Computers & Industrial Engineering 62 (2012) 84–95

Contents lists available at SciVerse ScienceDirect

Computers & Industrial Engineering

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

A genetic algorithm for the simultaneous delivery and pickup problems with time window ☆

Hsiao-Fan Wang *, Ying-Yen Chen

Department of Industrial Engineering and Engineering Management, National Tsing Hua University, No. 101, Section 2 Kuang Fu Road, Hsinchu 30013, Taiwan

ARTICLE INFO

Article history:
Received 3 September 2009
Received in revised form 11 April 2011
Accepted 22 August 2011
Available online 30 August 2011

Keywords:
Bi-directional logistics
Simultaneous delivery and pickup
Vehicle routing problem with time windows
Co-evolution genetic algorithm

ABSTRACT

This paper concerns a Simultaneous Delivery and Pickup Problem with Time Windows (SDPPTW). A mixed binary integer programming model was developed for the problem and was validated. Due to its NP nature, a co-evolution genetic algorithm with variants of the cheapest insertion method was proposed to speed up the solution procedure. Since there were no existing benchmarks, this study generated some test problems which revised from the well-known Solomon's benchmark for Vehicle Routing Problem with Time Windows (VRPTW). From the comparison with the results of Cplex software and the basic genetic algorithm, the proposed algorithm showed that it can provide better solutions within a comparatively shorter period of time.

1. Introduction

With global resources rapidly decreasing, introduction of reverse logistics may make use of returned merchandise, and thereby increase enterprise profits. Recently, many enterprises have incorporated reverse logistics into the conventional forward supply chain to form a closed-loop supply chain. Within such a loop, the logistics between the distribution/collection center and the customers is the most complicated part because it is related to the bi-directional logistics regarding delivery and pickup activities. In the literature, such problems have been referred to as Delivery and Pickup Problems (DPP).

Berbeglia, Cordeau, Gribkovskaia, and Laporte (2007) provided a survey on DPP. Referring to two different service strategies, DPP is divided into two categories: Vehicle Routing Problem with Backhauls (VRPB) and Simultaneous Delivery and Pickup Problem (SDPP). VRPB was first addressed by Deif and Bodin (1984). This problem considers that each driver picks up goods only after the last delivery of the vehicle is made. Due to the additional cost placed on separate trips for delivery and pick-up, efforts were further made on performing simultaneous delivery and pickup. Consequently, the customers were served once only. This latter planning situation can be called a Vehicle Routing Problem with Simultaneous Delivery and Pickup (VRPSDP) or a Simultaneous Pickup and Delivery Problem (SDPP). This paper refers it to a SDPP.

Such an application is frequently encountered, for example, in the distribution system of grocery store chains. Each grocery store may have a demand for both delivery (cf. fresh food or soft drinks) and pickup (cf. outdated items or empty bottles) and is serviced with a single stop by the supplier. The foundry industry is another example (Dethloff, 2001). Collection of the used sand and delivery of the purified reusable sand at the same customer location are carried out with only a single stop.

For more realistic applications, this paper further investigates a more general situation, called the Simultaneous Delivery and Pickup Problem with Time Windows (SDPPTW). A co-evolution genetic algorithm is proposed to resolve the SDPPTW. In order to evaluate the performance of the proposed method, some test problems are generated by revising the benchmark problems of the Vehicle Routing Problem with Time Windows (VRPTW) from Solomon (1987).

This paper is organized as follows. Section 2 discusses related work. Section 3 first formally defines the SDPPTW, and develops a mathematical model. Then, a Co-evolution Genetic Algorithm is developed based on the idea of the cheapest insertion method. Section 4 gives a detailed description of how the proposed co-evolution genetic algorithm is implemented on the SDPPTW. Section 5 provides comparisons among the Cplex software, the proposed algorithm, and the basic genetic algorithm; it then demonstrates the superiority of the proposed algorithm. Finally, conclusions are drawn in Section 6.

2. Literature review

Vehicle Routing Problems (VRP) originally focused on how to dispatch a group of vehicles to serve a group of customers with a

☆ This manuscript was processed by Area Editor Fernando Ordonez.

* Corresponding author. Tel.: +886 3 5742654; fax: +886 3 5722204.
E-mail address: hfwang@ie.nthu.edu.tw (H.-F. Wang).

0360-8352/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2011.08.018

given demand regarding minimum operation cost. Although demand could be uncertain, and there are articles addressing such a problem in the literature, this study focused on deterministic situations. This section will first review a variety of deterministic VRPs by their classes of problems, as shown in Fig. 1. In the Capacitated Vehicle Routing Problem (CVRP), one has to deliver goods to a set of customers with known demands, on minimum-cost vehicle routes originating and terminating at a depot. The vehicles are constrained by limited capacities (e.g. Prins (2004)). Theoretically speaking, if vehicles have unlimited capacity, the CVRP is relaxed into the VRP. Within the framework of the CVRP, two major categories regarding routing activities are extended: single activity CVRP which considers only pickup or delivery, and multiple activities which consider both pickup and delivery.

Single activity CVRP can be further extended to two classes of problems which have been studied extensively. The first is the Multi-Depot Capacitated Vehicle Routing Problem (MDVRP) which allows multiple depots (e.g. Cordeau, Gendreau, and Laporte (1997)). The second is the Vehicle Routing Problem with Time Windows (VRPTW) which correlates time windows with the customers (e.g. Cordeau, Desaulniers, Desrosiers, Solomon, and Soumis (2001)). VRPTW has been further extended to the Vehicle Routing Problem with Multiple Time Windows (VRPMTW), allowing each customer to have multiple time windows of services (e.g. Wang and Chiu (2009, chap. XIV)). There are some other variants like split delivery, soft time windows, uncertain models, etc. Because they are not close related to our study, we would not list them here.

In regard to Multi-activity CVRP, it also includes two subclasses: Pickup and Delivery Problems (PDP); and Delivery and Pickup Problems (DPP). PDP considers several requests from a customer for a vehicle. Each request consists of picking up an amount of goods at one location and delivering it to another location (e.g. Hoff, Gribkovskaia, Laporte, and Løkketangen (2009)), which has been extended to the Pickup and Delivery Problem with Time Windows (PDPTW) to associate time windows with the customers (e.g. Ropke and Pisinger (2006a)).

While PDP is often referred to as a mail express system, DPP can be regarded as a bi-logistic problem. In general DPP, two types of customers are served from a single depot by a fleet of vehicles. The first type of customers is known as "linehaul" customers, who require delivery of goods to their specific location, and the second type is known as "backhaul" customers, who require pickups from their specific locations. Berbeglia et al. (2007) provided a survey on both PDP and DPP. Referring to two different service strategies, DPP is divided into two categories consisting of Vehicle Routing Problem with Backhauls (VRPB) and Simultaneous Delivery and Pickup Problem (SDPP). Corresponding to some practical applications, the goods were picked up after the last delivery was made and thereby pickup customers were all served after delivery customers on the same route; in this case, the Vehicle Routing Problems with Backhauls (VRPB) was modeled (e.g. Ropke and Pisinger (2006b)).

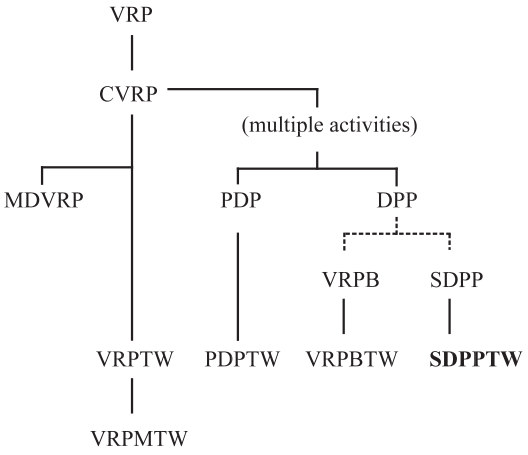


Fig. 1. Classification of VRPs.

Nowadays, a new distribution/redistribution planning situation arises in the closed-loop supply chain network. In particular, with environmentally motivated distribution/redistribution systems, customers have both a pick-up and a delivery demand. To reduce the service effort and interference of the customers, performing simultaneous delivery and pick-up services with a single stop for each customer is favored from both the service suppliers' and customers' viewpoints. The problem evolved from this strategy, called the Simultaneous Delivery and Pickup Problem (SDPP), recently has been studied extensively (see Ai and Kachitvichyanukul (2009), Berbeglia et al. (2007), Bianchessi and Righini (2007), Chen and Wu (2006), Dell'Amico et al. (2006), Dethloff (2001), Min (1989), Montané and Galvao (2006), and Nagy and Salh (2005)).

