IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received August 31, 2019, accepted September 13, 2019, date of publication October 8, 2019, date of current version October 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2946219

Location-Routing Problem With Demand Range

VINCENT F. YU[1], PANCA JODIAWAN[1], YI-HSUAN HO[1], AND SHIH-WEI LIN[2,3,4]

[1]Department of Industrial Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan
[2]Department of Information Management, Chang Gung University, Taoyuan 333, Taiwan
[3]Department of Neurology, Linkou Chang Gung Memorial Hospital, Taoyuan 333, Taiwan
[4]Department of Industrial Engineering and Management, Ming Chi University of Technology, New Taipei City 243, Taiwan

Corresponding author: Shih-Wei Lin (swlin@mail.cgu.edu.tw)

This work was supported in part by the Ministry of Science and Technology of the Republic of China (Taiwan) under Grant MOST 108-2221-E-011-051-MY3, and in part by the Center for Cyber-Physical System Innovation from the Featured Areas Research Center Program within the Framework of the Higher Education Sprout Project by the Ministry of Education of the Republic of China (Taiwan). The work of S.-W. Lin was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST107-2410-H-182-005-MY2, and in part by the Linkou Chang Gung Memorial Hospital under Grant BMRPA19.

ABSTRACT This research proposes a new variant of the location-routing problem (LRP) called LRP with Demand Range (LRPDR) by allowing flexibility in the delivery quantity. The goal of the LRPDR is to minimize the objective value calculated by the total cost minus the extra revenue. The total cost consists of the travelling cost of vehicles, the opening cost of the depots, and the activation cost of vehicles. This study proposes a new hybrid algorithm, SAPSO, that combines simulated annealing (SA) and particle swarm algorithm (PSO) for solving the LRPDR. Since this problem has not yet been studied in the literature, a mathematical model is proposed and solved by the Gurobi solver. The results obtained by Gurobi are then compared with those obtained by the proposed SAPSO algorithm. In addition, the performance of the proposed SAPSO algorithm is assessed by solving the LRP benchmark instances, and comparing the results with those of existing state-of-the-art algorithms for LRP. Based on the experimental results, the proposed SAPSO algorithm improves the performance of the basic SA algorithm and outperforms Gurobi. Moreover, the benefits of the LRPDR over LRP are presented in terms of total cost reduction.

INDEX TERMS Demand range, hybrid algorithm, location routing problem, particle swarm algorithm, simulated annealing.

I. INTRODUCTION

In order to sustain in a highly competitive environment, logistics companies have put numerous efforts to optimize their strategic, tactical, and operational planning. Determining the location of facilities and operational vehicle routes are the most common, yet essential, decisions that all logistics companies encounter. Solving these two problems interdependently allows the companies to gain more benefits, i.e. operational cost reduction since the opened facilities are selected by considering the operational routes of vehicles while serving customers. The Location-Routing Problem (LRP), as the name implies, combines these two planning tasks simultaneously. During recent years, LRP has gained interest and been studied to address various applications, e.g. environmental issues [1], traffic conditions [2], periodic location-routing problems [3]–[5], two-echelon location-routing problems [6]–[8], and location-routing problems with pickup and delivery [9], [10]. Generally, goods delivered to customers in a vehicle routing problem are set to a fixed amount. The idea of allowing flexibility on delivered quantity was first investigated by Campbell [11] to complement the Vendor Managed Inventory (VMI) policy, which authorizes suppliers to manage the inventories of retailers [12]. This means that the suppliers decide both the quantity and timing of deliveries. However, due to errors in demand forecasts and customer's lack of trust, VMI did not work well; therefore, adding limited delivery volume flexibility could result in potential cost savings in terms of the distance travelled by utilized vehicles. In particular, the total travelled distance could be reduced because the flexibility assumption allows vehicles to choose a combination of demand quantities [11]. Recently, Archetti *et al.* [13] further studied and confirmed the potential benefits of harnessing the concept of flexibility, i.e., delivery quantity and service time, in the vehicle routing problem.

Although LRP research has been developed to address various issues, the research integrating demand flexibility

The associate editor coordinating the review of this manuscript and approving it for publication was Shuihua Wang.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

into LRP has not been addressed in previous works. The study of demand flexibility is currently still limited to operational scope, i.e. vehicle routing issues, in spite of the aforementioned potential benefits. Therefore, in order to further increase the advantages of LRP, demand flexibility is integrated into LRP to become the Location Routing Problem with Demand Range (LRPDR). To the best of the authors' knowledge, this problem has not yet been studied in literature.

The LRPDR consists of a number of customers whose demands need to be fulfilled. A lower-bound and an upper-bound are defined for the amount of demand delivered to each customer. There are a number of potential capacitated supplying facilities with a particular fixed cost if the facility is selected. No split deliveries are allowed, thus, customers could receive goods once from only one facility. The vehicles used to make deliveries are homogenous, i.e., the capacity of each vehicle is the same. The objective of the LRP is to minimize the total cost, which consists of the fixed cost of selected facilities, the fixed cost of utilized vehicles, and the travel distance cost of vehicles. The LRPDR utilizes a more general objective, which considers the tradeoff between the aforementioned total cost and total delivered quantity to customers. In particular, the additional revenue generated by customers with extra delivered quantity is added to the objective function, and this type of customer creates incentives for the delivery company, in order to receive more delivery quantity.

