R2 Indicator based Multiobjective Memetic Optimization for the Pickup and Delivery Problem with Time Windows and Demands (PDP-TW-D)

Dũng H. Phan
Department of Computer Science
University of Massachusetts, Boston
Boston, MA, 02125-3393, USA
phdung@cs.umb.edu

Junichi Suzuki
Department of Computer Science
University of Massachusetts, Boston
Boston, MA, 02125-3393, USA
jxs@cs.umb.edu

ABSTRACT

This paper defines a multiobjective variant of the Pickup and Delivery Problem (PDP), called PDP with Time Windows and Demands (PDP-TW-D), and approaches the problem with a novel memetic optimization algorithm. With respect to multiple optimization objectives, the goal of PDP-TW-D is to find a set of Pareto-optimal routes for a fleet of vehicles in order to serve given transportation requests. This paper considers five optimization objectives for PDP-TW including the number of vehicles, the total travel distance and the total waiting time. The proposed algorithm is designed as an evolutionary multiobjective optimization algorithm that is augmented by a local search algorithm. It uses the population of individuals, each of which represents a solution candidate, and evolves them via genetic operators (e.g., crossover and mutation operators) through generations. In addition to this global search process, it allows individuals to improve themselves in each generation through local search. Experimental results show that the global and local search processes complement to improve convergence speed and can effectively obtain quality trade-off solutions with respect to conflicting objectives.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms

Keywords

Multiobjective optimization, Memetic algorithms, Pickup and Delivery Problem

BICT 2014, December 01-03, Boston, United States

DOI 10.4108/icst.bict.2014.258229

1. INTRODUCTION

This paper formulates a new multiobjective optimization problem, the Pickup and Delivery Problem with Time Windows and Demands (PDP-TW-D). PDP-TW-D is a combinatorial optimization problem of finding a set of optimal routes for a fleet of vehicles in order to serve given transportation requests. Each transportation request is defined by a pickup location, a delivery location, goods to be transported, a pickup time window and a delivery time window. Each pickup/delivery location needs to be visited by a vehicle within a certain time window. Each vehicle has a capacity to load goods and follows an assigned route (i.e., a sequence of pickup/delivery locations) by loading goods at pickup locations and unloading them at delivery locations. This paper considers five optimization objectives for PDP-TW-D including the number of vehicles used to fulfill transportation requests, the total distance that vehicles travel, and the total amount of goods that vehicles transport. With respect to these objectives, PDP-TW-D is to find the Pareto-optimal routes for vehicles to travel to fulfill all given transportation requests.

In PDP-TW-D, all vehicles must satisfy four types of constraints: time window constraints, precedence constraints, pairing constraints and capacity constraints. Time window constraints enforce vehicles to visit pickup and delivery locations within given time windows. Each time window is a pair of the earliest possible arrival time and the latest possible departure time. Precedence constraints enforce vehicles to visit pickup locations prior to their corresponding delivery locations. Pairing constraints restrict that one vehicle has to do both the pickup and delivery of goods specified in a transportation request. Capacity constraints enforce vehicles not to exceed its capacity by overloading goods.

PDP-TW-D extends two existing problems, the Pickup and Delivery Problem (PDP) and the PDP with Time Windows (PDP-TW), which in turn extend the Vehicle Routing Problem. As a variant of PDP and PDP-TW, PDP-TW-D represents a large number of logistics and transportation applications such as airlift and sealift [1, 3], parcel services, taxi dispatching [19] and dial-a-ride services (e.g., the transport of the elderly and handicapped among their homes, hospitals and other locations) [10, 18].

Given the facts that PDP and PDP-TW are NP-hard [15], PDP-TW-D is NP-hard as well. Therefore, this paper studies a stochastic heuristic algorithm to approach PDP-TW-D. The proposed algorithm, R2-IBEA-LS, is designed as an evo-

lutionary multiobjective optimization algorithm (EMOA) that is augmented by a local search algorithm. It uses the population of individuals, each of which represents a solution candidate, and evolves them via genetic operators (e.g., crossover and mutation operators) through generations to seek the Pareto-optimal solutions with respect to given multiple objectives. In addition to this global search process, R2-IBEA-LS allows individuals to improve themselves in each generation through local search. Experimental results show that the global and local search processes complement with each other in R2-IBEA-LS and the local search process contributes to expedite its convergence. R2-IBEA-LS can effectively obtain quality trade-off solutions to PDP-TW-D in a relatively large-scale problem instance that has 100 pickup and delivery locations. It outperforms two existing EMOAs.

2. PROBLEM STATEMENT

This paper uses the following notations to define PDP-TW-D.