The Simultaneous Delivery and Pickup Problem with Time Windows (SDPPTW) extends the SDPP by associating time windows with the customers. Although Ai and Kachitvichyanukul (2009) discussed SDPPTW and proposed a model to formulate SDPPTW, they did not develop a solution procedure for SDPPTW. Also, their model can be simplified by reducing some redundant variables and redundant constraints. Therefore, their model should be revised into a simpler form for SDPPTW so that both the number of vehicles and the traveling cost are minimized. Consequently a solution procedure should be developed accordingly.

It is known that the VRPTW is NP-hard (Solomon, 1987) and VRPTW is polynomial time reducible from SDPPTW by setting all pickup demands equal to zero. Hence the SDPPTW is also NP-hard. Solving large scale SDPPTW to optimality is impossible within reasonable computational time; therefore, an efficient and effective solution procedure is needed. Since no algorithms are specified for solving SDPPTW, this paper refers to those heuristic and meta-heuristic solution methods designed to produce high quality solutions for VRPTW in a limited time. Most exact methods and some meta-heuristics for the VRPTW minimize total travel distance instead of minimizing the number of vehicles used. But in some literature, the number of vehicles (NV) was chosen as the first objective and the total travel distance (TD) only as the second factor (Alvarenga, Mateusb, and De Tomi (2007), Bräysy, Dullaert, and Gendreau (2004), Bräysy & Gendreau (2005a, 2005b), Bräysy, Hasle, and Dullaert (2004), Cordeau, Gendreau, Laporte, Potvin, and Semet (2002), Cordeau, Gendreau, Hertz, Laporte, and Sormany (2005), Gendreau, Laporte, and Potvin (2001), Mester, Bräysy, and Dullaert (2007), and Pisinger and Ropke (2007)).

The cheapest insertion method, used in Mester et al. (2007) and Osman (1993) will be adopted to develop a co-evolution genetic algorithm for SDPPTW in the following section due to its conceptual simplicity and computational efficiency.

3. Simultaneous delivery and pickup problem with time window

3.1. Problem description

Given a number of customers who require both forward supply service and reverse recycling service within a certain time period, the problem concerned in this paper is how to send out a fleet of capacitated vehicles, stationed at a distribution center (DC), to meet the requests with the minimum number of vehicles and traveling cost. Since the greater the number of vehicles, the less the travel cost, trade-offs between these two kinds of costs needed to be considered.

Based on the simultaneous delivery and pick-up activities of a vehicle, all vehicles should return to the collection center (CC) to unload the recycled material. If the distribution center is the same as the collection center, then CC = DC.

3.2. Mathematical formulation

Based on the above problem description, the flow operation is analyzed in this section. Basic notations are introduced, first with the number of customers denoted by n, DC by 0, and CC by $n+1$. If DC = CC, then $0 \equiv n+1$.

3.2.1. Notations

Sets

J	Set of all customers, $J = j\|j = 1, \ldots, n)$
J_F	Set of all forward channel nodes, i.e. DC and customer locations, $J_F = \{0\} \cup J$
J_R	Set of all reverse channel nodes, i.e. customer locations and CC, $J_R = J \cup \{n+1\}$
J_C	Set of all nodes, $J_C = \{0\} \cup J \cup \{n+1\}$
V	Set of all vehicles, $V = \{v\|v = v_1, \ldots, v_{\|V\|}\}$

Coefficients

q_v	Capacity of vehicle v, $q_v \in \boldsymbol{R}^+$
g_v	Dispatching cost of vehicle v, $g_v \in \boldsymbol{R}^+$
c_{ij}	Distance between nodes $i \in JF$, $j \in JR$; $i \neq j$, $c_{ij} \in \boldsymbol{R}^+$
t_{ij}	Travel time between nodes $i \in JF$, $j \in JR$; $i \neq j$, $t_{ij} \in \boldsymbol{R}^+$
d_j	Delivery demand of customer $j \in J$, $d_j \in \boldsymbol{Z}^+$
p_j	Pickup amount of customer $j \in J$, $p_j \in \boldsymbol{Z}^+$
s_j	Service time of customer $j \in J$, $s_j \in \boldsymbol{R}^+$
a_j	Earliest service time of customer $j \in J$, $a_j \in \boldsymbol{R}^+$
b_j	Latest service time of customer $j \in J$, $b_j \in \boldsymbol{R}^+$
a_0	Earliest departure time of any vehicle from DC, $a_0 \in \boldsymbol{R}^+$
b_{n+1}	Latest arrival time that a vehicle must return CC, $b_{n+1} \in \boldsymbol{R}^+$
M	An arbitrary large constant
α	A parameter indicating the trade-off between dispatching cost and travel cost, $\alpha \in [0, 1]$

Decision variables

L_{0v}	Load of vehicle $v \in$ V when leaving DC, $L_{0v} \in \boldsymbol{Z}^+$
L_j	Remaining load of a vehicle after having served customer $j \in J$, $L_j \in \boldsymbol{Z}^+$
x_{ijv}	Traveling variable of a vehicle $v \in V$, $x_{ijv} \in \{0,1\}$; if it travels directly from node $i \in J_F$ to node $j \in J_R$, $x_{ijv} = 1$; otherwise $x_{ijv} = 0$
T_j	Time to begin servicing customer $j \in J$, $T_j \in \boldsymbol{R}^+$
T_{0v}	Departure time of vehicle $v \in V$ at DC, $T_{0v} \in \boldsymbol{R}^+$
$T_{(n+1)v}$	Arrival time of vehicle $v \in V$ at CC, $T_{v(n+1)} \in \boldsymbol{R}^+$

3.2.2. Model SDPPTW

$$\text{Minimize } z = \alpha \sum_{v\in V} \sum_{j\in J} g_v x_{0jv} + (1-\alpha) \sum_{i\in J_F} \sum_{j\in J_R} \sum_{v\in V} c_{ij} x_{ijv} \tag{1}$$

(Minimize total dispatching cost and total traveling cost) subject to

$$\sum_{i\in J_F} \sum_{v\in V} x_{ijv} = 1 \ \forall j \in J \tag{2}$$

(Service all customer nodes exactly once)

$$\sum_{i\in J_F} x_{ihv} = \sum_{j\in J_R} x_{hjv} \ \forall h \in J, \ \forall v \in V \tag{3}$$

(Arrive at and leave each customer with the same vehicle)

$$\sum_{j\in J} x_{0jv} = \sum_{i\in J} x_{i(n+1)v} \ \forall v \in V \tag{4}$$

(Vehicles which depart from D.C. should finally return C.C.)

$$L_{0v} = \sum_{i\in J_F} \sum_{j\in J} d_j x_{ijv} \ \forall v \in V \tag{5}$$

(Initial vehicle loads)

$$L_j \geqslant L_{0v} - d_j + p_j - M(1 - x_{0jv}) \ \forall j \in J, \ \forall v \in V \tag{6}$$

(Vehicle loads after first customer)

$$L_j \geqslant L_i - d_j + p_j - M\left(1 - \sum_{v\in V} x_{ijv}\right) \ \forall i \in J, \ \forall j \in J \tag{7}$$

(Vehicle loads 'en route')

$$L_{0v} \leqslant q_v \ \forall v \in V \tag{8}$$

$$L_j \leqslant q_v + M\left(1 - \sum_{\forall i\in J_F} x_{ijv}\right) \ \forall j \in J, \ \forall v \in V \tag{9}$$

(Vehicle capacity constraints)

$$T_{0v} + t_{0j} - M(1 - x_{0jv}) \leqslant T_j \ \forall j \in J, \ \forall v \in V \tag{10}$$

$$T_i + s_i + t_{ij} - M\left(1 - \sum_{v\in V} x_{ijv}\right) \leqslant T_j \ \forall i \in J, \ \forall j \in J \tag{11}$$

$$T_i + s_i + t_{i(n+1)} - M(1 - x_{i(n+1)v}) \leqslant T_{(n+1)v} \ \forall i \in J, \ \forall v \in V \tag{12}$$

$$a_0 \leqslant T_{0v} \ \forall v \in V \tag{13}$$

$$a_j \leqslant T_j \leqslant b_j \ \forall j \in J \tag{14}$$

$$T_{(n+1)v} \leqslant b_{n+1} \ \forall v \in V \tag{15}$$

(Ensure feasibility of the time schedule)

$$x_{ijv} \in \{0,1\} \ \forall i \in J_F, \ \forall j \in J_R, \ \forall v \in V \tag{16}$$

This model contains $(n+1)^2|V| + 2n + 3|V|$ variables and $2n^2 + 5n|V| + 2n + 6|V|$ constraints. Although Ai and Kachitvichyanukul (2009) have proposed a similar model, it contains $2(n+1)^2|V| + (n+2)|V|$ variables and $2(n+1)^2|V| + 2n|V| + n + 3|V|$ constraints. The additional variables and constraints are due to the fact that variables T_{jv} and L_{jv} in Ai and Kachitvichyanukul's (2009) model were used to represent the service time point and the vehicle load of vehicle v at customer j. However, these variables can be simplified by T_j and L_j because x_{ijv} have been restricted to exactly one vehicle v to serve customer j.

