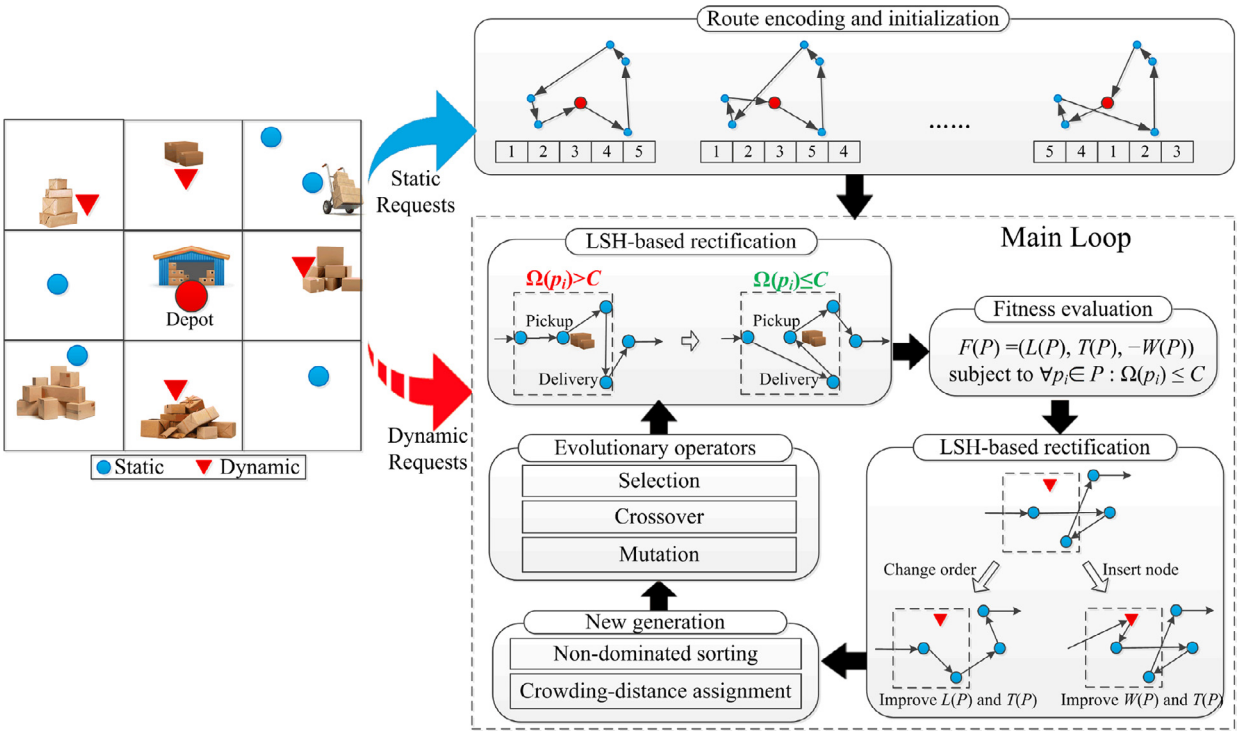


Fig. 3. The flowchart of LSH-MOMA.

Algorithm 2 LSH-MOMA for DPDP.

BEGIN

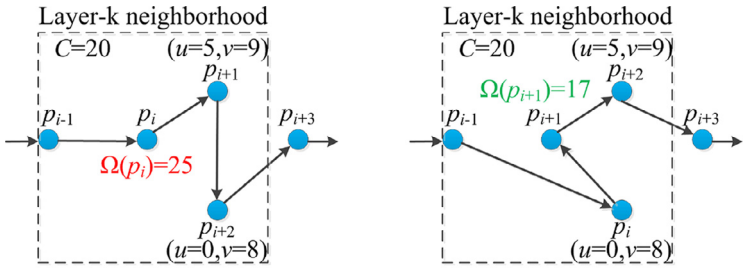
$i = 1$;
Randomly initialize a population Q_i of candidate routes considering the static nodes only;
Apply LSH-based rectification to each individual in Q_i;
Evaluate the fitness of each individual based on Eq. (7);
While the maximum number of generations is not reached **do**
  If (i mod Γ) == 0 **then** {//Update the request pool in a fixed time slice}
    Fix the first unserved node in each individual route;
    If new dynamic requests arise **then**
      Add new requests to the request pool;
    End If
  End If
  Generate an offspring population Q_i' with selection, crossover, and mutation;
  Apply LSH-based rectification on each offspring;
  Evaluate the fitness of each offspring based on Eq. (7);
  Apply LSH-based local search on each offspring;
  Generate new population Q_{i+1} from $Q_i \bigcup Q_i'$ based on Pareto optimality;
  $i = i + 1$;
End While

END

positive deliver-pickup residual (i.e., $v_{p_j} - u_{p_j}$) is first identified, and then node p_j is rearranged to the position right before node p_i in P, as such the real-time load can be reduced by $v_{p_j} - u_{p_j}$. The rectification operation is repeated until the real-time load falls below C. If no suitable node can be found in $S^k_{p_i}$, the operation increases k to enlarge the search scope. Since we assume the total amount of pickup commodities of all static nodes is smaller than the vehicle capacity, i.e., $\sum_{i\in S} v_i \le C$ and no dynamic pickup request exceeding $C - \sum_{i\in S} v_i$ is involved, the rectification is guaranteed to reach a feasible path with sufficiently large k.

A simple example is given in Fig. 5 to illustrate the procedure of LSH-based rectification. In Fig. 5(a), the first violation is detected at node p_i where the real-time load $\Omega(p_i) = 25$ exceeds the vehicle capacity $C = 20$, so the rectification operator searches the layer-k nearest neighbors of node p_i and identifies node p_{i+2} as the one possessing the largest deliver-pickup residual

Fig. 4. The chromosome encoding scheme.



(a) Before rectification (b) After rectification

Fig. 5. An example of route rectification.

Algorithm 3 The procedure of LSH-based rectification.

INPUT: a candidate route $P = \{p_0, p_1, \ldots, p_l\}$;
BEGIN
1: Identified the first unserved node say p_m in P;
2: **For** $i = m$ to $l - 1$ **do**
3: $k = 0$;
4: **While** $\Omega(p_i) > C$ **do**
5: Identify $S^k_{p_i}$ based on Algorithm 1;
6: $S = S^k_{p_i} \bigcap \{p_{i+1}, p_{i+2}, \ldots, p_l\}$; {//Identify descendant route nodes that fall within the layer-k nearest neighborhood of p_i}
7: **If** $\exists p_j : p_j \in S$ and $\forall s \in S : \max(v_s - u_s, 0) < v_{p_j} - u_{p_j}$ **then** {//p_j possesses the largest positive deliver-pickup residual in S}
8: Reorder node p_j right before node p_i in P;
9: $i = i + 1$;
10: **Else**
11: $k = k + 1$;
12: **End If**
13: **End While**
14: **End For**
END

($v_{p_{i+2}} - u_{p_{i+2}} = 8$) in the neighborhood. Afterward, node p_{i+2} is reordered to right before node p_i in P and the orders of the other nodes are changed accordingly. After the rectification, the violation is avoided as the corresponding real-time load $\Omega(p_{i+1})$ reduces to 17.

4.3. Evolutionary operators

After fitness evaluation, evolutionary operators including selection, crossover, and mutation are applied to the population Q_i to generate an offspring population Q'_i. More specifically, the binary tournament selection is first applied to select parent individuals to undergo order-based crossover [54], and then the generated offspring are mutated by randomly adding/deleting a dynamic node, swapping two nodes, or segment reversion. In segment reversion, a subsequence of a route is randomly selected and the order of the corresponding route nodes is reversed. For example, given a route $P = \{0, 1, 5, 3, 2, 4, 0\}$, assume that the segment ranging from the second node to the fifth one is selected, then the segment reversion converts the route to $P = \{0, 2, 3, 5, 1, 4, 0\}$. The procedure of mutation is provided in Algorithm 4.