- $P = \{1, 2, ..., n\}$ is the set of pickup nodes. $D = \{n + 1, n+2, ..., 2n\}$ is the set of delivery nodes. $\{0\}$ denotes the depot.
- $N = \{1, 2, ..., n\}$ is the set of transportation requests. A request i is represented by a pickup node i, a delivery node $i+n$ and a demand q_i. q_i is the amount of goods that are available at i to be picked up and delivered to $i + n$.
- K is the set of vehicles. $|K| = m$. Each vehicle has its capacity Q.
- For all nodes $i, j \in P \cup D$, d_{ij} denotes the travel distance from i to j. t_{ij} denotes the travel time from i to j.
- Each node $i \in P \cup D$ has an associated time window $[e_i, l_i]$, during which a vehicle is required to visit i to load or unload goods. Service time s_i is also associated with i. It is the time required to load/unload goods to/from a vehicle at i. A vehicle is allowed to arrive at i before e_i; however, it needs to wait for starting to load/unload goods until e_i.
- A pickup and delivery (PD) route for a vehicle k, R_k, is a sequence of nodes that k visits, starting and ending with the depot 0. Every PD route is required to satisfy three types of constraints: (1) *capacity constraints* (The amount of goods loaded on each vehicle should not exceed the capacity of the vehicle.), (2) *time window constraints* (Each vehicle should arrive at a node before the end of its time window.), and (3) *precedence constraints* (A pickup node i should be visited before its corresponding delivery node $i + n$).
- A pickup and delivery (PD) plan is a set of PD routes, $R = \{R_k | k \in K\}$. A PD plan may not fulfill all the n requests. No redundant nodes exist in R except the depot. (A non-depot node is not visited more than once.)

PDP-TW-D is to find the Pareto-optimal PD plans with respect to the following five objectives.

- The number of vehicles used in a PD plan (i.e., the number of routes in a PD plan). This objective is to be minimized. It is computed as follows:

$$f_{vehicle} = |R| \tag{1}$$

- The total travel distance. This objective is to be minimized. It is computed as follows:

$$f_{distance} = \sum_{k \in K} \sum_{i \in P \cup D} \sum_{j \in P \cup D} d_{ij} x_{ijk} \tag{2}$$

  x_{ijk} is true (1) iff the vehicle k travels from the node i to the node j; otherwise, x_{ijk} is false (0).

- The total demands. This objective is to be maximized. It represents the total amount of goods that are transported in a PD plan. It is computed as:

$$f_{demand} = \sum_{k \in K} \sum_{i \in N} q_i z_{ik} \tag{3}$$

  z_{ik} is true (1) iff the vehicle k fulfills the request i; otherwise, z_{ik} is true (0).

- The total waiting time. This objective is to be minimized. It represents the total time that vehicles have to wait if they arrive at a node before starting time of of its time window. It is computed as:

$$f_{wait} = \sum_{k \in K} \sum_{i \in P \cup D} w_{ik} \tag{4}$$

  w_{ik} is the waiting time of vehicle k-th at node i-th. It will be zero if the vehicle k does not visit node i-th, or it arrive on/after the starting time of node i-th.

- The travel distance of the longest route. This objective is to be minimized. It is computed as:

$$f_{max} = \max_{k \in K} \sum_{i \in P \cup D} \sum_{j \in P \cup D} d_{ij} x_{ijk} \tag{5}$$

  x_{ijk} is true (1) iff the vehicle k travels from the node i to the node j; otherwise, x_{ijk} is false (0).

Objectives conflict with each other in PDP-TW-D. For example, the first two objectives conflict with the third one. A smaller number of vehicles and a shorter travel distance yields lower total demands. On the contrary, higher total demand yield a larger number of vehicles and a longer travel distance.

Constraint violations are computed as follows:

- Capacity violation of a vehicle k in a PD route R_k

$$g_c(R_k) = \sum_{i \in R_k} \Delta_i \tag{6}$$

  Δ_i denotes the amount of overloaded goods on k at i.

- Time window violation of a vehicle k in a PD route R_k

$$g_{tw}(R_k) = \sum_{i \in R_k} |t_{ik} - l_i| I_{ik} \tag{7}$$

  t_{ik} denotes the time when k arrives at i. $I_{ik} = 1$ if $t_{ik} > l_i$, otherwise $I_{ik} = 0$.

3. RELATED WORK

Nanry and Barnes [11] is one of the first to present a metaheuristic for PDPTW. The metaheuristic is based on a reactive tabu search. First, a feasible solution is constructed using greedy insertion method. Next, tabu search is used to improve the initial solution. Three neighborhood moves are proposed in this paper. They are: Single Pair Insertion (SPI), Swapping Pairs Between Route (SBR) and Within Route Insertion (WRI). In order to evaluate their work, the authors created PDPTW test instances from standard VRPTW problems proposed by Solomon [16].