The main contributions of the paper are as follows. This study first introduces the LRPDR, which is a new problem that addresses a well-known logistics problem, i.e., LRP with demand flexibility. The LRPDR increases the complexity of the classical LRP, as the delivered quantity must be determined. We formulate a mathematical programming model for the LRPDR. Since this problem has never been addressed in previous works, this study proposes a hybrid algorithm, called SAPSO, which combines Simulated Annealing (SA) and Particle Swarm Optimization (PSO) algorithm to solve the LRPDR. In addition, numerical experiments were conducted to justify the benefits of hybridizing SA and PSO by comparing the results obtained from SAPSO with those obtained by SA.

II. LITERATURE REVIEW

The LRPDR is a new problem that extends capacitated LRP (CLRP) by allowing demand flexibility. Therefore, this study first discusses the CLRP and its extensions, then, discusses the idea of integrating flexibility into the vehicle routing problem, and finally, presents the current applications of SA and PSO.

The CLRP has the objective of minimizing total distribution costs, which consists of opening cost of depots, fixed cost of utilized vehicles, and vehicles' travel cost. Therefore, determining the depots that should be opened, the number of utilized vehicles, and the route that each vehicle must travel to serve all customers are necessary decisions in CLRP. Tuzun and Burke [14] proposed a two-phase Tabu Search (TS) algorithm for CLRP. The first phase focuses on determining a set of opened depots, while the second phase is performed to build a set of routes from the opened depots.

Since Nagy and Salhi [15] conducted literature review on the location-routing problem, numerous researchers have started to address the problem with various methods. Belenguer *et al.* [16] proposed a branch-and-cut algorithm to solve the CLRP, where an integer linear program with several families of constraints was proposed and embedded in a cutting plane scheme to obtain a valid lower bound for the CLRP. The algorithm initially starts by solving a relaxation of linear program (LP). At each iteration, the solution produced by solving the relaxation of LP is checked to determine whether it violates any valid inequality, and the algorithm terminates when no violated inequality is found. They utilized three benchmark datasets. 14 out of 22 instances in the first two datasets could be solved to optimality, including an instance that had never been solved to optimality in previous works. Regarding the third dataset, the results show that all instances were solved to optimality.

Several metaheuristics are also employed to solve CLRP. Prins *et al.* [17] presented a metaheuristic consisting of two phases. GRASP, which is extended from the Clarke and Wright algorithm, was executed in the first phase and post-optimization using path-relinking was utilized in the second phase. Duhamel *et al.* [18] proposed a hybridization of GRASP and evolutionary local search (ELS) to solve CLRP. Two solution spaces – giant tours without trip delimiters and true CLRP solutions – were utilized during the search process. Yu *et al.* [19] developed a Simulated Annealing algorithm to tackle CLRP, where three neighborhood moves, swap move, insertion move, and 2-opt move, were employed. Hemmelmayr *et al.* [20] proposed an Adaptive Large Neighborhood Search to solve the two-echelon location routing problem (2E-LRP), an extension of CLRP by considering more than one echelon. The proposed ALNS was also utilized to solve CLRP benchmark problems, which outperformed the previous works.

The integration of demand flexibility into the vehicle routing problem was first addressed by Campbell [11]. Demand flexibility is defined by the upper and lower-bound values of the original delivery quantity for each customer, and an integer program is formulated to model the problem. The impact of flexibility on the total distance required to serve a set of customers was analyzed, and the results showed that allowing demand flexibility resulted in savings, in terms of travelled distance. Francis *et al.* [21] dealt with the periodic vehicle routing problem with service choices (PVRP-SC), which considers multiple periods of time where customers could be serviced several times in the considered planning periods. By allowing the flexibility of service frequency, system efficiency could be improved. Most recently, Archetti *et al.* [13] proposed an extension of PVRP-SC, called flexible periodic vehicle routing problem (FPVRP), by further relaxing constraints on the amount of delivered quantity, which resulted in higher flexibility. The authors analyzed the benefit of FPVRP by comparing the results of FPVRP with those of PVRP, and concluded that FPVRP resulted in savings, in terms of total travel distance.

SA is a well-known algorithm that has been used to solve various types of routing and scheduling problems. Recently, SA has been applied to solve vehicle routing problem with simultaneous pickup-delivery and time windows [22], open vehicle routing problem with cross-docking [23], and hybrid vehicle routing problem [24]. SA has also been developed to deal with several types of LRP, e.g. CLRP [25], open location routing problem (OLRP) [26], and two echelon OLRP [7]. PSO is a population-based algorithm, which is inspired by a flock of living creatures, e.g. birds and fishes. This algorithm was first developed by Eberhart and Kennedy [27]. The popularity of PSO has also been recognized in various optimization problems, e.g. TSP [28], VRP with stochastic demand [29], VRPSPD [30], berth allocation problem [31], cross-docking distribution problem [32], and team orienteering problem [33]. Considering the successful application of SA and PSO, this study proposes a hybridization of SA and PSO to tackle LRPDR, and hopes to obtain competitive results.