4. A genetic algorithm for the large-scale SDPPTW

"Time" is always a managerial issue for an industry. Determining how best to support a quick and accurate decision is always desired by a manager. Since exact algorithms are both theoretically and practically impossible for the large-scaled SDPPTW, it is necessary to introduce heuristic or meta-heuristic solutions. This section will propose a co-evolution genetic algorithm to show its efficiency and effectiveness with the continuous improvement mechanism.

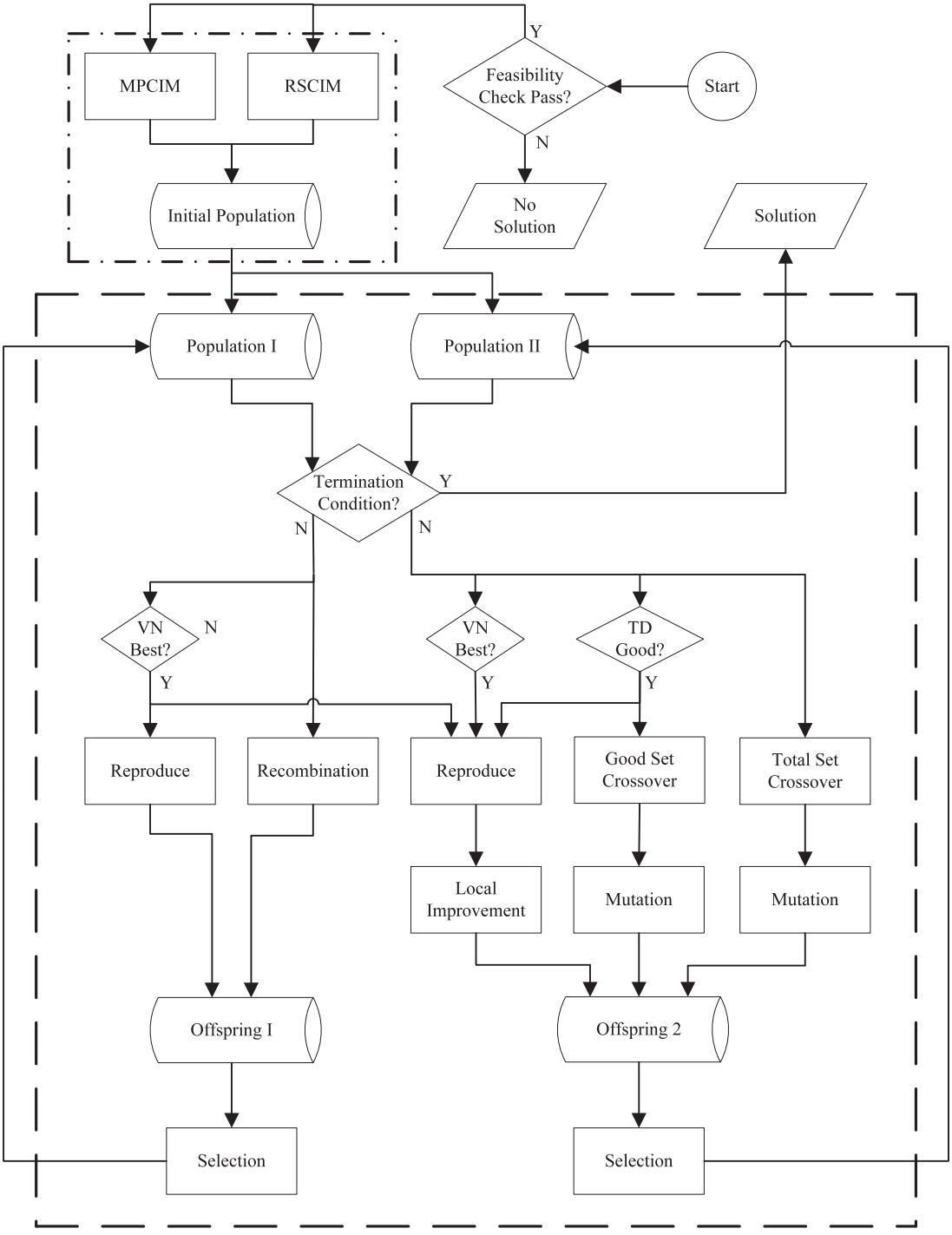
4.1. Framework of the proposed co-evolution genetic algorithm

The Genetic Algorithm (GA) was first proposed by Holland (1975). Due to its global search mechanism, GA has shown its capability to find good solutions for complex mathematical problems,

like the VRP and other NP-hard problems, in a reasonable amount of time. The traditional design of a GA faced the dilemma of converging too quickly with non-acceptable local optima or converging too slowly and resulting in exhaustive time consumption for deriving an acceptable solution. This study proposes a co-evolution genetic algorithm to avoid any one of the above situations. Two simple heuristics (MPCIM and RSCIM) used for generating initial population will be introduced later. Based on the framework shown in Fig. 2, the GA employs two populations for two respective purposes: diversification and intensification.

4.2. Initial population

If a fast and simple heuristic procedure to distribute all customers to the vehicles is used to obtain the first generation, it can significantly reduce the GA's computational time required to reach the reasonable local minima. In other words, the methods to create initial solutions for a GA should compute quickly as well as raise as many properties as possible, that the optimal solution possesses. With this purpose, the concept of cheapest insertion heuristic, based on the savings procedure of Clarke and Wright (1964), was



MPCIM, RSCIM: two methods to generate initial solutions; they will be introduced later.

Fig. 2. The framework of the proposed co-evolution genetic algorithm.

frequently used by many researchers. For the descriptions of some variants of the cheapest insertion heuristic, one can refer to Mester et al. (2007) and Osman (1993).

This section will firstly describe the cheapest insertion method (CIM). Based on the concept of CIM, two revised CIM with a crossover mechanism are proposed to generate the required initial solutions. The details of these methods and mechanism are introduced in the next subsections.

4.2.1. Cheapest insertion method

The cheapest insertion method (CIM) begins with an initial solution in which each customer is served individually by a vehicle, i.e. the number of vehicles = the number of routes = the number of customers. Reinsertions of single customers to alternative positions in the solution vector are then attempted in a loop as illustrated in Fig. 3.

For a single route customer k, the method considers alternative positions between adjacent customers l and m in other routes. The reinsertions are evaluated using the cost saving criterion of Osman (1993), $(c_{0k} + c_{k(n+1)} + c_{lm}) - (c_{lk} + c_{km})$, where c_{ij} refers to the costs of the associated arcs (i,j). Each reinsertion examines all single route customers; the reinsertion trial with the maximum cost savings is executed. A simple example for cost saving criterion is illustrated in Fig. 4. In this example, node 3 is going to be inserted into another route and there are three choices. Fig. 4 illustrates that the cheapest insertion is that node 3 is inserted after node 1. The reinsertion is stopped if the algorithm cannot reduce the number of single customer routes. The solution obtained from this method contains several minimum cost arcs which the optimal solution contains.