Li and Lim [9] propose a tabu-embedded simulated annealing approach to solve PDPTW. The authors modify Solomon's insertion heuristic [16] by initializing each route with a pickup and delivery pair which satisfies some criteria (e.g. early time window interval, far distance from depot). Three different neighborhood moves (i.e. PD-Shift, PD-Swap, PD-Rearrange) are presented. The authors extend local search method to a descent local search (DLS) which tries to improve the current solution for a number of iterations. After a given number of iterations without improvement, the search is restarted from current best solution. To avoid cycling, a tabu list is used to keep track of the recently investigated solutions. Additionally, the authors generated 56 test instances for PDPTW problem based on all 56 Solomon VRPTW instances. The generated data set became the standard benchmark test data for PDPTW. Another tabu search based approach is presented by Lau and Liang [8] . Several construction heuristics (i.e.. insertion heuristic, sweep heuristic, portioned insertion heuristic) are investigated in this paper

Bent and Van Hentenryck [2] propose a two-stage hybrid algorithm for MV-PDPTW. At the first stage, a simple simulated annealing algorithm is applied to minimize the number of vehicles. A Large Neighborhood Search (LNS) is used to minimize the total travel cost of the solution at second stage. An extension of LNS, called adaptive LNS is also presented in [15]. The adaptive LNS uses several removal and insertion heuristics are used during the same search while normal LNS uses only one method for removal and one method for insertions. In each iteration, a removal or insertion heuristic is chosen based on its adaptive weight value. The experiment results on benchmark data [9] show the effectiveness the LNS based approaches since they were able to produce many new best solutions.

Besides Simulated Annealing, Tabu Search and LNS, genetic algorithm has also been applied to solve PDPTW in some studies. CrÃľput et al [4] present an evolutionary algorithm to solve PDPTW. The individual is represented by a list of vehicles routes, where each route consists a sequence of pickup and delivery pairs. The fitness function is to minimize the number of vehicles and travel cost. Two crossover operators are investigated. The first one exchanges fragments of routes between parents, while the other exchanges complete routes. The mutation operators are designed to reduce the number of routes by merging two routes and improve a route by rearranging its nodes.

Pankratz [12] applies a Grouping Genetic Algorithm (GGA) to PDPTW. In this study, individual is represented as a set of genes, each gene represents a group of requests that are assigned to one vehicles. The crossover operator removes vehicles from one parent and inserts into the other parent. Then, a cleaning up procedure is executed to remove duplicate vehicles and the repeated requests. Ding et al [6] also uses GGA with three different routing adjustment strategies.

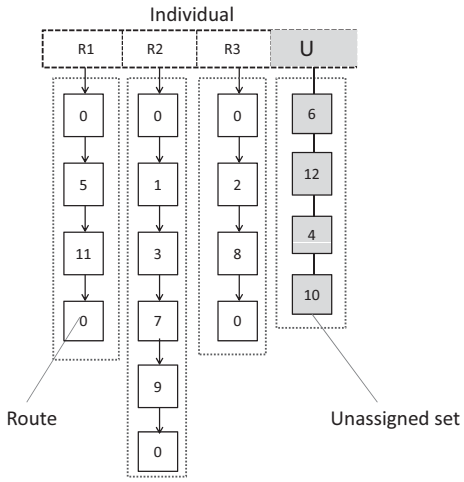


Figure 1: An Example Individual

Takoudjou et al [17] considers a new variant of PDP which is the pickup and delivery problem with transshipment (PDPT). In this variant, a request can be satisfied by more than one vehicle. One vehicle can pick up a load at the pick-up node and drops it off at a transshipment node. Then, other vehicle can carry and deliver this load to the delivery node. The problem is solved using multi-start hybrid heuristic. At each iteration of the heuristic, a PDP solution is created. After that, it is destroyed and repaired to obtain a PDPT solution. The objective function is a weighted function of traveling distance, number of vehicle and the square of number nodes visited by each vehicle. Other variant of VRP is Sumultaneous Delivery and Pickup Problem with Time Window (SDPPTW). In the SDPPTW problem, customers require both forward supply services and reverse recycling services. The vehicles collect recycled materials at customer node and return them to collection center. Wang and Chen [20] uses co-evolution genetic algorithm to solve SDPPTW. The GA maintains two populations for two purposes: diversification and intensification.

4. THE PROPOSED ALGORITHM: R2-IBEA-LS

4.1 Individual Representation

In R2-IBEA-LS, each individual is a variable-length representation of routes . It encodes the number of routes and the order of nodes visited by each vehicle. Each individual also has a set of unassigned nodes. Figure 1 shows an example individual. There are 6 requests in this example. The pickup-delivery (PD) pairs are: (1,7), (2,8), (3,9), (4,10), (5,11) and (6,12). The individual forms a routing plan which has three routes, starting and ending at the depot 0. There are two pairs (4,10), (6,12) are not served in this routing plan.

4.2 Algorithmic Structure

Algorithm 1 shows the algorithmic structure of R2-IBEA-LS. It follows the optimization process in R2-IBEA [14].