III. MATHEMATICAL MODEL

The LRPDR is described as follows. There is a set of customers, each with a known demand range and a set of potential depots, each with known coordinates. The objective of LRPDR is to minimize the sum of the costs associated with the fixed cost of opening the depots and using the vehicles, and the travel cost of the vehicles minus the total extra revenue. Each customer is assigned to one potential depot and served exactly once by a vehicle, which starts from and ends its route at the depot to fulfill customers' demand. The depots and vehicles are capacitated. Additionally, the load of the vehicle before returning to the depot should be zero.

In graph theory terms, let $G = (N, A)$ be a complete undirected graph, where N is the set of nodes ($N = N_0 \cup N_c$) and A is the set of arcs. Each arc $a = (i, j) \in A$ connects nodes i and j in set N and has a travel cost c_{ij} . N_0 is the set of potential depots. Each $k \in N_0$ has a capacity CD_k and an opening cost FD_k. N_c is the set of customers. Each $i \in N_c$ has a demand d_i, a unit extra revenue p_i, and a range of possible delivery quantity defined by β_i ($0 \leq \beta_i \leq 1$). In other words, the delivery quantity ranges from $(1-\beta_i)$ to $(1+\beta_i)$. Each vehicle has a fixed capacity CV and a fixed cost FV. The total initial load of the vehicles originate from depot k should not exceed the depot's capacity CD_k.

The goals of LRPDR are to determine: (1) Which potential depots should be opened. (2) Which routes should be serviced by vehicles assigned to such depot. (3) Which combination of delivery quantity for each customer results in the least objective value. The decision variables are as follows:

x_{ijk} = 1 if vehicle k travels directly from node i to node $j, \forall i, j \in N, i \neq j, \forall k \in K$. 0 otherwise;

y_d = 1 if depot d is opened, $\forall d \in N_0$, 0 otherwise;

z_{id} = 1 if customer i is assigned to depot d, $\forall i \in N_c, \forall d \in N_0$, 0 otherwise;

U_{ij} = Remaining delivery demand after leaving node i for node $j, \forall i, j \in N, i \neq j$;

q_{ik} = Quantity delivered to customer i by vehicle $k, \forall i \in N_c, \forall k \in K$.

$$\min z = \sum_{k\in K}\sum_{i\in N}\sum_{j\in N, i\neq j} c_{ij}x_{ijk} + \sum_{d\in N_0} FD_d y_d + \sum_{k\in K}\sum_{d\in N_0}\sum_{i\in N_c} FV x_{dik} - \sum_{i\in N_c}\sum_{k\in K} p_i q_{ik} \quad (1)$$

Subject to

$$\sum_{k\in K}\sum_{j\in N, j\neq i} x_{ijk} = 1, \quad \forall i \in N_c \quad (2)$$

$$\sum_{i\in N, i\neq j} x_{ijk} = \sum_{j\in N, i\neq j} x_{jik}, \quad \forall i \in N, \forall k \in K \quad (3)$$

$$U_{ij} \leq CV \sum_{k\in K} x_{ijk} \quad (4)$$

$$\sum_{i\in N_c} U_{id} = 0, \quad \forall d \in N_0 \quad (5)$$

$$\sum_{d\in N_0} z_{id} = 1, \quad \forall i = N_c \quad (6)$$

$$\sum_{k\in K} x_{idk} \leq z_{id}, \quad \forall i \in N_c, \forall d \in N_0 \quad (7)$$

$$\sum_{k\in K} x_{dik} \leq z_{id}, \quad \forall i \in N_c, \forall d \in N_0 \quad (8)$$

$$\sum_{k\in K} x_{ijk} + z_{id} + \sum_{m\in N_0, m\neq d} z_{jm} \leq 2, \quad \forall i, j \in N_c, i \neq j, \quad \forall d \in N_0 \quad (9)$$

$$\sum_{i\in N_c} q_{ik} \leq CV, \quad \forall k \in K \quad (10)$$

$$\sum_{k\in K} q_{ik} \geq (1-\beta_i)d_i, \quad \forall i \in N_c \quad (11)$$

$$\sum_{i\in N, i\neq j} (1+\beta_j)d_j x_{jik} \geq q_{jk} \quad , \forall j \in N_c, \forall k \in K \quad (12)$$

$$0 \leq q_{ik} \leq (1+\beta_i)d_i, \quad \forall i \in N_c, \forall k \in K \quad (13)$$

$$\sum_{j\in N, i\neq j} U_{ji} - \sum_{j\in N, i\neq j} U_{ij} = \sum_{k\in K} q_{ik}, \quad \forall i \in N_c, \forall k \in K \quad (14)$$

$$\sum_{i\in N_c} U_{di} = \sum_{k\in K}\sum_{i\in N_c} z_{id} q_{ik}, \quad \forall d \in N_0 \quad (15)$$

$$\sum_{k\in K}\sum_{i\in N_c} q_{ik} z_{id} \leq CD_d y_d, \quad \forall d \in N_0 \quad (16)$$

$$U_{ij} \leq CV \sum_{k\in K} x_{ijk} - \sum_{k\in K} x_{ijk} q_{ik}, \quad \forall i \in N_c, \quad \forall j \in N, i \neq j \quad (17)$$

$$U_{ij} \geq q_{jk} \sum_{k\in K} x_{ijk}, \quad \forall i \in N, \forall j \in N_c, i \neq j \quad (18)$$

12	0	4	3	8	0	6	1	2	5	7	9	10	11
*	*	1680	960	120	*	320	880	560	2520	840	400	480	*

FIGURE 1. Solution representation of an LRPDR instance.