4.2.2. Multi-parameter cheapest insertion method

Conventional CIM generates one solution only. In order to speed up the global search process, a multi-parameter cheapest insertion method (MPCIM) was proposed. MPCIM is the same as CIM except for the insertion criterion. In MPCIM, the reinsertions are evaluated using a modified insertion criterion of Mester et al. (2007),

$$(c_{0k} + c_{k(n+1)} + c_{lm}) - \beta(c_{lk} + c_{km})$$

where β refers to a parameter as a cost saving threshold. In the case of 100 customers, Mester et al. (2007) suggested that for β all values in range 0.2–1.4 are tried in increments of 0.2 units, which are resultant in seven initial solutions.

4.2.3. Random seeds cheapest insertion method

In addition to finding high quality initial solutions, the initial population of GA also needs to search widely in order to avoid falling into local optimum. For this reason, random seeds cheapest insertion method (RSCIM) was proposed. RSCIM is also based on the concept of CIM. Instead of beginning with an initial solution in which each customer is supplied individually by a separate route, RSCIM generates a random order of customers for route growing. Top k customers of the order are the seeds for route growing where k = (Total Demand/Average Vehicle Capacity). Addition of a single route customer from the order to the partial solution vector and reinsertions of single route customers to alternative positions in the partial solution vector are then attempted in a loop. All customers will be added to the partial solution vector, one by one, by a random order; and the reinsertion trail with the maximum cost savings is executed. The reinsertion is stopped if the algorithm cannot reduce the number of single customer routes. This method can search a wider space than MPCIM can for the initial population.

4.2.4. Crossover

In the proposed GA, the search space is limited to the feasible region; therefore, every individual is feasible. Consequently, caution should be taken on the crossover and mutation operators, because a simple exchange between two customers can violate time and capacity constraints. Referring to Alvarenga et al. (2007), this study proposed a revised crossover algorithm which does not entail bias in any particular direction and lets offspring inherit good properties from parents. This crossover operator has the offspring inherit as many routs as possible from parents. Once inherited routes are chosen, they can be regarded as seed routes and all other un-routed customer can be inserted into seed routes or other single customer routes. The criterion of the insertion is the same as the one used in RSCIM. The procedure to construct a solution after inherited routes are chosen is called Fixed Seeds Cheapest Insertion Method (FSCIM). This crossover operator generates one offspring each time.

Algorithm 1. Crossover algorithm

```
function Crossover;
begin
  repeat
    Copy Random Route from Parent 1 to the offspring;
    Copy Random Route from Parent 2 to the offspring;
  until (no more inherited routes are feasible)
  All un-routed customers form single customer routes;
  Reduce all single customer routes by FSCIM;
end;
```

4.2.5. Initial population construction

As illustrated in Fig. 2, the initial population is constructed by MPCIM and RSCIM with the crossover mechanism described above. Each of MPCIM and RSCIM creates seven initial solutions, respectively. If the population size N is used, the remaining $N - 14$ initial

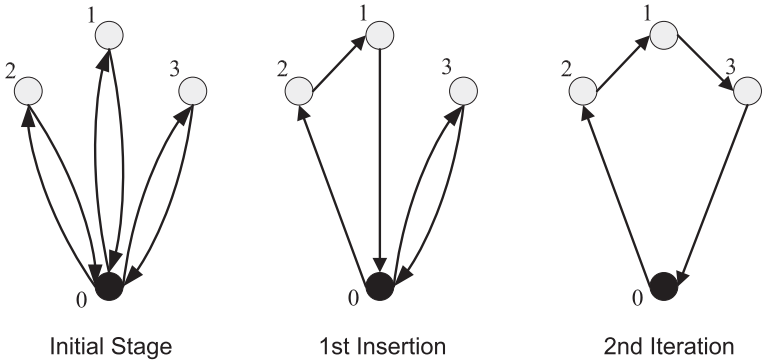


Fig. 3. Illustration for cheapest insertion method.

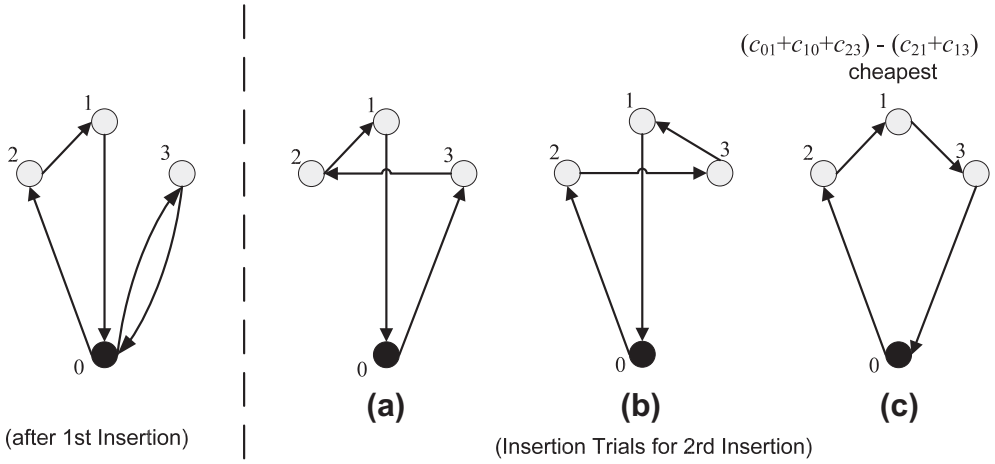


Fig. 4. Illustration for cost savings criterion.

solutions are created by crossover from these two groups of solutions for deep and wide searches. The crossover operation is introduced in the next part. The proposed GA has two populations and these N initial solutions are copied directly into Population I and Population II for different purposes of evolution.

Population I is used for diversification purposes, while Population II is employed for evolutionary intensification. The co-evolution structure of these two populations is illustrated in Fig. 5. Evolution of each population is described in the following section.

4.3. Co-evolution

Population I aims to retain the wide searching ability of the proposed GA through three operators: Reproducing, Recombination and Selection, while Population II aims to reach high quality solutions quickly and improve them constantly through four operators: Reproducing, Local Improvement, Crossover, Mutation, and Selection. In both populations, N parents generate $2N$ offspring and then $2N$ offspring compete with each other for only N to survive as the parents of the next generation.

In Population I, the Reproducing and Recombination operators are used to generate $2N$ offspring from N parents. The Selection operator is used to generate N parents of the next generation from $2N$ offspring. In Population II, There are 2 types of parents and 2 types of offspring, each denoted by high class and massive class, respectively. High class offspring are generated from high class parents only, and massive offspring are generated from all parents. High class parents include the best individuals of both Population I and Population II, plus some good individuals of Population II. The high-class parents and high-class offspring are discussed below. The definition of good individuals is introduced in the subsection for the Selection operator.

The proportion of high-class offspring to massive offspring is suggested to be 3:7 by experimentation taken on different ratios of: 1:9, 2:8, 3:7, 4:6, and 5:5. In order to generate $2N \times 30\%$ high-class offspring, about $1+1+\sqrt{N}$ high-class parents are needed, in which one is the best individual of Population I; one is the best individual of Population II and $\sqrt{N}$ are good individuals of Population II. In spite of the unbalanced large proportion, high-class offspring still need to compete with massive offspring. Class

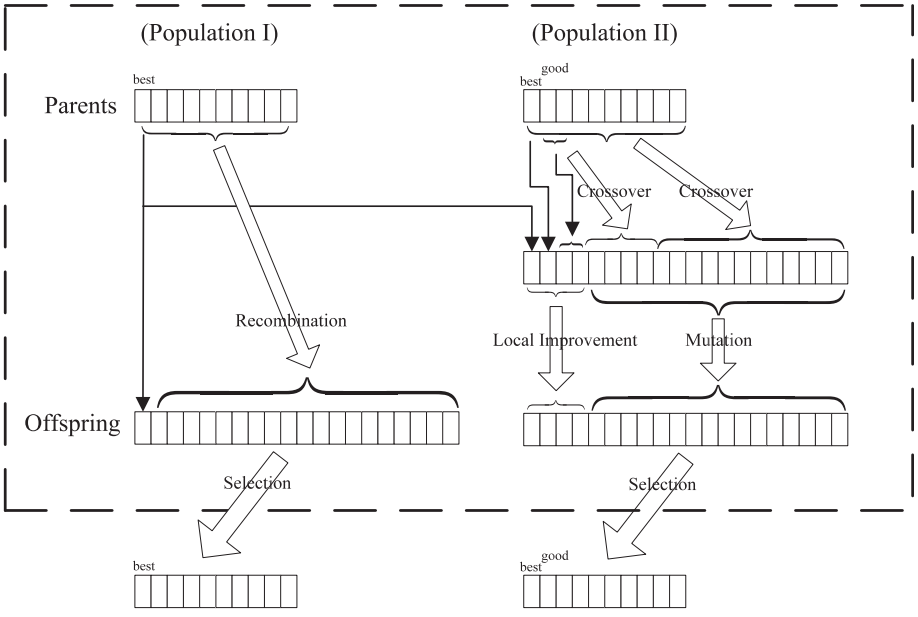


Fig. 5. Co-evolution structure.

redistribution is done in the $2N$ offspring by using a selection operator on Population II to generate N parents of the next generation.