Note that crossover and mutation do not affect the nodes that have already been served in each chromosome. After crossover and mutation, LSH-based rectification and local search are used to fix the capacity constraint violations and improve the quality of the offspring routes. A new population Q_{i+1} is generated from the combination of the offspring population Q'_i and the original population Q_i based on non-domination rank and crowding distance following NSGA-II [16].

4.4. LSH-based local search

To improve the quality of candidate routes, an LSH-based local search is introduced here to fine-tune the offspring generated by evolutionary operators. Particularly, given a candidate route P, the LSH-based local search randomly selects an unserved node

p_i in P, and then locally changes the order of the node in P or adds a nearest neighbor of p_i to the route. If the resulting new route dominates the original one, it replaces the original one. Conversely, if the new one is dominated by the original one, the new route is abandoned. If no dominance can be found in either way, the new route is added to the offspring population. The details of the LSH-based local search are given in Algorithm 5.

Algorithm 4 The procedure of mutation.

INPUT: a candidate route $P = \{p_0, p_1, \ldots, p_l\}$;
BEGIN
Generate a random number r in $[0, 1]$;
If $0 \leq r < 0.25$ **then**
  Randomly select a new dynamic node and insert it to P in a randomly selected position.
Else If $0.25 \leq r < 0.5$ **then**
  Randomly remove a dynamic node from P.
Else If $0.5 \leq r < 0.75$ **then**
  Randomly swap the orders of two nodes in P.
Else
  Randomly select a segment in P and reverse it.
End If
END

Algorithm 5 The procedure of LSH-based local search.

INPUT: a candidate route $P = \{p_0, p_1, \ldots, p_l\}$ and k;
BEGIN
Randomly select an unserved node p_i in P;
Identified $S^k_{p_i}$ based on Algorithm 1;
Randomly select an unserved node j in $S^k_{p_i}$; {//If $j \in D$, $u_j < C - \sum_{i \in S} v_i$ should hold true}
If $j \in P$ **then**
  Reorder j to right before or after p_i with equalpossibility; {//Illustrated in Fig. 6 (left branch)}
Else
  Insert j to P right before or after p_i with equalpossibility; {//Illustrated in Fig. 6 (right branch)}
End If
Apply LSH-based rectification to the new route P'.
Evaluate the fitness of P' based on Eq. (7);
If P' dominates P **then**
  Replace P with P';
Else If P' is better that P in terms of any one objective **then**
  Add P' to the offspring population;
Else {//P dominates P'}
  Abandon P';
End If
END

Examples of LSH-based local search are provided in Fig. 6. Suppose route node p_i is selected to undergo local search, there could be two ways to improve it: (1) select another route node say p_{i+2} which is also within the layer-k neighborhood of p_i, and reorder it to p_{i+1} (as shown in the left branch), such that both route length $L(P)$ and response time $T(P)$ are reduced; (2) select a dynamic node excluded in P from the neighbors of node p_i, and insert the new node to the route right before p_i (as shown in the right branch). As such, the workload $W(P)$ is increased and the response time $T(P)$ is very likely reduced. The improvement of $T(P)$ mainly comes from the substantial reduction of the dynamic node's response time that could overwhelm the small extra waiting time caused to the other route nodes.

5. Experimental studies

5.1. Experimental design

In this section, we use four benchmark 1-M-1 DPDPs of different scales to test the performance of LSH-MOMA. In the first two DPDPs, 30 and 50 static nodes are first distributed on a Euclidean plane randomly, and then another 50 dynamic nodes are added to each DPDP. The dynamic nodes are sequentially visible to LSH-MOMA by one in a time slice. Note that because the dynamic nodes are selectively responded, increasing the number of dynamic nodes does not necessarily increase the complexity of the

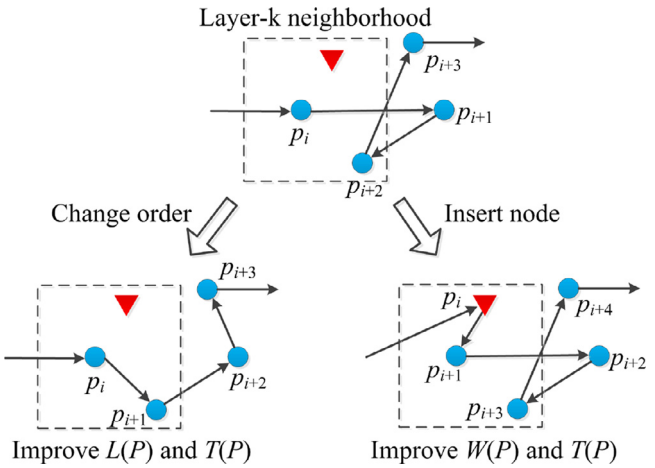


Fig. 6. Examples of LSH-based local search.

Table 1
Information of the four benchmark DPDPs.

	Environment	C	$\|S\|$	$\|D\|$	Distance measure
DPDP1	Euclidean plane	875	30	50	Euclidean distance
DPDP2	Euclidean plane	1338	50	50	Euclidean distance
DPDP3	Real map (Shenzhen)	1021	30	50	Google Distance Matrix API
DPDP4	Real map (Shenzhen)	1774	50	50	Google Distance Matrix API

problem. Here we fix the number of dynamic nodes to a large enough value for the sake of simplicity. The distance of every two nodes in the first two DPDPs is measured with Euclidean distance. The other two DPDPs are generated similarly, except that they are planted on a real map of downtown Shenzhen and the distance of every two nodes on the map is calculated using Google Distance Matrix API. The data of customer nodes, including coordinates, pickup commodities, and delivery commodities of the four DPDPs are publicly available in [61]. The maximum vehicle capacity C in each DPDP is configured as $\max(\sum_{i\in S} v_i, \sum_{i\in S} u_i)/0.7$, such that the existence of feasible route(s) to serve all static nodes is guaranteed. The information of the four benchmarks is summarized in Table 1 and the distributions of customer nodes are plotted in Fig. 7.

LSH-MOGA, i.e., LSH-MOMA without LSH-based local search, and MOGA, i.e., LSH-MOMA without LSH-based rectification nor local search, are considered in the comparison studies to test the effects of LSH-based rectification and 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 slice $\Gamma = 10$. LSH-MOMA is terminated when the generation number exceeds 500. To ensure a fair comparison, 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 mean results are reported. All algorithms are implemented in C++ and run on a PC with Intel Pentium 4 2.4 GHz with 2GB memory.