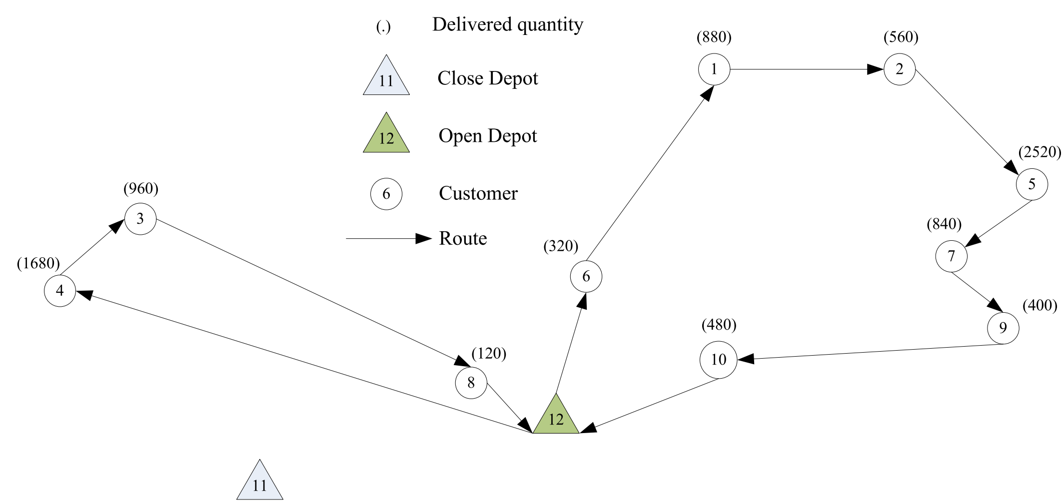


FIGURE 2. Illustration of an LRPDR solution.

$$\sum_{i\in N_0}\sum_{j\in N_c} x_{ijk} \leq 1, \quad \forall k \in K \tag{19}$$

$$x_{ijk} \in \{0, 1\}, \quad \forall i,\ j \in N,\ \forall k \in K \tag{20}$$

$$y_d \in \{0, 1\}, \quad \forall d \in N_0 \tag{21}$$

$$z_{id} \in \{0, 1\},\ \forall i \in N_c, \tag{22}$$

$$U_{ij} \geq 0,\ \forall i,\ j \in N \tag{23}$$

$$q_{ik} \geq 0, \quad \forall i \in N_c,\ \forall k \in K \tag{24}$$

Objective (1) minimizes the total cost minus the extra revenue. The total cost consists of vehicle traveling cost, depot opening cost, and vehicle fixed cost. Constraint (2) ensures that each customer is visited at most once. Constraint (3) ensures that the number of entering arcs is equal to the number of leaving arcs for each node. Constraint (4) states that the remaining demand does not exceed the capacity of the vehicle at any time. Constraint (5) guarantees that there is no remaining delivery quantity in the vehicle after the vehicle has serviced its last customer. Constraint (6) denotes that each customer is assigned to one depot. Constraints (7) to (9) prohibit infeasible routes. Constraint (10) restricts that the load of a vehicle does not exceed the vehicle's capacity. Constraints (11) to (13) ensure that the delivery quantity for each customer is within its demand range. Constraint (14) is the flow constraint for delivery quantity. Constraint (15) ensures that the total delivery quantity to customers assigned to a specific depot is satisfied by the vehicles dispatched from the depot. Constraint (16) guarantees that the total delivered quantity of the vehicles dispatched from a depot does not exceed the capacity of the depot. Constraints (17) and (18) are the bounds of the variable U_{ij}. Constraint (19) restricts that each vehicle is assigned to at most one depot.

Constraints (20) to (22) define all binary decision variables. Constraints (23) and (24) define all non-negative continuous decision variables.

IV. SAPSO HEURISTIC FOR LRPDR

A. SOLUTION REPRESENTATION

The solution representation of LRPDR is divided into two parts. The first part is a permutation of n customers $\{1, 2, 3, \ldots, n\}$, m potential depots $\{n+1, n+2, n+3, \ldots, n+m\}$, and N_{dummy} zeros that are used to terminate routes. The formula of calculating N_{dummy} is $\lceil \sum_i (1+\beta_i)d_i/CV \rceil$, where $(1+\beta_i)d_i$ is the maximum of the delivery quantity of customer i, and CV denotes vehicle capacity. The second part shows the potential combinations of each customer's delivery quantity.

In the solution representation, the first number in a route must be a depot. Customers are added one by one into the current route of the current depot, provided that the capacity constraint of the route is not violated. The purpose of using zero in the solution representation is to terminate a route and start a new route, even though the accumulated demand has not exceeded vehicle capacity. Consequently, it results in a larger search space and increases the possibility to find a better solution.

To demonstrate the solution representation, an example of LRPDR solution is shown in Fig. 1 and Fig. 2, which provides a sample solution representation and a visual illustration of the distribution network corresponding to the sample solution representation. In this example, there are 10 customers and 2 potential depots, as shown in Table 1 and Table 2. Table 1 displays the coordinates (X, Y), demand (d_i), and the unit extra revenue (p_i) for each customer. Table 2 lists

TABLE 1. Customer information of an LRPDR instance.