4.3.1. Reproducing

In population I, the reproducing operator copies the best individual of each set of parents to the first offspring. The best individual is the one with minimal objective value. In some literature, minimizing the number of vehicles (NV) is considered the primary objective. Then, with the minimal number of vehicles, minimizing the total travel distance (TD) is used as the secondary objective. This criterion can be achieved by setting a large value of α with $\alpha \to 1$. Reproducing to keep the best individual is also known as the Elitism strategy, which guarantees that GA never retreats in the high quality solution.

In Population II, the Reproducing operator copies the best individuals from both Population I and Population II, and $\sqrt{N}$ good individuals from Population II to form $1+1+\sqrt{N}$ offspring prototypes. A Local Improvement operator is used to improve these $1+1+\sqrt{N}$ offspring prototypes to form $1+1+\sqrt{N}$ offspring and is introduced in the next part. $\sqrt{N}$ good parents are those individuals with the top $\sqrt{N}$ minimal travel distances and are defined in the subsection for the Selection operator.

4.3.2. Recombination

The recombination operator is a *remove-insert mechanism* which preserves the wide searching ability of the proposed GA. In the first step, the algorithm randomly removes $1/2 \sim 1/10$ of its customers from their routes. Then, the reinsertion of isolated customers is done by FSCIM, where the existing routes are regarded as seed routes.

Algorithm 2. Recombination algorithm

```
function Recombination;
begin
  Randomly remove 1/2 ~ 1/10 of customers from their route;
  Reinsert isolated customers by FSCIM;
end;
```

4.3.3. Local improvement

Two types of Local Improvement are used in this work: Reinsertion Improvement and Swap Improvement. Either one of these two kinds of improvements can be used to improve the offspring prototypes which the Reproducing operator generates.

Reinsertion Improvement: This operator reinserts one single customer into alternative positions in the solution vector by Osman (1993) cost savings criterion. For a customer k currently serviced between customers i and j, the operator considers all alternative positions in the solution vector. Consider the position between the adjacent customers l and m, the cost savings is evaluated as:

$$(c_{ik}+c_{kj}+c_{lm})-(c_{lk}+c_{km}+c_{ij}).$$

The improvement operator is applied here with the best-move strategy, i.e. all possible moves in the current neighborhood are evaluated and the best improvement move is selected.

Swap Improvement: This operator swaps the position of two customers simultaneously in the solution vector by Osman (1993) cost savings criterion. For customers k and h currently serviced between customers i and j, and l and m, respectively, the swap possibility is evaluated on cost savings as:

$$(c_{ik}+c_{kj}+c_{lh}+c_{hm})-(c_{lk}+c_{km}+c_{ih}+c_{hj}).$$

Based on the best-move strategy, the local improvement operator is applied to all of $1+1+\sqrt{N}$ reproduced individuals to form $1+1+\sqrt{N}$ high-class offspring.

4.3.4. Crossover

In Population II, the crossover algorithm is the same as the one developed for the initial population in Section 4.2. Part of the high-class offspring and massive offspring are generated by Crossover and Mutation operators from good parents and all parents, respectively. Twenty-one high-class offspring are generated by the crossover of all combinations of $\sqrt{N}$ good parents. Seventy massive offspring are generated by the crossover of two arbitrary individuals of all parents. The Mutation operator is used to mutate these $N-1-1-\sqrt{N}$ offspring to form $N-1-1-\sqrt{N}$ of the next generation, which is introduced in the next paragraph.

4.3.5. Mutation

A total of eleven different operators are used in the mutation phase of the proposed GA in order to input new characteristics to the current population. This in turn will enlarge the feasible search space. In particular, some specific operators are developed in this study to speed up the evolution of the individuals, for instance, mutation 11 for SDPPTW. Fig. 6 is an illustration for mutation 01. Complete illustrations for all mutation operators can be seen in Appendix A.

Mutation_01 (Random customer reinsertion): This operator randomly chooses a customer, removes it, and re-inserts it randomly in a feasible alternative position in the same route if possible (See Fig. 6).

Mutation_02 (Random customer migration): This operator randomly chooses a customer and tries a feasible migration of this customer to another non-empty vehicle.

Mutation_03 (Bringing a random customer): This operator randomly chooses a vehicle and tries to bring a random customer from other vehicles in such a way that it does not violate feasibility.

Mutation_04 (Random customer exchange): This operator randomly chooses two vehicles and tries to exchange a couple of customers in these two routes in such a way that it does not violate feasibility.

Mutation_05 (Route partitioning): This operator randomly chooses a vehicle, two random customers and divides this route in two others, using those two customers as reference.

Mutation_06 (Reducing one route): This operator randomly chooses a vehicle, isolates all customers in the route, and then reinserts them into other routes by RSCIM, if possible.

Mutation_07 (Customer reinsertion with best savings): This operator randomly chooses a customer, removes it, and re-inserts it into the best position, i.e., the position in the same route with a minimal travel distance.

Mutation_08 (Customer migration with best savings): This operator randomly chooses a customer; then tries to find the best migration for this customer to another non-empty vehicle with maximal positive cost savings.

Mutation_09 (Bringing the best customer): This operator randomly chooses a vehicle and tries to bring a customer from other vehicles with maximal positive cost savings).

Mutation_10 (Best customer exchange): This operator randomly chooses two vehicles, then verifies all possibilities to exchange a couple of customers in these two routes, until a maximal reduction in total travel distance is obtained.

Mutation_11 (Early-pickup late-delivery customer exchange): This operator randomly chooses one early-pickup customer or late-delivery customer. If the chosen customer is an early-pickup one, they are randomly swapped with another later customer, if possible. If the chosen customer is a late-delivery one, they are randomly swapped with another earlier customer, if possible.

4.3.6. Selection

Instead of using the objective function as a fitness function, the fitness is related to total travel distance (total travel cost). This is

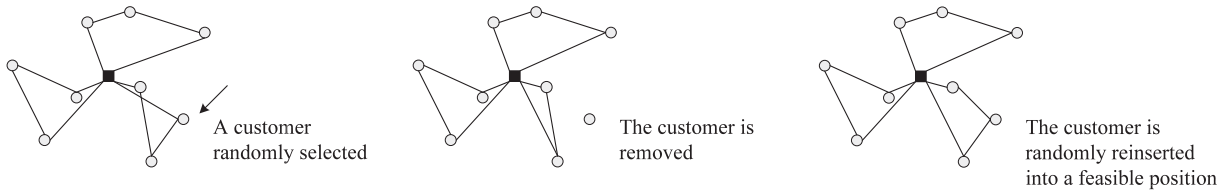


Fig. 6. Mutation_01: Random customer reinsertion.

motivated by the fact that better individuals are more easily evolved from the ones with the lower TD than the ones with the lower NV. Recall that the population size is N and the number of offspring is $2N$. The fitness values of the individuals with the minimal TD and the maximal TD are set to be $4N$ and $2N+1$, respectively. This implies that the individual with the minimal TD has about double the probability to be reproduced compared to the one with the maximal TD. The fitness is defined as below:

$$\text{fitness} = 4N + 1 - (\text{ranking of TD}).$$

All offspring are evaluated twice by the fitness function and objective function. In both populations, the individual with the best objective function is kept directly by the Elitism strategy. In Population II, $\sqrt{N}$ additional good offspring are kept to be the good parents of the next generation. Good offspring are defined as those $\sqrt{N}$ individuals with lower TD (higher fitness). The remaining parents of the next generation are reproduced by the Roulette wheel selection rule, related to the fitness values. In other words, these individuals are reproduced from all offspring where the individual k has the probability of reproduction as:

$$\Pr(\text{individual } k \text{ to be reproduced}) = \frac{\text{fitness}}{\text{total fitness}}$$

4.4. Termination condition

Referring to Algorithm 1, the termination condition of the proposed GA is the convergence of the procedure. At the beginning of each generation, updating the best individual of these two populations is based on the comparison between the best individuals of two consecutive generations. In other words, the evolution is under a 'stagnancy' state as the rate of improvement between two consecutive generations is zero or is relatively small (less than 10^{-3}). The proposed GA converges when the evolution has been under 'stagnancy' for 500 consecutive generations. Once the terminal condition is satisfied (the proposed GA converges), the evolution process will be terminated and the best individual is reported as the best solution.