Algorithm 1 Optimization Process in the Proposed EMOA

1: $g = 0$
2: $\mathcal{P}_g$ = initializePopulation(μ)
3: **while** g < g_{max} **do**
4: $\mathcal{O}_g = \emptyset$
5: **while** $|\mathcal{O}_g| < \mu$ do **do**
6: p_1 =binaryTounament($\mathcal{P}_g$)
7: p_2 =binaryTounament($\mathcal{P}_g$)
8: **if** random() $\leq P_c$ **then**
9: $\{o_1, o_2\}$ = crossover(p_1, p_2)
10: **if** random() $\leq P_m$ **then**
11: o_1 = mutation(o_1)
12: **end if**
13: **if** random() $\leq P_m$ **then**
14: o_2 = mutation(o_2)
15: **end if**
16: doLocalSearch(o_1)
17: doLocalSearch(o_2)
18: $\mathcal{O}_g = \{o_1, o_2\} \cup \mathcal{O}_g$
19: **end if**
20: **end while**
21: $\mathcal{R}_g = \mathcal{P}_g \cup \mathcal{O}_g$
22: Update the reference point z^*
23: Calculate the fitness of each individual $x_i \in \mathcal{R}_g$ as: $F(x_i) = \sum_{y_i \in \mathcal{R}_g \setminus \{x_i\}} -e^{-I_{R2}(y_i, x_i)/\kappa}$
24: **while** $|\mathcal{R}_g| > \mu$ **do**
25: $x^* = \arg\min_{x_i \in \mathcal{R}_g} F(x_i)$
26: $\mathcal{R}_g = \mathcal{R}_g \setminus \{x^*\}$
27: Update the fitness of each individual $x_i \in \mathcal{R}_g$ as: $F(x_i) = F(x_i) + e^{-I_{R2}(x^*, x_i)/\kappa}$
28: **end while**
29: $g = g + 1$
30: **end while**

Algorithm 2 generateInitialIndividual

1: **function** generateInitialIndividual()
2: Create s, an individual with one empty route
3: R = a set of all requests sorted in random order
4: **for** each request $d \in R$ **do**
5: **for** each route r in s **do**
6: **if** r is empty **then**
7: Add d into r
8: **else**
9: Find the location in r such that the traveling distance is minimum and no constraint violations
10: Add d to that route
11: **end if**
12: **end for**
13: If d could not be added to any available routes, create new empty route and add d into it
14: **end for**
15: return s
16: **end function**

At the 0-th generation, N individuals are generated as the initial population $\mathcal{P}_0$ (Line 2). In N individuals, δ percent are randomly generated. Each of them has a random number of routes. Each routes in the individual contains a randomly-selected pickup-delivery pairs in a random order. A correction procedure is executed to swap the potions of pickup node and delivery node if the delivery nodes is visited before its corresponding pickup node is. The remaining individuals are generated using Algorithm 2

At each generation (g), two parent individuals (p_1 and p_2) are selected from the current population $\mathcal{P}_g$ with binary tournaments (Lines 6 and 7). A binary tournament randomly takes two individuals from $\mathcal{P}_g$, compares them based on α-dominance relationship, and chooses a superior one as a parent.

With the crossover rate P_c, two parents reproduce two offspring with a crossover operator (Lines 8 to 9). Each offspring performs mutation with the mutation rate P_m (Lines 11 to 16). After that, local search is applied for both offspring to improve their quality. The binary tournament, crossover, mutation and local search operators are executed repeatedly on $\mathcal{P}_g$ to reproduce N offspring. The offspring ($\mathcal{O}_g$) are combined with the parent population $\mathcal{P}_g$ to form $\mathcal{R}_g$ (Line 19).

Environmental selection follows offspring reproduction (Line 22 to 28). In Line 22, reference point (z^* is updated using the same method (i.e. Adaptive Reference Point Adjustment) proposed in [14]. In Line 23 the fitness of each individual in $\mathcal{R}_g$ is calculated by applying the individual's I_{R2} value to an exponential amplification function. Then, the worst individual (i.e., the one with the lowest fitness) is removed from $\mathcal{R}_g$ (Lines 25 and 26). In Line 2, fitness is recalculated for each of the remaining7 individuals in $\mathcal{R}_g$. By repeating this removal process until $|\mathcal{R}_g| = \mu$, R2-IBEA selects μ individuals from $\mathcal{R}_g$ as the individuals to be used in the next generation ($g + 1$).

In order to handle two contraints, this paper uses constrained R2 binary indicator instead of original R2 binary indicator [14]. It takes two individuals (x and y) as input. It is defined based on the feasibility of individuals and R2 indicator [14] as follows:

- if x and y are both feasible

$$I^c_{R_2}(x, y) = R_2(\{x\}) - R_2(\{x \cup y\}) \tag{8}$$

  In this case, R_2 is calculated in objective space.
- if x is feasible y is not.

$$I^c_{R_2}(x, y) = 0 \tag{9}$$

$$I^c_{R_2}(y, x) = R_2(\{x\}) \tag{10}$$

- if both x and y are infeasible

$$I^c_{R_2}(x, y) = R^c_2(\{x\}) - R^c_2(\{x \cup y\}) \tag{11}$$

  R^c_2 is R_2 indicator in constraint space. It is calculated as normal R2 indicator except that it uses constraint space instead.

4.3 Crossover

This paper adopts the group-oriented crossover operator proposed in [12] as its crossover operator. The crossover operator is illustrated in Figure 2.