5.2. Evaluation criteria

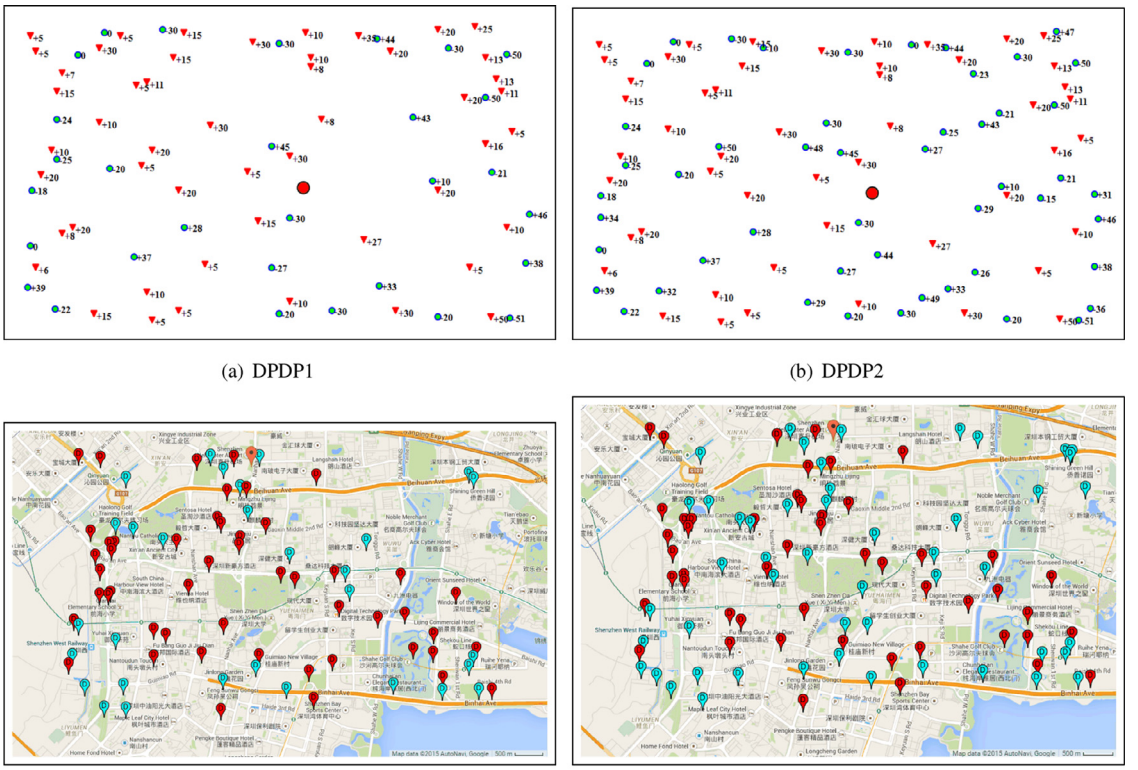
The performances of the algorithms are evaluated in terms of the following six criteria:

- Best route length $L^*(P)$: the best route length obtained in a single run.
- Best response time $T^*(P)$: the best response time obtained in a single run.
- Best workload $W^*(P)$: the best workload obtained over in a single run.
- Convergence metric (C-Metric) [14]: this metric is introduced to evaluate the distance between the final set of non-dominated solutions and the true Pareto-optimal set. Let $\mathcal{P}$ and Q^* indicate the true Pareto-optimal set and the set of non-dominated solutions obtained by an MOEA, respectively. For each solution Q_i^* in Q^*, the smallest normalized distance to $\mathcal{P}$ is calculated as follows:

$$d_i = \min_{j=1}^{|\mathcal{P}|} \sqrt{\sum_{m=1} \frac{f_m(Q_i^*) - f_m(\mathcal{P}_j)}{f_m^{\max} - f_m^{\min}}} \tag{10}$$

where $f_m(\cdot)$ computes the mth objective value. $f_m^{\max}$ and $f_m^{\min}$ represent the maximum and minimum values of the mth objective in $\mathcal{P}$. The C-Metric of Q^* is defined as the average normalized distance of all solutions in it:

$$M_C(Q^*) = \frac{1}{|Q^*|} \sum_{i=1}^{|Q^*|} d_i \tag{11}$$



(a) DPDP1 (b) DPDP2

(c) DPDP3 (d) DPDP4

Fig. 7. The node distributions of the four DPDPs with static and dynamic nodes in blue and red spots, respectively (the load change at each node, i.e., $u_i - v_i$, is also labeled in the first two DPDPs). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

The lower the C-Metric is, the better the convergence ability an algorithm processes. Since the true Pareto-optimal set $\mathcal{P}$ is unknown a priori in this study, $\mathcal{P}$ is approximated by the overall non-dominated solutions obtained by all algorithms in all experimental runs.

- Spacing metric (S-Metric) [48]: this metric evaluates the variance of the distance of each member in the non-dominated solutions to its closest neighbor. Let Q^* be the non-dominated solutions obtained by a multi-objective algorithm. For each solution Q_i^* in Q^*, the smallest normalized distance to other solutions in Q^* is defined as follows:

$$d_i = \min_{j=1, j\neq i}^{|Q^*|} \sum_{m=1} |f_m(Q_i^*) - f_m(Q_j^*)| \tag{12}$$

where $f_m(\cdot)$ calculates the mth objective function value. The S-Metric of Q^* is defined as follows:

$$M_S(Q^*) = \sqrt{\frac{1}{|Q^*|-1} \sum_{i=1}^{|Q^*|} (\bar{d} - d_i)^2} \tag{13}$$

where $\bar{d}$ is the average value of d_i. A set of non-dominated solutions that has the lower S-Metric has the better uniformity. Note that a value of zero for S-Metric would mean that all members of Q^* are equally spaced from one another.

- Coverage-of-two-sets metric (C2S-Metric) [65]: this metric is adopted to compare two sets of non-dominated solutions. Let Q_A^* and Q_B^* be two sets of non-dominated solutions obtained by two algorithms respectively. The percentage of solutions in Q_B^* dominated by that of Q_A^* is calculated with $I(Q_A^*, Q_B^*)$:

$$I(Q_A^*, Q_B^*) = \frac{|\{b \in Q_B^* \,|\, \exists a \in Q_A^*, a \succeq b\}|}{|Q_B^*|} \tag{14}$$

where $a \succeq b$ suggests that solution a dominates or equals to solution b. Similarly, $I(Q_B^*, Q_A^*)$ computes the percentage of solutions in Q_A^* dominated by solutions in Q_B^*. If $I(Q_A^*, Q_B^*) \geq I(Q_B^*, Q_A^*)$, then the solutions in Q_A^* are better than those in Q_B^*. It is worth noting that both $I(Q_A^*, Q_B^*)$ and $I(Q_B^*, Q_A^*)$ fall within [0,1] and $I(Q_A^*, Q_B^*) + I(Q_B^*, Q_A^*)$ need not equal to 1.

Table 2
The results in terms of mean route length, response time, workload, convergence metric, and spacing metric of LSH-MOMA, LSH-MOGA, and MOGA on the four benchmark DPDPs over 25 runs. ($^-/^\approx$: the average performance of the corresponding method is significantly worse than or similar to the highlighted best average performance at level $\alpha = 0.05$ in Welch's t-test.)