5. Evaluation and discussion

Since there have not been any studies with testing problems which were dedicated to SDPPTW, for evaluation, this study generated some SDPPTW test problems which were revised from Solomon's VRPTW benchmarks (refer to Solomon (1987)). The set of Solomon's test problems is composed of six different problem types (C1, C2, R1, R2, RC1, & RC2). Each data set contains between eight to twelve 100-customer problems. The categories of the six problem types refer to:

C: with clustered customers whose time windows were generated based on a known solution;

R: with customer locations generated uniformly randomly over a square;

RC: with a combination of randomly placed and clustered customers.

where

Type 1 has narrow time windows and small vehicle capacity, and

Type 2 has large time windows and large vehicle capacity.

Revised from Solomon's test problems, this study generated three 10-customer problems, three 25-customer problems, three 50-customer problems, and fifty-six 100-customer problems. Due to different objective functions used in the literature, this analysis employed the trade-off parameter α to adjust for different decision criteria and compared the results with a hierarchical objective function where the primary objective is to minimize the number of vehicles and the secondary objective is to minimize the total distance or travel time. All experiments were executed on an Intel Core2 Quad 2.4G computer with 1G memory.

5.1. Definition of the parameters

Appropriate adjustment of parameters in GAs can make a significant difference in terms of performance. Some values can provide very high performance in specific instances while giving premature convergence in others, even over the same kind of problems. The parameters used in the proposed GA are listed in Table 1 and explained below.

SIZE_POP1 and SIZE_POP2 represent the population sizes of Population 1 and Population 2 respectively in which values of (50,50) are empirically chosen.

NUM_MPCIM and NUM_RSCIM represent the numbers of MPCIM and RSCIM solutions respectively which are used to construct the initial population as described in Section 4.

NUM_GOOD_TD represents the number of good TD solutions kept in Population 2 selection.

MUT_RATE = 0.5 is set for the mutation rate. Once mutation is executed, all mutation operators have equal probabilities to be

Table 1
Parameters used in the GA.

GA parameter (description)	Value
SIZE_POP1 (population size of population 1)	50
SIZE_POP2 (population size of population 2)	50
NUM_MPCIM (number of MPCIM solutions)	7
NUM_RSCIM (number of RSCIM solutions)	7
NUM_GOOD_TD (number of good TD solutions)	7
MUT_RATE (mutation rate)	0.5
CONV_COUNT (definition for convergence in terms of number of consecutive stagnancy generations)	500

Table 2
Experiments to SDPPTW RC1-type test problems under different mutation rates.

Mutation rate	Cumulative NV	Cumulative TD	Computational time
0	103	11225.73	4145
0.1	103	11124.04	3863
0.3	102	11200.94	5415
0.5	102	11138.47	4382
0.7	103	11185.82	3454
0.9	103	11133.94	3990

Table 3
Comparison between the solutions of Cplex and the proposed GA to the small-scale SDPPTW.

Problem	NV	Cplex		(Lower bound)		GA		
		TD	Com. time	NV	TD	NV	TD	Com. time
RCdp1001	3	348.98	1			3	348.98	1
RCdp1004	2	216.69	1503			2	216.69	1
RCdp1007	2	310.81	25			2	310.81	1
RCdp2501	5	551.05	16			5	551.05	3
RCdp2504	7*	738.32*	485,660*	0	423.88	4	473.46	2
RCdp2507	7*	634.20*	439,321*	0	1254.62	5	540.87	3
RCdp5001	9	994.18	327,404			9	994.18	18
RCdp5004	*14	1961.53*	839,320*	0	466.83	6	725.59	23
RCdp5007	13*	1814.33*	1,546,429*	0	388.51	7	809.72	22

* The "out of memory" values.

Table 4
Comparison between Cplex solutions and the proposed GA's solutions to the SDPPTW.

Problem	Basic GA		Proposed GA		Gap		Problem	Basic GA		Proposed GA		Gap	
	NV	TD	NV	TD	NV	TD (%)		NV	TD	NV	TD	NV	TD (%)
Rdp101	19	1656.68	19	1653.53	0	**−0.19**	Rdp201	4	1299.17	4	1280.44	0	**−1.44**
Rdp102	18	1474.93	17	1488.04	**−1**	0.89	Rdp202	4	1121.88	4	1100.92	0	**−1.87**
Rdp103	14	1217.6	14	1216.16	0	**−0.12**	Rdp203	3	1031.02	3	950.79	0	**−7.78**
Rdp104	10	1029.38	10	1015.41	0	**−1.36**	Rdp204	3	774.75	3	775.234	0	0.06
Rdp105	15	1391.29	15	1375.31	0	**−1.15**	Rdp205	4	998.80	3	1064.43	**−1**	6.57
Rdp106	13	1258.82	13	1255.48	0	**−0.27**	Rdp206	3	992.14	3	961.32	0	**−3.11**
Rdp107	11	1091.89	11	1087.95	0	**−0.36**	Rdp207	3	868.59	3	835.01	0	**−3.87**
Rdp108	10	988.91	10	967.49	0	**−2.17**	Rdp208	3	732.00	3	718.51	0	**−1.84**
Rdp109	12	1248.35	12	1160	0	**−7.08**	Rdp209	4	894.45	3	930.26	**−1**	4.00
Rdp110	12	1157.78	12	1116.99	0	**−3.52**	Rdp210	3	1037.20	3	983.75	0	**−5.15**
Rdp111	11	1084.27	11	1065.27	0	**−1.75**	Rdp211	3	827.17	3	839.61	0	1.50
Rdp112	10	998.08	10	974.03	0	**−2.41**							
Cdp101	11	1001.97	11	1001.97	0	0.00	Cdp201	3	591.56	3	591.56	0	0.00
Cdp102	10	1030.68	10	961.38	0	**−6.72**	Cdp202	3	591.56	3	591.56	0	0.00
Cdp103	10	905.71	10	897.65	0	**−0.89**	Cdp203	3	591.17	3	591.17	0	0.00
Cdp104	10	889.3	10	878.93	0	**−1.17**	Cdp204	3	590.6	3	590.6	0	0.00
Cdp105	11	983.1	11	983.1	0	0.00	Cdp205	3	588.88	3	588.88	0	0.00
Cdp106	11	878.29	11	878.29	0	0.00	Cdp206	3	588.49	3	588.49	0	0.00
Cdp107	11	913.81	11	913.81	0	0.00	Cdp207	3	588.29	3	588.29	0	0.00
Cdp108	11	922.59	10	951.24	−1	3.11	Cdp208	3	588.32	3	588.32	0	0.00
Cdp109	10	938.1	10	940.49	0	0.25							
RCdp101	15	1665.2	15	1652.9	0	**−0.74**	RCdp201	5	1376.98	4	1587.92	**−1**	15.32
RCdp102	14	1516.34	14	1497.05	0	**−1.27**	RCdp202	4	1210.75	4	1211.12	0	0.03
RCdp103	13	1370.7	12	1338.76	**−1**	**−2.33**	RCdp203	4	971.11	4	964.65	0	**−0.67**
RCdp104	11	1198.99	11	1188.49	0	**−0.88**	RCdp204	3	857.23	3	822.02	0	**−4.11**
RCdp105	15	1561.01	14	1581.26	**−1**	1.30	RCdp205	5	1302.51	4	1410.18	**−1**	8.27
RCdp106	14	1453.34	13	1422.87	**−1**	**−2.10**	RCdp206	4	1167.52	3	1176.85	**−1**	0.80
RCdp107	12	1301.58	12	1282.1	0	**−1.50**	RCdp207	4	1058.86	4	1036.59	0	**−2.10**
RCdp108	11	1195.39	11	1175.04	0	**−1.70**	RCdp208	3	908.08	3	878.57	0	**−3.25**

Boldface indicates that those solutions of the proposed GA have less vehicles or lower total distances.

applied. Since RC1-type problems are the most complicated, particular tests are carried out by taking the mutation rates of 0.1, 0.3, 05, 0.7, and 0.9 to solve eight RC1-type 100-customer SDPPTW test problems. Results are listed in Table 2 where the mutation rate 0.5 shows the best performance.