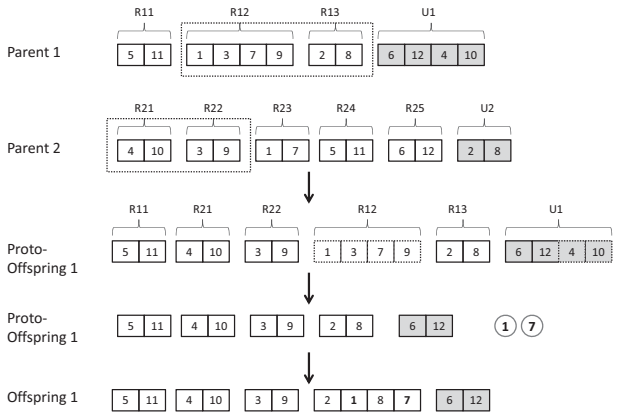


Figure 2: An Example of Crossover Process

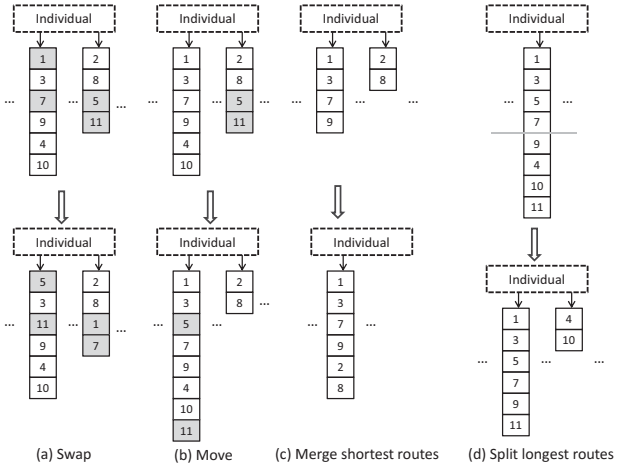


Figure 3: Mutation operators

The operator first selects two crossing points in each of the two parent individual at random. In the example in Figure 2, parent 1's crossing section is {R12,R13} and parent 2's is {R21,R22}. Next, the routes in the crossing section of the second parent are inserted in the parent 1 at the first crossing point. In figure 2, the routes {R22, R23} in parent 2 are inserted into parent 1 right before R12 of parent 1. As a result of this operation, a number PD pairs are duplicated in the proto-offspring (PD pairs: (3, 9), (4, 10)). In order to solve this problem, all parent 1's routes which contain duplicated PD pairs are removed (route R12). If duplicated PD pairs are found in unassigned set of parent 1, they will be deleted from the unassigned set. Note that, at this step all routes imported from parent 2 are left unchanged. Finally, the PD pairs which are belonged to the removed routes of parent 1 but are not contained in the imported routes (PD pair: (1, 7)) are reinserted in random order into existing routes (R11, R21, R22, R13). The reinsertion process can be described as follows: For each PD pair to be reinserted, two random positions at a random existing route are examined. If the insertion does not make any increment in time window violation, the insertion is accepted. Otherwise, other positions are examined. If all possible insertions make increment in time window violation, a new route is created for the current PD pair. In figure 2, the PD pair (1,7) is inserted into the R13 of parent 1.

4.4 Mutation

The proposed EMOA uses the following seven mutation operators. (See Fig. 3 for four of them.):

- *Add*: randomly chooses a PD pair from the unassigned set and inserts its pickup-node and delivery node to a randomly-selected positions in a randomly-selected route. This operator ensures that the pickup node is visited before the delivery node.
- *Delete*: removes a randomly-selected PD pair from a randomly-selected route and put it into the unassigned set.
- *Exchange*: randomly chooses a PD pair in a randomly-selected route and replaces it with a pair selected from the unassigned set randomly.
- *Swap*: exchanges the positions of two randomly-selected PD pairs in a route which is also selected randomly.
- *Move*: randomly removes a PD pair from a route and reinsert it into a randomly-selected routes at random positions.
- *Merge the shortest routes*: identifies the two shortest routes and appends one of them to the other.
- *Split the longest route*: identifies the longest route and splits it to two routes at a randomly-chosen point. If there are PD pairs of which pick up node and delivery node are contained in different routes, the delivery node will be moved to the route which contains its corresponding pickup node.

The proposed EMOA classifies these seven mutation operators to two categories. The first category consists of *Add*, *Delete* and *Exchange*. They exhibit the interactions between one route and the unassigned set. The remaining three operators are in the second category. They exhibit the interactions between two routes in the individual. These two categories have the same probably to be used. In each category, mutation operators are selected randomly.

4.5 Local Search

Local search is designed to improve the routes in the individual by swapping the positions of nodes within their respective route. The swapping has to ensure the precedence constraints of PD pairs. Algorithm 3, 4 show how local search improve an individual. The current route and new route are evaluated using R_2 indicator (Lines 2 and 9, Algorithm 4). The new route will replace the current route only when the new route has better $R2$ indicator value (i.e. smaller) and less constraint violations than the current route.

5. EXPERIMENTAL EVALUATION

This section evaluates R2-IBEA-LS through experiments with a well-known PDP-TW problem instance, called lc109 [9].