		LSH-MOMA	LSH-MOGA	MOGA
DPDP1	$\overline{L^*(P)}(\times 10^3)$	**4.73 ± 0.05**	5.02 ± 0.25$^-$	5.04 ± 0.29$^-$
	$\overline{T^*(P)}(\times 10^4)$	**13.66 ± 0.47**	18.18 ± 1.22$^-$	18.84 ± 0.82$^-$
	$\overline{W^*(P)}(\times 10^3)$	**1.49 ± 0.00**	**1.49 ± 0.00**	1.48 ± 0.00$^-$
	$\overline{M_C}(\times 10^3)$	**4.39 ± 2.33**	11.42 ± 5.06$^-$	20.39 ± 9.13$^-$
	$\overline{M_S}(\times 10^6)$	**0.30 ± 0.10**	0.55 ± 0.11$^-$	1.25 ± 0.64$^-$
DPDP2	$\overline{L^*(P)}(\times 10^3)$	**5.80 ± 0.17**	6.25 ± 0.41$^-$	6.74 ± 0.36$^-$
	$\overline{T^*(P)}(\times 10^4)$	**21.58 ± 0.65**	24.18 ± 1.49$^-$	25.09 ± 2.66$^-$
	$\overline{W^*(P)}(\times 10^3)$	**2.28 ± 0.00**	**2.28 ± 0.00**	2.27 ± 0.00$^-$
	$\overline{M_C}(\times 10^3)$	**5.31 ± 1.49**	23.90 ± 10.41$^-$	74.05 ± 39.39$^-$
	$\overline{M_S}(\times 10^6)$	**0.30 ± 0.09**	0.44 ± 0.11$^-$	3.32 ± 2.19$^-$
DPDP3	$\overline{L^*(P)}(\times 10^3)$	**66.96 ± 1.62**	68.04 ± 1.84$^-$	68.58 ± 2.32$^-$
	$\overline{T^*(P)}(\times 10^6)$	**1.37 ± 0.09**	1.48 ± 0.08$^-$	1.73 ± 0.11$^-$
	$\overline{W^*(P)}(\times 10^3)$	**1.74 ± 0.00**	**1.74 ± 0.00**	1.72 ± 0.01$^-$
	$\overline{M_C}(\times 10^3)$	**10.21 ± 7.00**	35.76 ± 41.98$^-$	912.37 ± 323.21$^-$
	$\overline{M_S}(\times 10^6)$	**3.60 ± 1.13**	3.64 ± 1.17$^\approx$	45.45 ± 13.27$^-$
DPDP4	$\overline{L(P)}(\times 10^3)$	**85.17 ± 2.37**	94.18 ± 5.91$^-$	94.43 ± 5.43$^-$
	$\overline{T(P)}(\times 10^6)$	**2.36 ± 0.18**	2.76 ± 0.19$^-$	2.88 ± 0.21$^-$
	$\overline{W(P)}(\times 10^3)$	**3.02 ± 0.00**	**3.02 ± 0.00**	3.00 ± 0.01$^-$
	$\overline{M_C}(\times 10^4)$	**7.85 ± 0.93**	31.64 ± 20.99$^-$	189.19 ± 43.57$^-$
	$\overline{M_S}(\times 10^6)$	**3.97 ± 0.93**	4.99 ± 1.80$^-$	88.60 ± 18.69$^-$

Table 3
The results in terms of coverage-of-two-sets metric of LSH-MOMA (A), LSH-MOGA (B), and MOGA (C) on the four benchmark DPDPs over 25 runs. ($^-/^\approx$: the average performance of the corresponding method is significantly worse than or similar to the highlighted best average performance at level $\alpha = 0.05$ in Welch's t-test.)

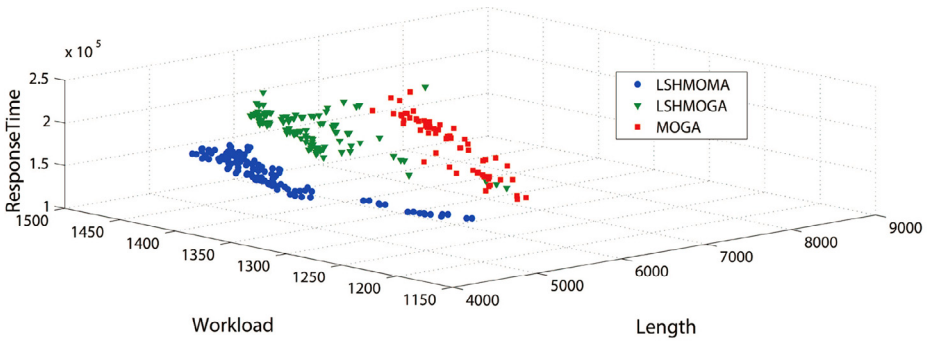
	LSH-MOMA vs. LSH-MOGA		LSH-MOMA vs. MOGA		LSH-MOGA vs. MOGA	
	$I(A, B)$	$I(B, A)$	$I(A, C)$	$I(C, A)$	$I(B, C)$	$I(C, B)$
DPDP1	**0.99 ± 0.01**	0.00 ± 0.00$^-$	**0.98 ± 0.02**	0.00 ± 0.00$^-$	**0.76 ± 0.37**	0.65 ± 0.28$^\approx$
DPDP2	**0.94 ± 0.02**	0.12 ± 0.30$^-$	**0.96 ± 0.08**	0.12 ± 0.29$^-$	**0.75 ± 0.25**	0.66 ± 0.37$^\approx$
DPDP3	**0.82 ± 0.31**	0.25 ± 0.36$^-$	**0.98 ± 0.02**	0.13 ± 0.26$^-$	**0.94 ± 0.07**	0.35 ± 0.36$^-$
DPDP4	**0.93 ± 0.18**	0.23 ± 0.31$^-$	**0.95 ± 0.11**	0.22 ± 0.34$^-$	**0.84 ± 0.20**	0.66 ± 0.35$^-$

5.3. Experimental results

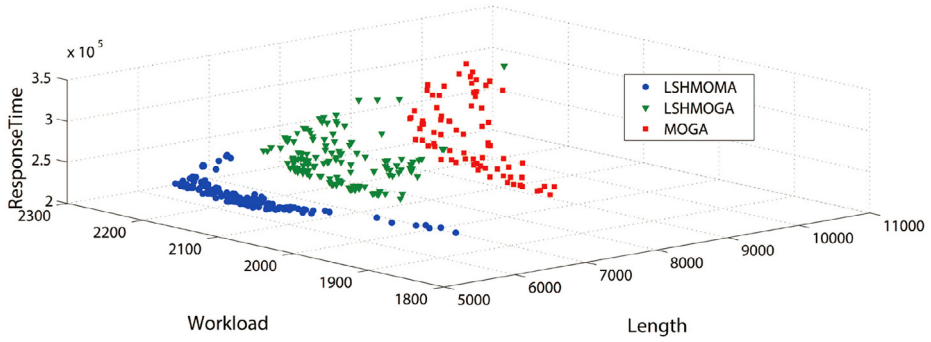
The mean results in terms of the best route length, best response time, best workload, convergence metric, and spacing metric of the three algorithms on the four benchmark DPDPs are summarized in Table 2. The results in terms of coverage-of-two-sets metric of the three algorithms are reported in Table 3. The approximate Pareto fronts obtained by the three algorithms from a single run are shown in Fig. 8. LSH-MOMA is shown to obtain better performance than the other two algorithms. Figs. 9–12 each shows four representative non-dominated solutions of LSH-MOMA in a single run in one DPDP. Among the four solutions, the first three are solutions of the best route length, the best response time, and the best workload, respectively. The last one is a centroid non-dominated solution that possesses the smallest sum of distance to all other non-dominated solutions. Detailed analysis and discussion of the results are provided as follows.

5.3.1. LSH-based methods vs. MOGA

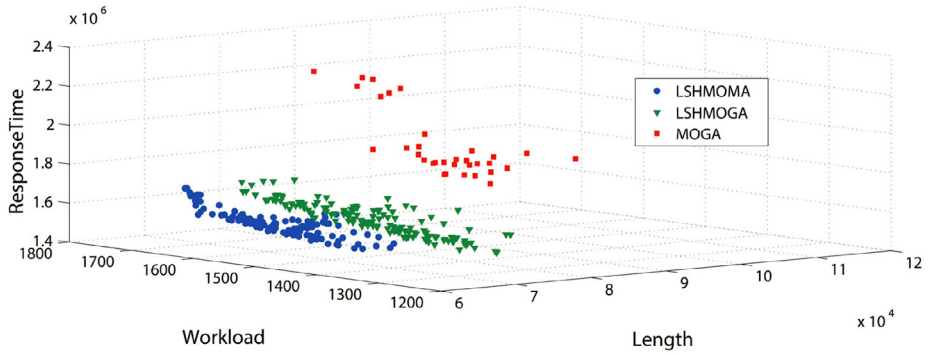
As shown in Tables 2 and 3, LSH-MOMA significantly outperforms MOGA in terms of all the six criteria. LSH-MOGA shows superiority to MOGA in the first five criteria. In terms of C2S-Metric, LSH-MOGA significantly outperforms MOGA in DPDP3 and DPDP4, whereas the two algorithms are comparable in DPDP1 and DPDP2. The final non-dominated solutions shown in Fig. 8 also indicate that MOGA fails to identify as many feasible solutions as the other two algorithms. Moreover, many solutions obtained by MOGA are dominated by that of the other two algorithms. The results suggest that the LSH-based rectification is capable of improving routing performance by searching more extensively in feasible solution space. Without the assistant of LSH-based rectification, MOGA wastes too much computational budge in searching unfeasible solutions, thus is inferior to the other two algorithms, especially in real-map cases which are more complex.