Convergence is defined as the evolution appearing in a given amount (CONV_COUNT) = 500 of the consecutive stagnancy generations and is used as a stopping condition.

5.2. Computational results for the small-scale SDPPTW

The commercial linear programming software, like ILOG Cplex, could find optimal solutions for the small-scale SDPPTW and hence can be used to evaluate the accuracy of the proposed model. Three 10-customer problems, three 25-customer problems, and three 50-customer problems were revised from Solomon's RC101, RC104 and RC107 problems and re-named as RCdp10101, RCdp10104, RCdp10107, RCdp25101, RCdp25104, RCdp25107, RCdp50101, RCdp50104, and RCdp50107 respectively. These nine small-scale problems were solved by Cplex and the proposed GA. The comparison between solutions of Cplex and the proposed GA to these small-size problems is listed in Table 3. One can see that Cplex is only able to solve five problems to optimality, but not the rest due to "out of memory." For those five problems, the proposed GA also could get their optimal solutions within much shorter time ($1 \sim 23$ s). For the other four problems, the "out of memory" values with Cplex are poor and the lower bounds with Cplex are not tight. GA's solutions are much better than Cplex's best solutions.

5.3. Computational results for the SDPPTW

Further evaluation was carried out by testing the performance of the proposed algorithm with respect to the basic genetic algorithm (GA) which has only one population with crossover, mutation and local-improvement operators. The results were shown in Table 4. The basic GA tended to fall into local optima, but the

proposed GA, with two populations, overcome this shortage and could find solutions with fewer vehicles. In other words, among fifty-six 100-customer SDPPTW test problems, the proposed GA's solutions were better than or as well as the basic GA's for 52 problems. The basic GA only performed slightly better than the proposed GA did in the other four problems. Besides, the proposed GA even found solutions with fewer vehicles for ten problems.

6. Summary and conclusions

Simultaneous delivery and pickup problems have drawn much attention in the past few years, especially in coping with recycling issues for environmental protection. Customers, in reality, request specific service time; in order to increase the service quality, the logistic companies often provide services to meet such requests. This study therefore considered a vehicle routing problem with simultaneous delivery and pickup problems with time windows and formulated the problem into a mixed binary integer programming model denoted by SDPPTW.

Due to the NP nature of the problem, this study developed a co-evolution genetic algorithm with the use of some variants of the cheapest insertion method. Because there were no existing benchmarks, this study generated some test problems which were revised from the well-known Solomon's benchmark for Vehicle Routing Problem with Time Windows (VRPTW). Comparisons between the results of Cplex software, the proposed algorithm, and the basic genetic algorithm showed that the proposed algorithm could provide better solutions within a comparatively shorter period of time.

Since uncertainty often accompanies pickup demand, fuzzy mathematics may be applied to enhance the model and may provide a broader perspective to the research of the bi-directional logistics problem. Hence, determining how to extend the model to cope with uncertainty is a direction of the further study.

Acknowledgements

The authors acknowledge the financial support from the National Science Council, Taiwan, ROC, under Project No. NSC95-2221-E007-212.

Appendix A

Figs. A1–A10 illustrate all mutation operators.

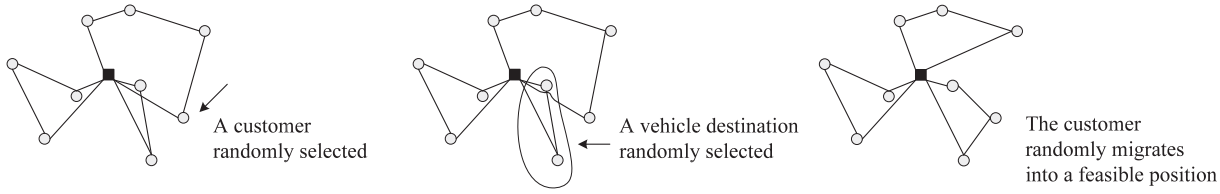


Fig. A1. Mutation_02: Random customer migration.

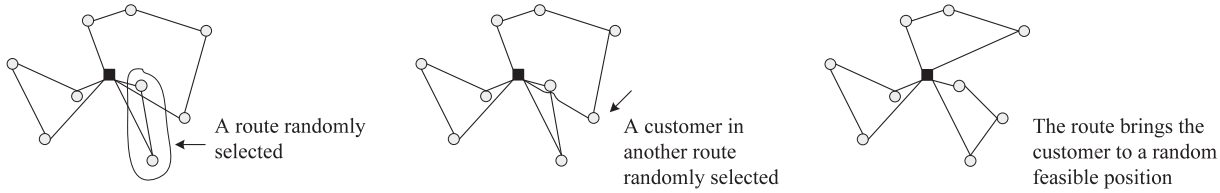


Fig. A2. Mutation_03: Bring a random customer.

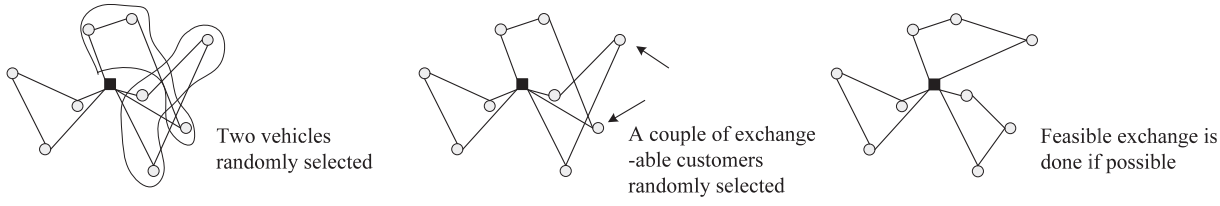


Fig. A3. Mutation_04: Random customer exchange.

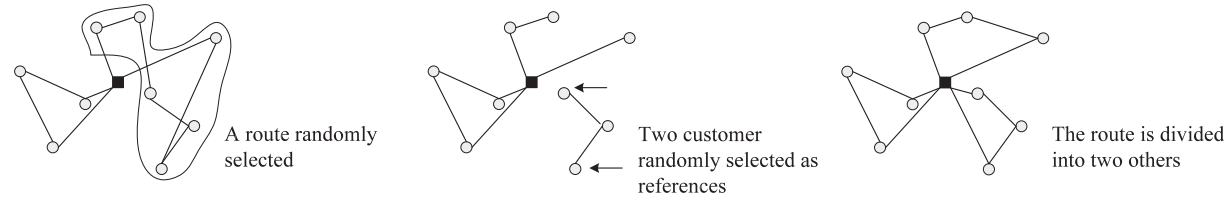


Fig. A4. Mutation_05: Route partitioning.

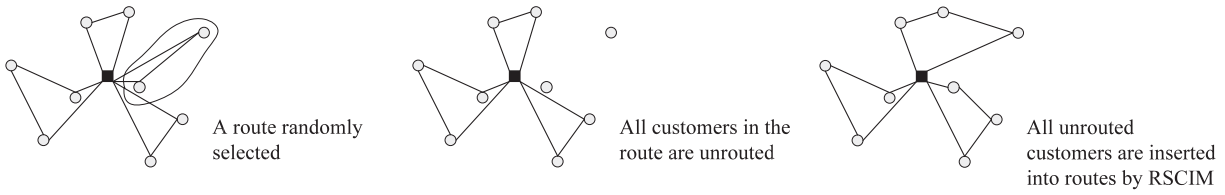


Fig. A5. Mutation_06: Reducing one route.