Algorithm 3 Pseudocode of Local Search

```
1: function doLocalSearch(o)
2: c = true
3: while c=true do
4:     for each route r ∈ o do
5:         c = improveRouteBySwapping(r)
6:     end for
7: end while
8: end function
```

Algorithm 4 improveRouteBySwapping

```
1: function improveRoute(r)
2: N = a set of all nodes in r sorted in random order
3: r2 = R2(r)
4: for each n ∈ N do
5:     for each node m, m is visited after n in r do
6:         Generate a route r' by swapping the positions of n
           and m
7:         if r' does not violate precedence constraint then
8:             if r' violates capacity and time window con-
               straints less or equal to r then
9:                 r2' = R2(r')
10:                if (r2' < r2) then
11:                    r = r'
12:                    return true
13:                end if
14:            end if
15:        end if
16:    end for
17: end for
18: return false
19: end function
```

Table 1 shows a set of parameter values used in the experiments. R2-IBEA-LS is compared with two EMOAs: (1) R2-IBEA, which disables the local search operator of R2-IBEA-LS and (2) a variant of NSGA-II [5], which is equipped with custom genetic operators tailored to PDP-TW-D [13]. All experiments were conducted with jMetal [7]. Every experimental result is obtained and shown based on 20 independent runs.

Parameter	Value
Population size	100
Max Generations	1000
Crossover rate	0.9
Mutation rate	0.3

Table 1: Parameter Configurations

5.1 Evaluation Metrics

This paper uses two evaluation metrics: hypervolume (HV) and $\mathcal{C}$-metric.

HV is the union of the volumes that non-dominated individuals dominate [21]. (It is computed with the reference point whose coordinate consists of the maximum objective values.) Thus, HV quantifies the optimality and diversity of non-dominated individuals. A higher HV indicates that non-dominated individuals are closer to the Pareto front and more diverse in the objective space.

$\mathcal{C}$-metric compares two sets of non-dominated individuals [22]. Given non-dominated individual sets A and B, $\mathcal{C}(A, B)$ measures the fraction of individuals in B that at least one individual in A dominates:

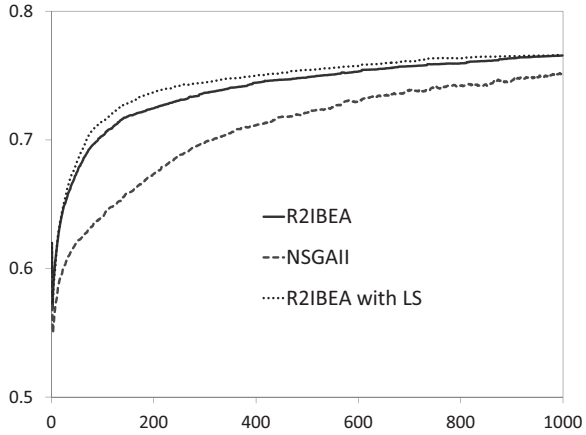


Figure 4: Hypervolume over Generations

$$\mathcal{C}(A, B) = \frac{|\{b \in B \mid \exists a \in A : a \succ b\}|}{|B|} \tag{12}$$

5.1.1 Evaluation Results

Figure 4 shows how HV changes over generations. Both R2-IBEA-LS and R2-IBEA rapidly increase HV values in the first 100 generations and gradually improve them after that. R2-IBEA-LS yields a higher HV value than R2-IBEA particularly in earlier generations in an experiment while the two algorithms yield very similar HV values. This result demonstrates that the local search process in R2-IBEA-LS expedites its convergence. NSGA-II is slower than R2-IBEA-LS and R2-IBEA in convergence and consistently yields a lower HV value than the two algorithms. Table 2 shows the average HV value that each algorithm yields at the last generation. A value in parentheses is a standard deviation. Figure 4 and Table 2 demonstrate that R2-IBEA-LS outperforms R2-IBEA and NSGA-II.

Table 3 compares three algorithms using the $\mathcal{C}$-metric. 84% of R2-IBEA individuals are dominated by at least one R2-IBEA-LS individual ($\mathcal{C}$(R2-IBEA-LS, R2-IBEA)=0.84). Conversely, 24% of R2-IBEA-LS individuals are dominated by at least R2-IBEA individual ($\mathcal{C}$(R2-IBEA, R2-IBEA-LS)=0.24). A similar observation is obtained in a comparison of R2-IBEA-LS with NSGA-II ($\mathcal{C}$(R2-IBEA-LS, NSGA-II)>$\mathcal{C}$(NSGA-II, R2-IBEA-LS)). Consistent with the results in Figure 4 and Table 2, Table 3 demonstrates that R2-IBEA-LS outperforms R2-IBEA and NSGA-II.

Figure 5 shows two-dimensional objective spaces that plot individuals obtained at the last generation. R2-IBEA-LS successfully reveals the relationships among optimization objectives and clearly exhibits the trade-offs among non-dominated individuals. This figure also show that R2-IBEA-LS individuals outperform NSGA-II individuals.