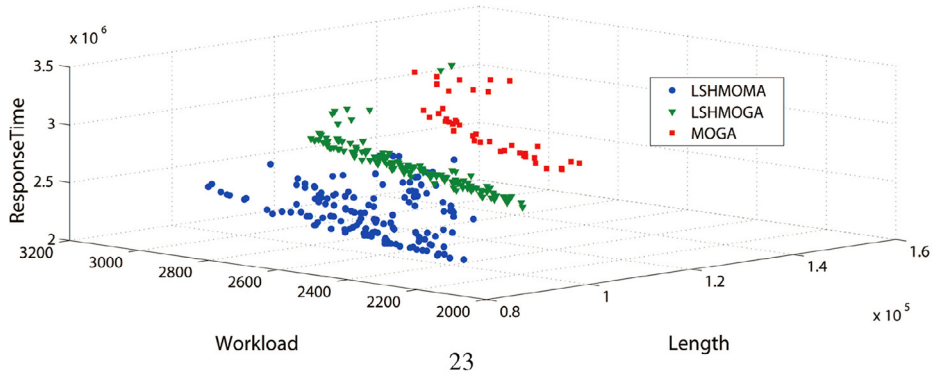
(a) DPDP1



(b) DPDP2

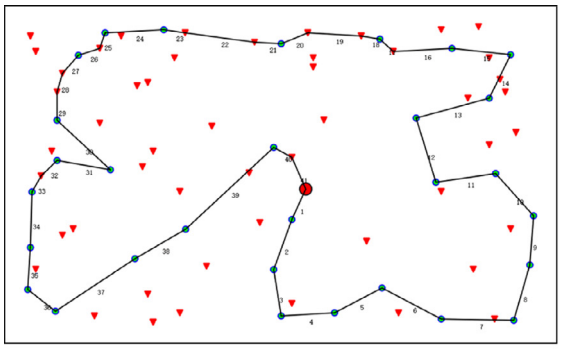


(c) DPDP3

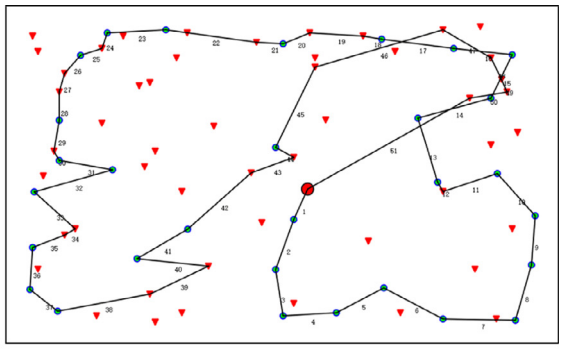


(d) DPDP4

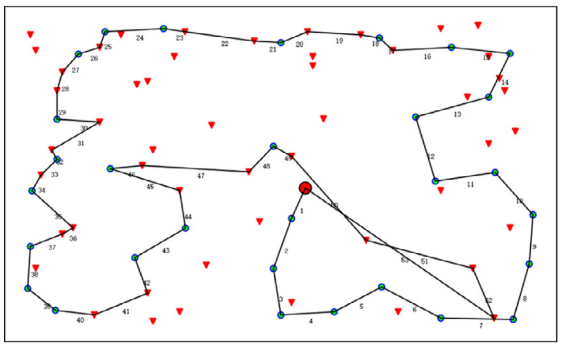
Fig. 8. The final non-dominated solutions of the three algorithms in a single run. The solutions of LSH-MOMA, LSH-MOGA, and MOGA are plotted with blue, green, and red spots, respectively. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).



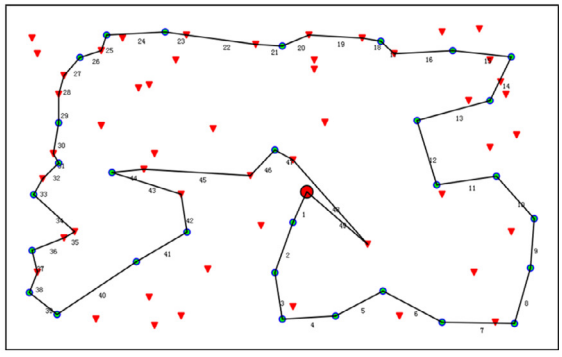
(a) The non-dominate solution of minimum path length



(b) The non-dominate solution of minimum response time

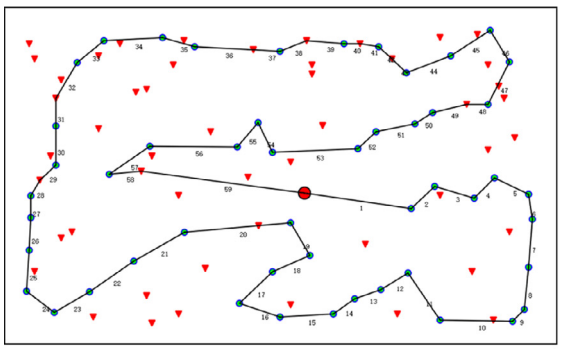


(c) The non-dominate solution of maximum workload

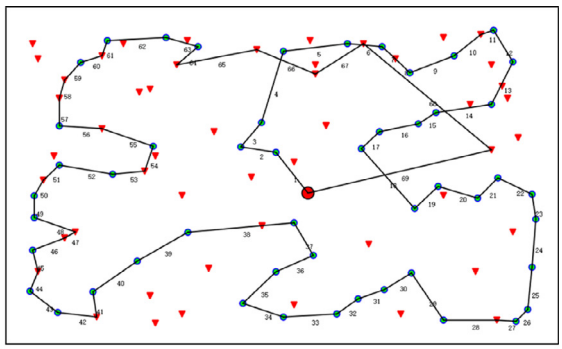


(d) The centroid solution of all non-dominate solutions

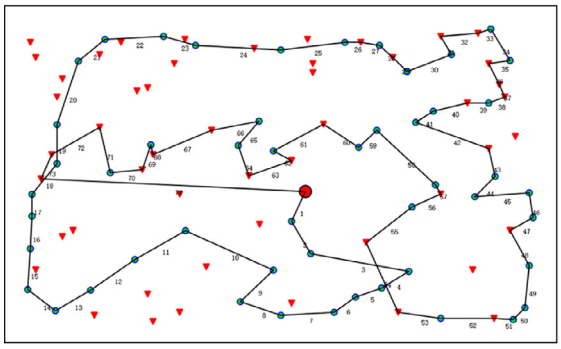
Fig. 9. Four representative non-dominated solutions of LSH-MOMA in DPDP1.