Customer no.	X	Y	d_i	*Pi*
1	151	264	1100	0
2	159	261	700	0.0019562
3	130	254	800	0.0026941
4	128	252	1400	0.0070953
5	163	247	2100	0.0065753
6	146	246	400	0.0024951
7	161	242	800	0.0044177
8	142	239	100	0.0013147
9	163	236	500	0
10	148	232	600	0

TABLE 2. Depot information of an LRPDR instance.

Depot no.	*X*	*Y*	Depot opening cost	Depot capacity
11	136	194	50	15000
12	143	237	50	15000

the (*X, Y*) coordinates, opening cost, and capacity of the two potential depots. The vehicle capacity is 6,000. β_i is 0.2 for each customer.

V. INITIAL SOLUTION

The greedy algorithm is applied to construct an initial solution. The following is the procedure for generating an initial solution in the proposed SAPSO.

Step 1: for each unassigned depot d in N_0, calculate the number of customers whose closest depot is depot d. Choose the depot with the largest number of customers. If two or more depots have the same number of customers, then select the depot with the largest capacity among these depots.

Step 2: arrange unassigned customers to the chosen depot in step 1 by an increasing order of distance between the customer and the depot until the capacity of the depot runs out. Next, eliminate the selected depot from further considerations.

Step 3: construct a TSP tour from selected customers to the chosen depot. This tour should start from and end at the chosen depot.

Step 4: divide the TSP tour into several routes based on the vehicle capacity constraint to ensure the feasibility of each route.

Step 5: if there is any unassigned customer, go to step 1; otherwise, the procedure ends.

A. NEIGHBORHOOD STRUCTURES

Insertion, swap, and reverse are three neighborhood moves commonly used in SA heuristic. Therefore, the proposed SAPSO uses these three neighborhood moves to find a better potential solution in the neighboring area of the current solution. Each move is equally likely to be selected. The positions of each move are randomly picked in the solution, and the selected positions could be a depot, a customer, or a zero. Therefore, feasibility check is performed for the newly generated solution.

Regarding the swap move, first, we randomly choose two positions in the solution. Second, we exchange these two positions to obtain a new solution. For the reverse move, first, we randomly choose two solution positions; second, we reverse the substring between these two positions to obtain a new solution. Regarding the insertion move, first, we randomly choose two positions of a solution, and then insert the first chosen position after the second chosen position to obtain a new solution. After performing one of these three moves, feasibility check is conducted. The move is repeated until a new feasible solution is produced.

B. THE SAPSO HEURISTIC

This research develops a hybrid metaheuristic that combines SA and PSO to solve LRPDR. Fig. 3 presents the procedure of the proposed SAPSO. Ten parameters are required to execute SAPSO. I_{iter} represents the number of inner iterations for the SA procedure, $N_{non-improving}$ represents the number of temperature reductions allowed without improving the current best solution. T_0 and T_f represent the initial temperature and the final temperature in the SA procedure, respectively. α is a constant used to update the current temperature, while K is a constant used to calculate the acceptance probability of a worse solution in the SA procedure. $N_{particle}$, ψ_{iter}, $N_{reinitializing}$, and V_{limit} are used in the ShakeDemandPSO(.) procedure to represent the number of particles, number of iterations, number of iterations before the reinitializing procedure, and the speed limitation of a particle, respectively. The value of current temperature T is first set to T_0. Then, the greedy algorithm is applied to generate an initial solution X as the current solution. Next, the current best solution X_{best} and the current best objective value F_{best} are set to be X and its objective value, respectively.

In the improvement phase, a new solution Y is generated from the current solution X by one of the three aforementioned neighborhood moves. After that, let Δ be the difference between the new solution and the current solution, i.e. $\Delta = obj(Y) - obj(X)$. If $\Delta < 0$, then the current solution X is replaced with the new solution Y. Otherwise, a random number $r \sim$ U(0,1) is generated. If r is smaller than the probability calculated by $\exp(-\Delta/KT)$, then X is replaced with Y. If the new solution Y replaces the current solution X, then the ShakeDemandPSO(.) procedure based on the PSO algorithm has a chance to be utilized to search for a potential delivery quantity combination in a larger space. The purpose of this procedure is to further improve the new solution Y before it replaces the current solution X. The best solution X_{best} and its objective function value F_{best} are updated whenever a new best solution is found. After a given number of iterations, a local search is performed on X_{best}, and the current temperature decreases by the formula $T = \alpha T$, where $0 < \alpha < 1$.