6. CONCLUSION

This paper formulates a multiobjective combinatorial optimization problem, PDP-TW-D, and approaches it with a novel memetic algorithm called R2-IBEA-LS. R2-IBEA-LS is designed as an evolutionary multiobjective optimization

Table 2: Comparison of R2-IBEA, R2-IBEA-LS and NSGA-II with Hypervolume

R2-IBEA-LS	R2-IBEA	NSGA-II
0.7662 (0.0325)	0.7656 (0.033)	0.7508 (**0.0193**)

Table 3: Comparison of R2-IBEA, R2-IBEA-LS and NSGAII with $\mathcal{C}$-metric

$\mathcal{C}$(R2-IBEA-LS,R2-IBEA)	**0.84**
$\mathcal{C}$(R2-IBEA, R2-IBEA-LS)	0.24
$\mathcal{C}$(R2-IBEA-LS, NSGA-II)	**0.85**
$\mathcal{C}$(NSGA-II, R2-IBEA-LS)	0.41

algorithm (EMOA) that is augmented by a local search algorithm. Experimental results show that the global and local search processes complement with each other in R2-IBEA-LS and the local search process contributes to expedite its convergence. R2-IBEA-LS can effectively obtain quality trade-off solutions to PDP-TW-D in a relatively large-scale problem instance that has 100 pickup and delivery locations. It outperforms two existing EMOAs.

References

[1] S. F. Baker, D. P. Morton, R. E. Rosenthal, and L. M. Williams. Optimizing military airlift. *Operations Research*, 50(4):582–602, 2002.

[2] R. Bent and P. V. Hentenryck. A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. In *Computers and Operations Research*, 2006.

[3] M. Christiansen. Decomposition of a combined inventory routing and time constrained ship routing problem. *Transportation Science*, 33:3–16, 1999.

[4] J. Créput, A. Koukam, J. Kozlak, and J. Lukasik. An Evolutionary Approach to Pickup and Delivery Problem with Time Windows . In *Proceedings of International Conference on Computational Science*, 2004.

[5] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proc. of Int'l Conference on Parallel Problem Solving from Nature*, 2001.

[6] G. Ding, L. Li, and Y. Ju. Multi-strategy grouping genetic algorithm for the pickup and delivery problem with time windows. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computatio*, 2009.

[7] J. Durillo, A. Nebro, and E. Alba. The jMetal framework for multi-objective optimization: Design and architecture. In *Proc. IEEE Congress on Evolutionary Computation*, 2010.

[8] H. C. Lau and Z. Liang. Pickup and delivery with time windows: Algorithms and test case generation. In *Proc. of IEEE Int'l Conference on Tools with Artificial Intelligence*, 2001.

[9] H. Li and A. Lim. A metaheuristic for the pickup and delivery problem with time windows. In *Proc. of IEEE Int'l Conference on Tools with Artificial Intelligence*.

[10] O. B. G. Madsen, H. F. Ravn, and J. M. Rygaard. A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities and multiple objectives. *Annals of Operations Research*, 60:193–208, 1995.

[11] W. P. Nanry and J. Barnes. Solving the Pickup and Delivery Problem with Time Windows Using Reactive Tabu Search. In *Transportation Research Part B: Methodological*, 2000.

[12] G. Pankratz. A grouping genetic algorithm for the pickup and delivery problem with time windows. In *OR Spectrum*, 2005.

[13] D. H. Phan and J. Suzuki. Evolutionary multiobjective optimization for the pickup and delivery problem with time windows and demands (PDP-TW-D). In *In Proc. of the 15th Asia Pacific Symposium of Intelligent and Evolutionary Systems*, 2011.

[14] D. H. Phan and J. Suzuki. R2-IBEA: R2 indicator based evolutionary algorithm for multiobjective optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 1836–1845, June 2013.

[15] S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[16] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. In *Operation Research 35*, 1987.

[17] R. T. Takoudjou, J. C. Deschamps, and R. Dupas. A hybrid multi-start heuristic for the pickup and delivery problem with and without transshipment. In *Proc. of International Conference of Modeling, Optimization and Simulation*, 2012.

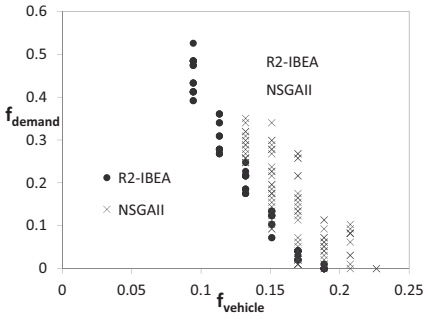
[18] P. Toth and D. Vigo. Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, 31:60–71, 1997.

[19] H. Wang, D-H-Lee, and R. Cheu. Pdptw based taxi dispatch modeling for booking service. In *Proc. of Int'l Conference on Natural Computation*, 2009.

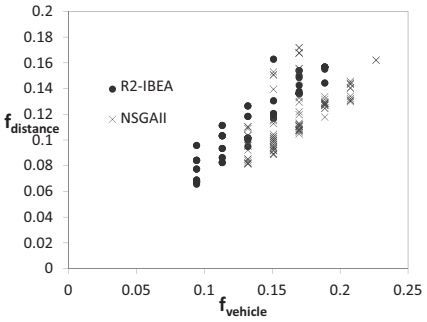
[20] H. F. Wang and Y. Y. Chen. A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering*, 62(1):84–95, 2012.