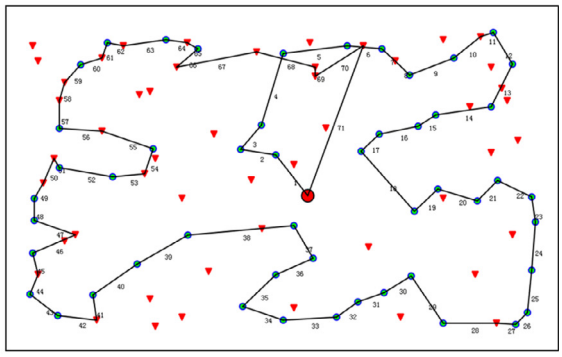
(a) The non-dominate solution of minimum path length



(b) The non-dominate solution of minimum response time

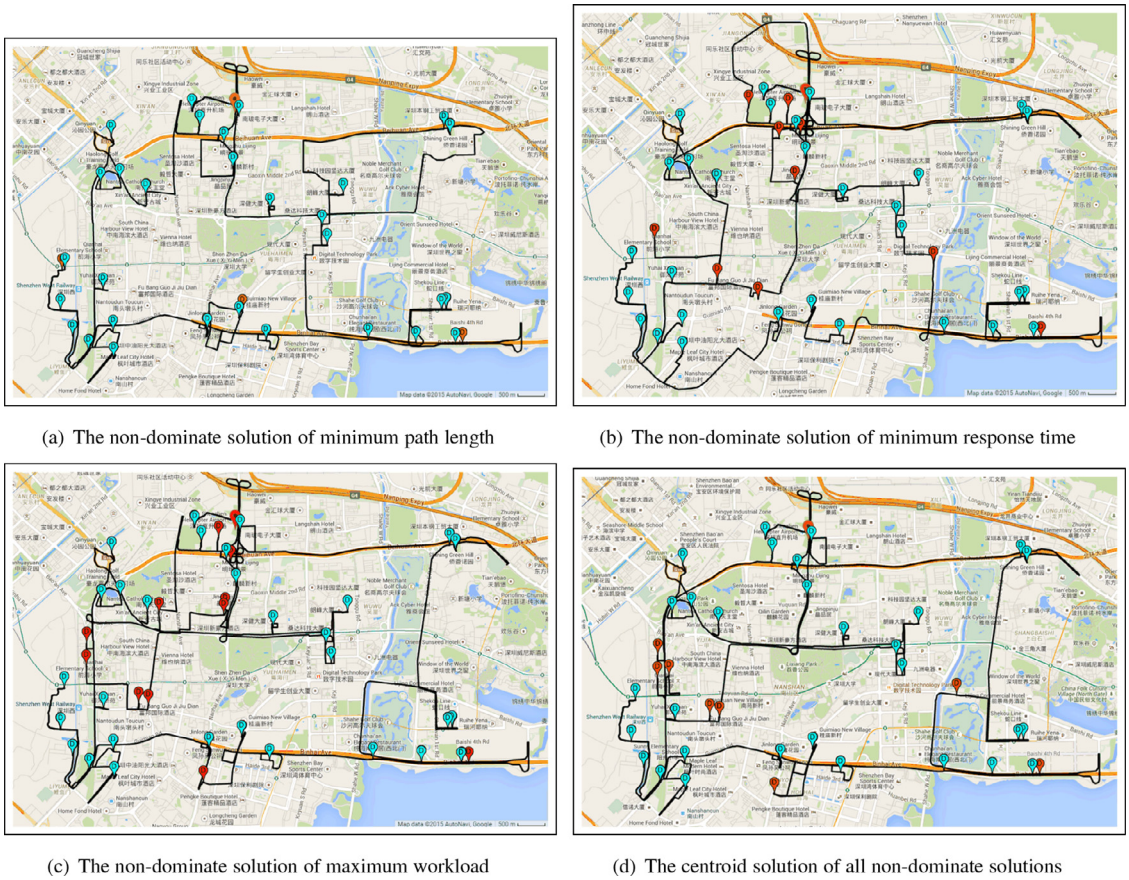


(c) The non-dominate solution of maximum workload



(d) The centroid solution of all non-dominate solutions

Fig. 10. Four representative non-dominated solutions of LSH-MOMA in DPDP2.



(a) The non-dominate solution of minimum path length

(b) The non-dominate solution of minimum response time

(c) The non-dominate solution of maximum workload

(d) The centroid solution of all non-dominate solutions

Fig. 11. Four representative non-dominated solutions of LSH-MOMA in DPDP3.

5.3.2. LSH-MOMA vs. LSH-MOGA

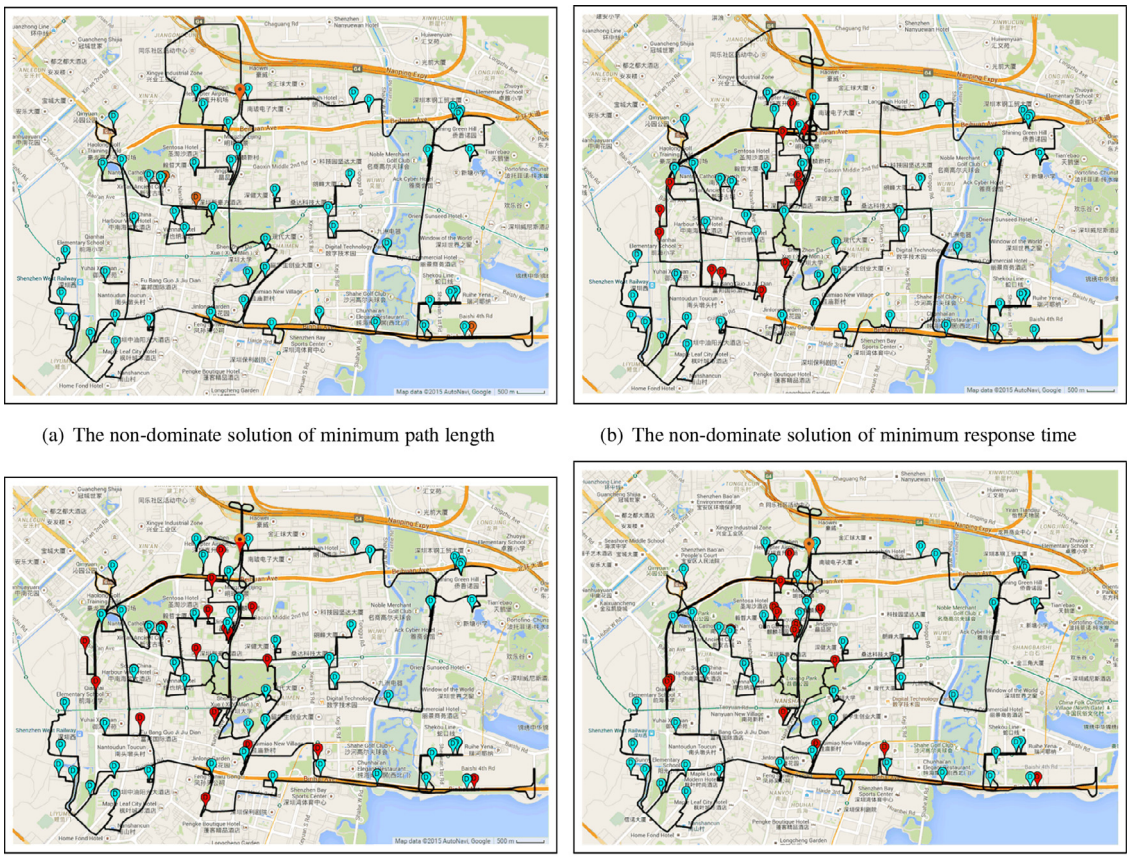
LSH-MOGA is different from LSH-MOMA for not using LSH-based local search. From the results shown in Tables 2, 3 and Fig. 8, it is observed that LSH-MOMA significantly outperforms LSH-MOGA in terms of the evaluation criteria expect the best workload, on which the two algorithms are competitive with each other. The observation demonstrates that LSH-based local search is efficient in reducing route length and response time. LSH-MOMA and LSH-MOGA achieve exactly the same maximum workloads in the four DPDPs over 25 independent runs (i.e., with zero standard deviation). Explanation for this observation is that it is relatively easy for both algorithms to reach the maximum workloads by keeping adding new dynamic nodes to the routes until the vehicle is filled to capacity. During the course, the LSH-based rectification acts to prevent the routes from capacity violation. Since both LSH-MOMA and LSH-MOGA manage to reach the extreme, LSH-MOMA does not show superiority to LSH-MOGA in terms of workload in these four cases. Traditional nearest neighbor identification methods is also applicable to the proposed routing framework, nevertheless, the LSH-based method has substantially reduced the computational time, which enables LSH-MOMA to scale up to larger map and handle more dynamic requests.