The purpose of the proposed local search is to improve the current best solution, which is conducted with swap and insertion moves. Each of them is conducted 100 times. We use a random number to decide their sequence. Whenever

```
SAPSO(I_iter, N_non-improving, T_0, T_f, α, K, N_particle, φ_iter, N_reinitializing, V_limit)
begin
    Generate an initial solution X by the greedy algorithm;
    X_best := X; F_best := obj(X);

    N_dummy := ⌈ Σ_{i∈N_c} (1+β_i)d_i / CV ⌉ ; T:=T_0; ζ:= 0;

    while (T > T_f) {
        I := 0; ζ:= ζ+1;
        while (I < I_iter){
            r_1 := random(1,3);
            switch (r_1) {
                case 1:
                    do{
                        Generate a new solution Y from X by swap move;
                    }while (Y is infeasible);
                case 2:
                    do{
                        Generate a new solution Y from X by inverse move;
                    }while (Y is infeasible);
                case 3:
                    do{
                        Generate a new solution Y from X by reverse move;
                    }while (Y is infeasible);
            }
            Δ := obj(Y) − obj(X);
            if (Δ < 0) {
                r_2 := random(0,1)
                if (r_2 < 0.5){
                    X := ShakeDemandPSO( X, N_particle, φ_iter, N_reinitializing, V_limit);
                    X := Y;
                }
            }
            else {
                η = random (0,1);

                If (η < e^(−Δ/KT)) {
                    X := Y;
                }
            }
            if (obj(X) < F_best) {X_best := X; F_best = obj(X); ζ:= 0}
            I = I + 1;
        }
        if (ζ = N_non-improving) {Terminate the SAPSO heuristic;}
        T := αT;
    }
end
```

FIGURE 3. Pseudocode of the proposed SAPSO algorithm.

one of the terminating conditions occurs, the algorithm is terminated. There are two terminating conditions: (1) when the current T is below or equal to the final temperature T_f, and (2) when the best solution X_{best} has not improved for $N_{non-improving}$ consecutive temperature reductions.

The PSO algorithm is hybridized with SA as the ShakeDemandPSO(.) procedure, which is explained in Fig. 4. The SA algorithm can only adjust the sequence of visited customers; therefore, other mechanism is needed to set the amount of delivery quantity. The solution representation of the PSO algorithm naturally consists of continuous variables; therefore, demand quantity could be directly utilized in the solution representation without any intermediate process, which may increase the computational time.

This procedure generated the initial solution by creating the positions of all particles X_{ij}, where each particle i represents one combination of delivery quantity for customer j. Therefore, this uniformly creates particles at random within the allowable demand range $[(1-\beta_i)d_j, (1+\beta_i)d_j]$ of customer j. Similarly, this procedure creates initial particle velocities v_{ij} according to V_{limit}, which is uniformly distributed within the range $[v_{min}, v_{max}]$. Next, we evaluate the objective values of all particles. x_{Pbest} and P_{best} are initially set to the

```
ShakeDemandPSO( X, N_particle, φ_iter, N_reinitializing, V_limit)
begin
    Generate an initial solution X
    I := 0; P_best = obj(X); x_Pbest := x,
    G_best := min_{p∈N_particle} (obj(X_p)), x_Gbest := arg min_{p∈N_particle} (obj(X_p));

    while (I < φ_iter) {

        for each p ∈ N_particle {

            update the velocity and position of X_p (v_p and x_p);
            calculate obj(X_p);
            if (obj(X_p) < P_best){
                P_best := obj(X_p); x_Pbest := x_p;
                if (P_best < G_best) {
                    G_best := obj(X_p); x_Gbest := x_p;
                }
                if (P_best = G_best){
                    N := N + 1;
                    if (N = N_reinitializing){
                        Reinitialize velocity and position of X_p;
                    }
                }
            }
        }
    I = I + 1;
    }
end
```

FIGURE 4. Pseudocode of the ShakeDemandPSO(.) procedure.

initial solution and its objective value, respectively. In subsequent iterations, x_{Pbest} is the location of the best objective function found by particle i. G_{best} is the best objective value of all particles, and x_{Gbest} is the location of G_{best}.

After initialization, the iterations begin. We then update the velocity and position of each particle by the following equations:

$$v_{ij}(t+1) = k \times \begin{bmatrix} v_{ij}(t) + r_1 C_1 (x_{Pbest} - x_{ij}(t)) \\ +r_2 C_2 (x_{Gbest} - x_{ij}(t)) \end{bmatrix} \tag{25}$$

$$k = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|} \text{ where } \varphi = C_1 + C_2, \varphi > 4 \tag{26}$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \tag{27}$$

v_{ij}(t) is the velocity of particle i in dimension j at time t; x_{ij}(t) is the position of particle i in dimension j at time t; x_{Pbest} is the personal best position of particle i in dimension j found so far at time t; x_{Gbest} is the global best position of particle i in dimension j found so far at time t. C_1 and C_2 are positive constants; r_1 and r_2 are random numbers in $(-1,1)$. Equation (25) and (26) update the velocity and the position of the particles, respectively.

The procedure used to update P_{best} and G_{best} is explained as follows. If the new solution is better than the previous one,