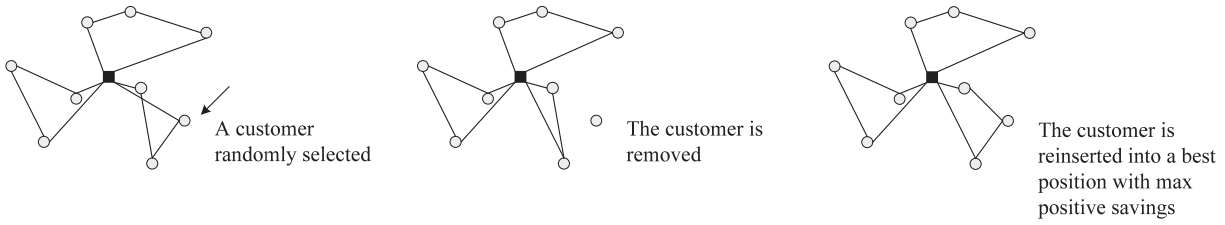


Fig. A6. Mutation_07: Customer reinsertion with best savings.

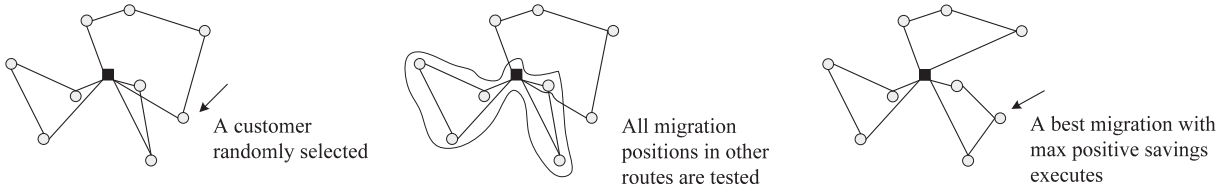


Fig. A7. Mutation_08: Customer migration with best savings.

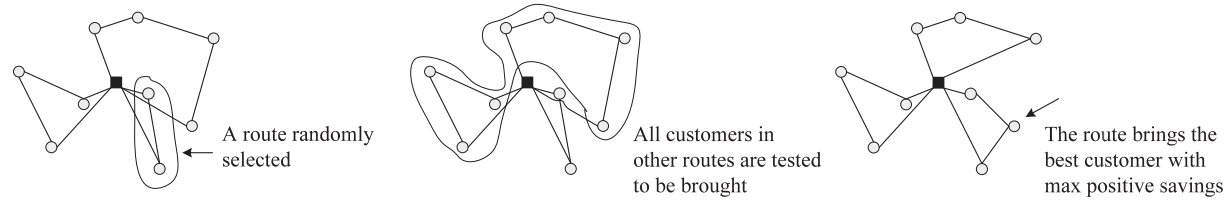


Fig. A8. Mutation_09: Bringing the best customer.

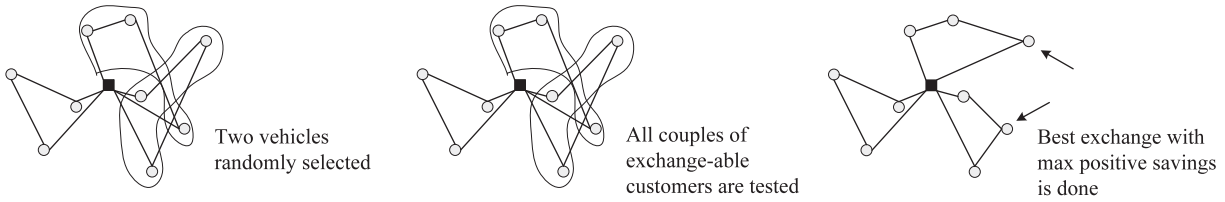


Fig. A9. Mutation_10: Best customer exchange.

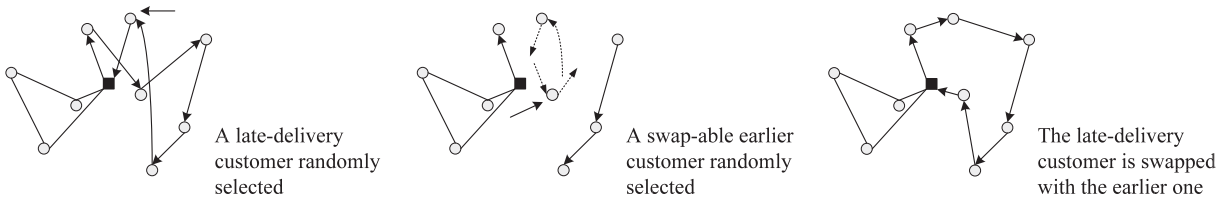


Fig. A10. Mutation_11: Early-pickup late delivery customer exchange.

References

Ai, T. J., & Kachitvichyanukul, V. (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers and Operations Research, 36*, 1693–1702.

Alvarenga, G. B., Mateusb, G. R., & De Tomi, G. (2007). A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. *Computers and Operations Research, 34*, 1561–1584.

Berbeglia, G., Cordeau, J. F., Gribkovskaia, I., & Laporte, G. (2007). Static pickup and delivery problems: A classification scheme and survey. *TOP, 15*, 1–31.

Bianchessi, N., & Righini, G. (2007). Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers and Operations Research, 34*, 578–594.

Bräysy, O., Dullaert, W., & Gendreau, M. (2004). Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics, 10*, 587–611.

Bräysy, O., & Gendreau, M. (2005a). Vehicle routing problem with time windows, part I: Route construction and local search algorithms. *Transportation Science, 39*, 104–118.

Bräysy, O., & Gendreau, M. (2005b). Vehicle routing problem with time windows, part II: Metaheuristics. *Transportation Science, 39*, 119–139.

Bräysy, O., Hasle, G., & Dullaert, W. (2004). A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research, 159*, 586–605.

Chen, J. F., & Wu, T. H. (2006). Vehicle routing problem with simultaneous deliveries and pickups. *Journal of the Operational Research Society, 57*, 579–587.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research, 12*, 568–581.

Cordeau, J. F., Desaulniers, G., Desrosiers, J., Solomon, M. M., & Soumis, F. (2001). VRP with time windows. In P. Toth & D. Vigo (Eds.), *The Vehicle Routing Problem* (pp. 157–193). Philadelphia: SIAM.

Cordeau, J. F., Gendreau, M., Hertz, A., Laporte, G., & Sormany, J. S. (2005). New heuristics for the vehicle routing problem. In A. Langevin & D. Riopel (Eds.), *Logistic systems, design and optimization* (pp. 279–297). New York: Springer.

Cordeau, J. F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks, 30*, 105–119.

Cordeau, J. F., Gendreau, M., Laporte, G., Potvin, J. Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society, 53*, 512–522.

Deif, I., & Bodin, L. (1984). Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling. In A. Kidder (Ed.), *Proceedings of the Babson conference on software uses in transportation and logistic management* (pp. 75–96). Babson Park.

Dell'Amico, M., Righini, G., & Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science, 40*, 235–247.

Dethloff, J. (2001). Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. *OR Spectrum, 23*, 79–96.

Gendreau, M., Laporte, G., & Potvin, J. Y. (2001). Metaheuristics for the capacitated VRP. In P. Toth & D. Vigo (Eds.), *The vehicle routing problem* (pp. 129–154). Philadelphia: SIAM.

Hoff, A., Gribkovskaia, I., Laporte, G., & Løkketangen, A. (2009). Lasso solution strategies for the vehicle routing problem with pickups and deliveries. *European Journal of Operational Research, 192*, 755–766.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbour: The University of Michigan Press.

Mester, D., Bräysy, O., & Dullaert, W. (2007). A multi-parametric evolution strategies algorithm for vehicle routing problems. *Expert Systems with Applications, 32*, 508–517.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research A, 23*(5), 377–386.

Montané, F. A. T., & Galvao, R. D. (2006). A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. *Computers and Operations Research, 33*(3), 595–619.

Nagy, G., & Salh, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research, 162*(1), 126–141.

Osman, I. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problems. *Annals of Operations Research, 41*, 421–451.

Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers and Operations Research, 34*, 2403–2435.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research, 31*, 1985–2002.

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

Ropke, S., & Pisinger, D. (2006b). A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research, 171*, 750–775.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research, 35*(2), 254–265.

Wang, H. F., & Chiu, Y. C. (2009). A vehicle routing and scheduling model for a distribution center. In H. F. Wang (Ed.), *Web-based green products life cycle management systems: Reverse supply chain utilization. Information science reference*. New York: Hershey.