5.3.3. Effects of different routing environments

DPDPs on both Euclidean plane and real map are investigated in this section. The only difference between these two environments lies in the distance measure used. On Euclidean plane, the distance of two nodes is measured by Euclidean distance, whereas on real map the distance depends on the length of the available physical path connecting these two nodes. Two geographically near nodes could have a long distance in real map due to the lack of direct connection between them. LSH-MOMA outperforms the other two algorithms in both environments. The LSH-based rectification and local search are efficient and robust in identifying nearest neighbors of a node in both environments. As shown in Fig. 8, in the real-map cases namely DPDP3 and DPDP4, which are more complex than the ones on Euclidean plane, LSH-based methods generate even better results than MOGA.

5.3.4. Preference of different objectives

Optimizing different objectives leads to different optimal solutions. The results plotted in Figs. 9–12 show that the three extreme solutions of the best route length, response time, and workload involve different numbers of dynamic nodes. Particularly,



(a) The non-dominate solution of minimum path length

(b) The non-dominate solution of minimum response time

(c) The non-dominate solution of maximum workload

(d) The centroid solution of all non-dominate solutions

Fig. 12. Four representative non-dominated solutions of LSH-MOMA in DPDP4.

in DPDP1 and DPDP2, about 25, 40 and 50% dynamic nodes are included in the extreme solutions of the best route length, response time, and workload, respectively. In DPDP3 and DPDP4, the corresponding percentages are about 5, 20, and 30%, respectively. The number of served dynamic nodes does not change as the number of static nodes increases from 30 to 50, but it is affected by the preference of different objectives and the vehicle capacity. In a decision epoch, a user can choose any one tradeoff solution in the current Pareto optimal set in light of actual needs. LSH-MOMA responds more dynamic nodes on Euclidean plane than on real map. The reason for this observation is that the amount of dynamic pickup commodities in real-map cases is larger than the Euclidean plane cases. As a result, adding a dynamic node to a route in real-map cases is more likely to violate the vehicle capacity constraint.

6. Conclusions

This paper formulates a one-to-many-to-one dynamic pickup-and-delivery problem (1-M-1 DPDP) as a multi-objective optimization problem with three conflicting objectives, i.e., route length, response time, and workload. A multi-objective memetic algorithm based on locality-sensitive hashing (LSH-MOMA) is proposed to solve the problem. LSH-MOMA introduces LSH-based rectification and local search to a MOEA framework to repair and fine-tune the candidate solutions. The algorithm is tested and compared with the other two counterpart algorithms using four benchmark 1-M-1 DPDPs that are based on both Euclidean plane and real map. The experimental results demonstrate the efficiency of LSH-MOMA. As an early attempt to solve the problem, other factors like fleet management, time window constraint, real-time traffic information, and occasional events are not included for the sake of simplicity. However, our algorithm could be extended to incorporate more realistic objectives and constraints to make it more applicable to solve real-world problems.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China, under grants 61471246, 61171125, 61575125, and 61205092, the National Natural Science Foundation of China Joint Fund with Guangdong under grant U1201256, the NSFC-RS joint project under grants IE111069, the Guangdong Foundation of Outstanding Young Teachers

in Higher Education Institutions under grant Yq2013141, Guangdong Special Support Program of Top-notch Young Professionals under grant 2014TQ01X273, Shenzhen Scientific Research and Development Funding Program under grants JCYJ20130329115450637, KQC201108300045A, and ZYC201105170243A, Nanshan Innovation Institution Construction Program under grants KC2014ZDZJ0026A and KC2013ZDZJ0011A, Guangdong Natural Science Foundation under grant S2012010009545, and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, MOE of China, under grand 20111568.