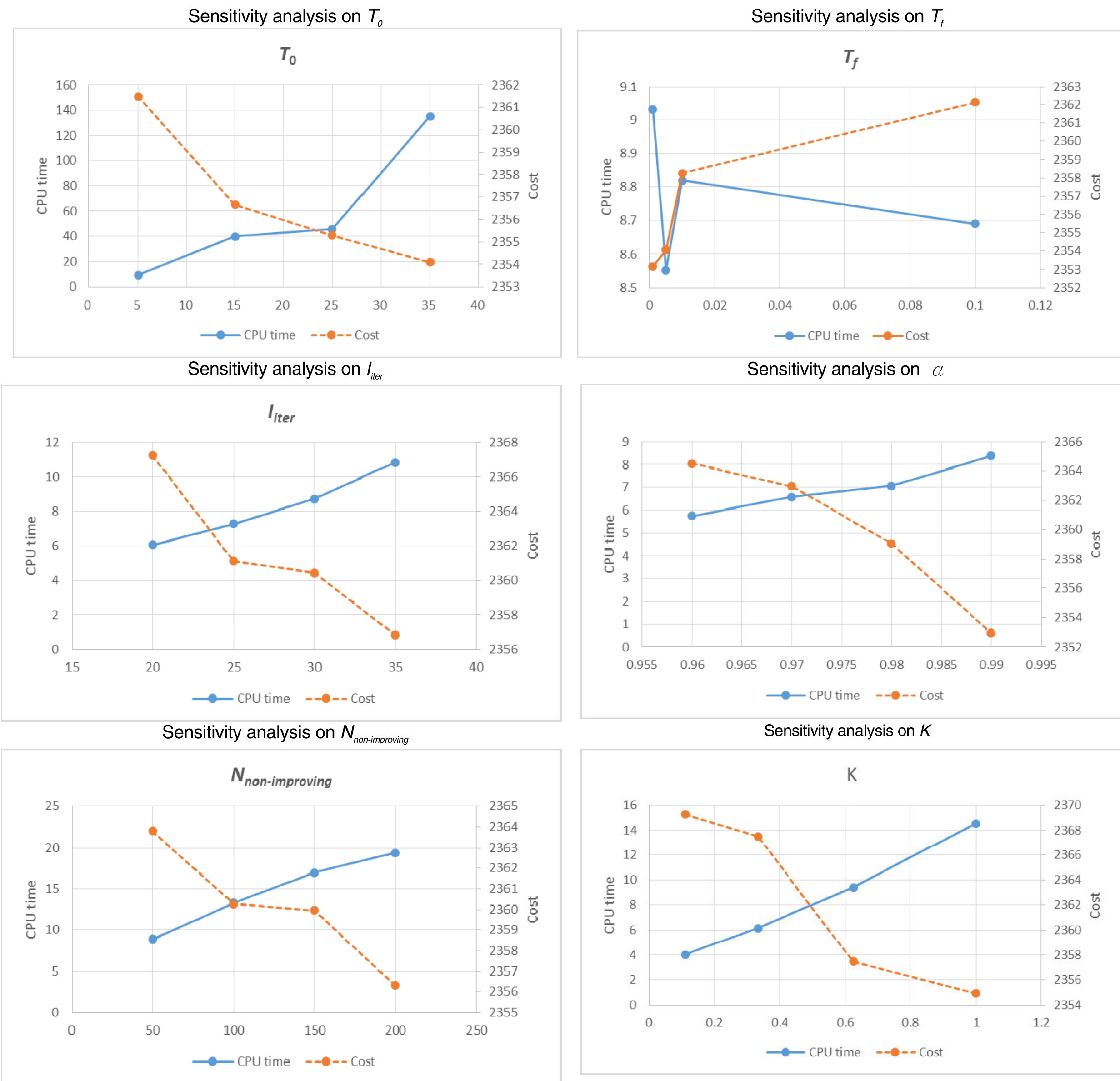


FIGURE 5. Sensitivity analyses of algorithmic parameters.

then the value of P_{best} and G_{best} will be updated; otherwise, proceed to the next iteration to find a new solution until the maximum number of iterations is reached. Furthermore, for each particle i, if the number of times that P_{best} equals G_{best} is equal to $N_{reinitialization}$, then particle i will be reinitialized. This condition is used to avoid getting stuck in a local optima before the algorithm terminates.

VI. COMPUTATIONAL STUDY

This section presents a numerical experiment to demonstrate the performance of the proposed SAPSO. The algorithm is implemented in C++ using Microsoft Visual Studio 2015, and the experiments are conducted on a computer with an Intel (R) Core (TM) i7-4790 3.6 GHz processor and 16 GB of RAM under Windows 10 Professional operating system. First, this research tests the proposed SAPSO algorithm on three sets of LRP benchmark instances, in order to assess

TABLE 3. Parameter values tested in experiments.

		I_{iter}	T_0	T_f	$N_{non\text{-}improving}$	α	K
LRP instance	Low	3000 L	30	0.01	50	0.97	1/9
	High	5000 L	40	0.1	150	0.98	1/1.6
LRPDR small instance	Low	30 L	5	0.01	50	0.98	1/1.6
	High	35 L	15	0.1	150	0.99	1
LRPDR original instance	Low	100 L	20	0.01	100	0.98	1/1.6
	High	110 L	25	0.05	150	0.99	1

TABLE 4. Comparison of proposed SAPSO and the best known solution (bks) on Barreto [34] dataset.

Instance	Customer	Depot	BKS	*bestCPU*	*best*Obj	*bestGap*[a]
Cli12x2	12	2	204.00	16.20	204.00	0.00
coordGaspelle1	21	5	424.90	35.43	424.90	0.00
coordGaspelle2	22	5	585.10	28.96	585.10	0.00
coordMin27	27	5	3062.00	31.06	3062.00	0.00
coordGaspelle3	29	5	512.10	46.79	512.10	0.00
coordGaspelle4	32	5	562.20	54.74	562.20	0.00
coordGaspelle5	32	5	504.30	60.76	504.30	0.00
coordGaspelle6	36	5	460.40	65.55	460.40	0.00
coordChrist50	50	5	565.60	102.13	565.60	0.00
Cli55x15	55	15	1112.10	199.56	1112.10	0.00
coordChrist75	75	10	844.40	199.84	844.58	0.02
Cli85x7	85	7	1622.50	294.67	1623.74	0.08
coordDas88	88	8	355.80	227.13	355.80	0.00
coordChrist100	100	10	833.40	274.28	833.40	0.00
coordOr117	117	14	12290.30	272.02	12304.90	0.12
coordMin134	134	8	5709.00	428.56	5719.25	0.18
coordDas150	150	10	43919.90	413.78	44185.80	0.61
Cli318x4	318	4	557275.20	1135.88	577126.00	3.56
Cli318x4_2	318	4	663070.00	1448.35	688415.00	3.82
		Average	4585.25	170.95	4603.51	0.44