[21] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms: A comparative study. In *Proc. Int'l Conf. on Parallel Problem Solving from Nature*, pages 292–301, 1998.

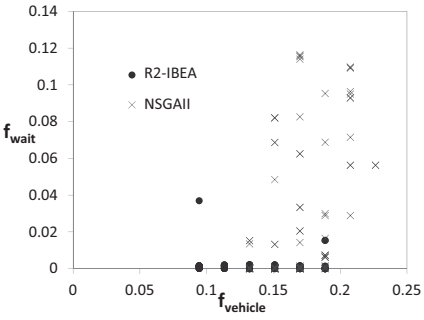
[22] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Trans.Evol. Comput.*, 3(4), 1999.



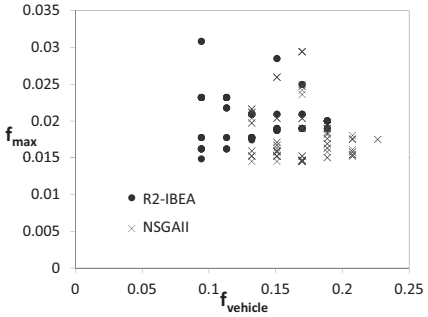
(a) Total Demands and Number of Vehicle



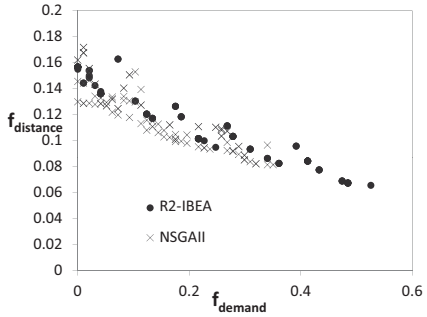
(b) Total Travel Distance and Number of Vehicles



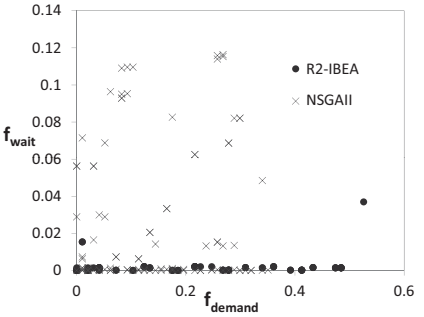
(c) Total Waiting Time An Number of Vehicles



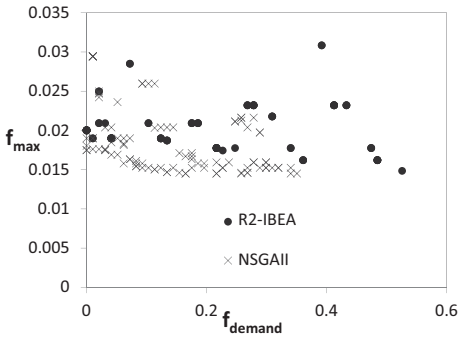
(d) Longest Route and Number of Vehicles



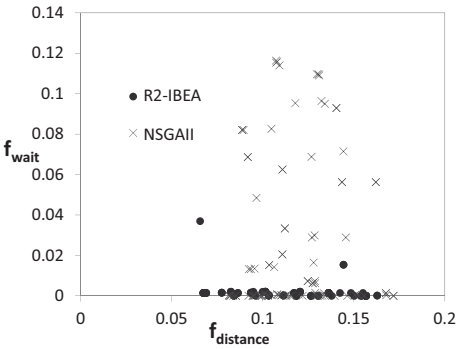
(e) Total Travel Distance and Total Demands



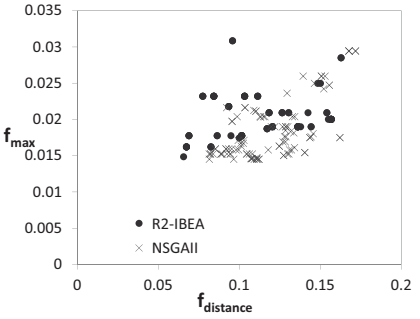
(f) Total Waiting Time and Total Demands



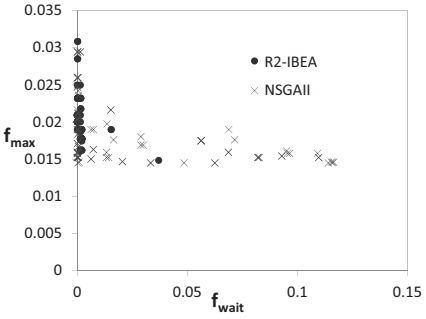
(g) Longest Route and Total Demands



(h) Total Waiting Time and Total Travel Distance



(i) Longest Route and Total Travel Distance



(j) Longest Route and Total Waiting Time

Figure 5: Two-dimensional Objective Spaces