References

[1] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, Commun. ACM 15 (1) (2008) 117–122.
[2] C.C. António, A memetic algorithm based on multiple learning procedures for global optimal design of composite structures, Memet. Comput. 6 (2) (2014) 113–131.
[3] M. Battarra, J.-F. Cordeau, M. Iori, Pickup-and-delivery problems for goods transportation, in: P. Toth, D. Vigo (Eds.), Vehicle Routing: Problems, Methods, and Applications, SIAM, 2014, pp. 161–191.
[4] I. Benyahia, J.-Y. Potvin, Decision support for vehicle dispatching using genetic programming, IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. 28 (3) (1998) 306–314.
[5] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, G. Laporte, Static pickup and delivery problems: a classification scheme and survey, TOP 15 (1) (2007) 1–31.
[6] G. Berbeglia, J.-F. Cordeau, G. Laporte, Dynamic pickup and delivery problems, Eur. J. Oper. Res. 202 (1) (2010) 8–15.
[7] F. Caraffini, F. Neri, L. Picinali, An analysis on separability for memetic computing automatic design, Inf. Sci. 265 (2014) 1–22.
[8] D. Cattaruzza, N. Absi, D. Feillet, J. González-Feliu, Vehiclerouting problems for city logistics, EURO J. Transp. Logist. Advanced access (2015) 1–29.
[9] X. Chen, Y.-S. Ong, M.-H. Lim, K.C. Tan, A multi-facet survey on memetic computation, IEEE Trans. Evolut. Comput. 15 (5) (2011) 591–607.
[10] B.K.-S. Cheung, K. Choy, C.-L. Li, W. Shi, J. Tang, Dynamic routing model and solution methods for fleet management with mobile technologies, Int. J. Prod. Econ. 113 (2) (2008) 694–705.
[11] C.A.C. Coello, G.B. Lamont, D.A.V. Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[12] C.E. Cortés, D. Sáez, A. Núñez, D. Muñoz-Carpintero, Hybrid adaptive predictive control for a dynamic pickup and delivery problem, Transp. Sci. 43 (1) (2009) 27–42.
[13] G.B. Dantzig, J.H. Ramser, The truck dispatching problem, Manag. Sci. 6 (1) (1959) 80–91.
[14] K. Deb, S. Jain, Running performance metrics for evolutionary multi-objective optimizations, in: Proceedings of the Fourth Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02), 2002, pp. 13–20.
[15] K. Deb, D. Kalyanmoy, Multi-Objective Optimization Using Evolutionary Algorithms, John Wiley & Sons, Inc., New York, NY, USA, 2001.
[16] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evolut. Comput. 6 (2) (2002) 182–197.
[17] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. 1 (4) (2006) 28–39.
[18] I.H. Dridi, R. Kammarti, M. Ksouri, P. Borne, et al., Multi-objective optimization for the m-PDPTW: aggregation method with use of genetic algorithm and lower bounds, Int. J. Comput. Commun. Control 6 (2) (2011) 246–257.
[19] J. Euchi, A. Yassine, H. Chabchoub, The dynamic vehicle routing problem: solution with hybrid metaheuristic approach, Swarm Evolut. Comput. 21 (2015) 41–53.
[20] M. Farina, K. Deb, P. Amato, Dynamic multiobjective optimization problems: test cases, approximations, and applications, IEEE Trans. Evolut. Comput. 8 (5) (2004) 425–442.
[21] A. Garcia-Najera, J.A. Bullinaria, An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows, Comput. Oper. Res. 38 (1) (2011) 287–300.
[22] S.F. Ghannadpour, S. Noori, R. Tavakkoli-Moghaddam, K. Ghoseiri, A multi-objective dynamic vehicle routing problem with fuzzy time windows: model, solution and application, Appl. Soft. Comput. 14 (2014) 504–527.
[23] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.
[24] L. Grandinetti, F. Guerriero, F. Pezzella, O. Pisacane, The multi-objective multi-vehicle pickup and delivery problem with time windows, Procedia-Soc. Behav. Sci. 111 (2014) 203–212.
[25] A. Haghani, S. Jung, A dynamic vehicle routing problem with time-dependent travel times, Comput. Oper. Res. 32 (11) (2005) 2959–2986.
[26] J.H. Holland, Adaptation in Natural Artificial Systems, 2nd ed., MIT Press, Cambridge MA, 1992.
[27] M.K. Islam, M. Chetty, Clustered memetic algorithm with local heuristics for ab initio protein structure prediction, IEEE Trans. Evolut. Comput. 17 (4) (2013) 558–576.
[28] P. Jaillet, M.R. Wagner, Generalized online routing: new competitive ratios, resource augmentation, and asymptotic analyses, Oper. Res. 56 (3) (2008) 745–757.
[29] N. Jozefowiez, F. Semet, E.-G. Talbi, Multi-objective vehicle routing problems, Eur. J. Oper. Res. 189 (2) (2008) 293–309.
[30] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Network (ICNN'95), 1995, pp. 1942–1948.
[31] N. Krasnogor, Studies on the Theory and Design Space of Memetic Algorithms, (Ph.D. thesis), University of the West of England, Bristol, U.K., 2002.
[32] B. Lacroix, D. Molina, F. Herrera, Region based memetic algorithm for real-parameter optimisation, Inf. Sci. 262 (2014) 15–31.
[33] J. Lee, D.-W. Kim, Memetic feature selection algorithm for multi-label classification, Inf. Sci. 293 (2015) 80–96.
[34] K. Li, Q. Zhang, S. Kwong, M. Li, R. Wang, Stable matching-based selection in evolutionary multiobjective optimization, IEEE Trans. Evolut. Comput. 18 (6) (2014) 909–923.
[35] Y. Mei, K. Tang, X. Yao, A memetic algorithm for periodic capacitated arc routing problem, IEEE Trans. Syst. Man. Cybern. Part B: Cybern. 41 (6) (2011) 1654–1667.
[36] K. Miettinen, Nonlinear Multiobjective Optimization, vol. 12, Springer Science & Business Media, 1999.
[37] P. Moscato, Memetic Algorithm: A Short Introduction. In New Ideas in Optimization, McGraw-Hill, London, 1999.
[38] D. Muñoz-Carpintero, D. Sáez, C.E. Cortés, A. Núñez, A methodology based on evolutionary algorithms to solve a dynamic pickup and delivery problem under a hybrid predictive control approach, Transp. Sci. 49 (2) (2015) 239–253.
[39] F. Neri, C. Cotta, Memetic algorithms and memetic computing optimization: a literature review, Swarm Evolut. Comput. 2 (2012) 1–14.
[40] T.T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: a survey of the state of the art, Swarm Evolut. Comput. 6 (2012) 1–24.
[41] A. Núñez, C.E. Cortés, D. Sáez, B. De Schutter, M. Gendreau, Multiobjective model predictive control for dynamic pickup and delivery problems, Control Eng. Pract. 32 (2014) 73–86.
[42] Y.S. Ong, M.H. Lim, X.S. Chen, Research frontier: memetic computation - past, present & future, IEEE Comput. Intell. Mag. 5 (2) (2010) 24–31.
[43] G. Pankratz, Dynamic Planning of Pickup and Delivery Operations by Means of Genetic Algorithms, FernUniv., Fachbereich Wirtschaftswiss, 2004.
[44] L.R. Pedro, R.H. Takahashi, Inspm: an interactive evolutionary multi-objective algorithm with preference model, Inf. Sci. 268 (2014) 202–219.
[45] V. Pillac, M. Gendreau, C. Guéret, A.L. Medaglia, A review of dynamic vehicle routing problems, Eur. J. Oper. Res. 225 (1) (2013) 1–11.
[46] D. Sáez, C.E. Cortés, A. Núñez, Hybrid adaptive predictive control for the multi-vehicle dynamic pick-up and delivery problem based on genetic algorithms and fuzzy clustering, Comput. Oper. Res. 35 (11) (2008) 3412–3438.
[47] M.W. Savelsbergh, M. Sol, The general pickup and delivery problem, Transp. Sci. 29 (1) (1995) 17–29.
[48] J.R. Schott, Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Technical Report, DTIC Document, 1995.
[49] X.-N. Shen, X. Yao, Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, Inf. Sci. 298 (2015) 198–224.

[50] A.G. Sutcliffe, D. Wang, Memetic evolution in the development of proto-language, Memet. Comput. 6 (1) (2014) 3–18.
[51] M.R. Swihart, J.D. Papastavrou, A stochastic and dynamic model for the single-vehicle pick-up and delivery problem, Eur. J. Oper. Res. 114 (3) (1999) 447–464.
[52] K.C. Tan, C.Y. Cheong, C.K. Goh, Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation, Eur. J. Oper. Res. 177 (2) (2007) 813–839.
[53] K.C. Tan, E.F. Khor, T.H. Lee, Multiobjective Evolutionary Algorithms and Applications, Springer-Verlag London, UK, 2005.
[54] K.C. Tan, L.H. Lee, Q. Zhu, K. Ou, Heuristic methods for vehicle routing problem with time windows, Artif. Intell. Eng. 15 (3) (2001) 281–295.
[55] H. Tang, M. Hu, Dynamic vehicle routing problem with multiple objectives: Solution framework and computational experiments, J. Trans. Res. Board 1923 (1) (2005) 199–207.
[56] K. Tang, Y. Mei, X. Yao, Memetic algorithm with extended neighborhood search for capacitated arc routing problems, IEEE Trans. Evolut. Comput. 13 (5) (2009) 1151–1166.
[57] L. Tersi, S. Fantozzi, R. Stagni, Characterization of the performance of memetic algorithms for the automation of bone tracking with fluoroscopy, IEEE Trans. Evolut. Comput. 19 (1) (2015) 19–30.
[58] C.-K. Ting, X.-L. Liao, The selective pickup and delivery problem: formulation and a memetic algorithm, Int. J. Prod. Econ. 141 (1) (2013) 199–211.
[59] Y. Wang, B. Li, T. Weise, Two-stage ensemble memetic algorithm: function optimization and digital IIR filter design, Inf. Sci. 220 (2013) 408–424.
[60] B. Yuan, B. Li, T. Weise, X. Yao, A new memetic algorithm with fitness approximation for the defect-tolerant logic mapping in crossbar-based nanoarchitectures, IEEE Trans. Evolut. Comput. 18 (6) (2014) 846–859.
[61] Z. Zhu, URL, 2015. Accessed September 23. http://csse.szu.edu.cn/staff/zhuzx/dpdpdata.zip.
[62] Z. Zhu, S. Jia, S. He, Y. Sun, Z. Ji, L. Shen, Three-dimensional gabor feature extraction for hyperspectral imagery classification using a memetic framework, Inf. Sci. 298 (2015) 274–287.
[63] Z. Zhu, Y.-S. Ong, J.-L. Kuo, Feature selection using single/multi-objective memetic frameworks, in: C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.), Multi-Objective Memetic Algorithms, Springer, 2009, pp. 111–131.
[64] Z. Zhu, J. Zhou, Z. Ji, Y.-H. Shi, DNA sequence compression using adaptive particle swarm optimization-based memetic algorithm, IEEE Trans. Evolut. Comput. 15 (5) (2011) 643–658.
[65] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms a comparative case study, in: Parallel Problem Solving from Nature (PPSN V), Springer, 1998, pp. 292–301.
[66] S. Zou, J. Li, X. Li, A hybrid particle swarm optimization algorithm for multi-objective pickup and delivery problem with time windows, J. Comput. 8 (10) (2013) 2583–2589.