[a] *best*GAP (%): percentage gap is calculated as (SAPSO best objective value - BKS) *100 /BKS.

its performance. Second, LRPDR benchmark datasets are generated from the LRP benchmark instances. We generate the upper bounds for each LRPDR instance by Gurobi, and utilize the SA algorithm to solve each instance, in order to assess the performance of the SAPSO algorithm in solving LRPDR.

A. TEST INSTANCES

Three benchmark instances of LRP are utilized to assess the performance of SAPSO. The first dataset contains 20 instances, which are adopted from Barreto [34]. The number of depots ranges from 2 to 14, while the number of customers varies between 8 and 318. The second dataset created by Prins *et al.* [35] consists of 30 instances. The number of depots is either 5 or 10, and the number of customers is chosen from the set {20, 50, 100, 200}. Finally, the third dataset of Tuzun and Burke [14] consists of 36 instances. The number of depots is either 10 or 20, and the number of customers is either 100, 150, or 200. Consequently, 85 instances are utilized to assess the performance of our algorithm in solving CLRP.

For assessing the performance of the algorithm in solving LRPDR, the dataset provided by Barreto [34] is adopted. The properties of the dataset, i.e. the coordinates of customers and depots, the capacity of vehicles and depots, the opening cost of depots, the fixed cost of the vehicles, and the original demands of customers, are all adopted. A new parameter, β_i, is used for the LRPDR dataset to define the demand range of customers. More specifically, in order to define the maximum and minimum amounts of delivery quantity, we multiply the original demand d_i by $(1+\beta_i)$ and $(1-\beta_i)$, respectively, for each customer i. In practice, the range of demand could be obtained by using the historical records of delivered quantity to the customers, or based on the agreement between suppliers and customers. Next, the extra revenue is generated, as follows. Half of the customers have the value of p_i equal to 0, which means adding an extra amount of delivery quantity brings no extra revenue. The p_i values of the other half of the customers are uniformly generated in (0.0, 0.01). The test instances are available at http://web.ntust.edu.tw/~vincent/lrp/.

B. PARAMETER SETTINGS

As it may affect performance, parameter tuning is an essential step in assessing the performance of algorithms. Thus, this study performs a preliminary experiment to set the best parameter settings. The tested parameter values are listed in Table 3. Each parameter combination is run five times, and the minimum average value of the objective is utilized to select the best combination. Two experiments are conducted; the first experiment was to determine the best parameter value combination in solving the LRP benchmark instances; while the second was performed to select the best parameter value combination for solving the LRPDR. Based on the first experiment, using $I_{iter} = 3000L$, $N_{non-improving} = 150$,

TABLE 5. Comparison of proposed SAPSO and the best known solution (bks) on Prins *et al.* [35] dataset.

Instance	Customer	Depot	BKS	*bestCPU*	*best*Obj	*bestGap*[a]
coord20-5-1	20	5	54793	25.05	54793	0.00
coord20-5-1b	20	5	39104	22.12	39104	0.00
coord20-5-2	20	5	48908	26.88	48908	0.00
coord20-5-2b	20	5	37542	22.58	37542	0.00
coord50-5-1	50	5	90111	63.17	90111	0.00
coord50-5-1b	50	5	63242	58.80	63242	0.00
coord50-5-2	50	5	88298	68.06	89838	1.74
coord50-5-2b	50	5	67308	58.42	67308	0.00
coord50-5-2bBIS	50	5	51822	59.66	51822	0.00
coord50-5-2BIS	50	5	84055	69.79	84055	0.00
coord50-5-3	50	5	86203	65.74	86203	0.00
coord50-5-3b	50	5	61830	58.96	61830	0.00
coord100-5-1	100	5	274814	169.87	275901	0.40
coord100-5-1b	100	5	213615	144.33	214549	0.44
coord100-5-2	100	5	193671	180.10	193929	0.13
coord100-5-2b	100	5	157095	177.46	157222	0.08
coord100-5-3	100	5	200079	201.24	201606	0.76
coord100-5-3b	100	5	152441	183.36	153528	0.71
coord100-10-1	100	10	287695	230.94	309019	7.41
coord100-10-1b	100	10	230989	168.14	238566	3.28
coord100-10-2	100	10	243590	328.07	244518	0.38
coord100-10-2b	100	10	203988	165.12	204422	0.21
coord100-10-3	100	10	250882	171.30	255142	1.70
coord100-10-3b	100	10	204317	179.51	205672	0.66
coord200-10-1	200	10	475294	672.04	481497	1.31
coord200-10-1b	200	10	377043	732.97	383539	1.72
coord200-10-2	200	10	449006	700.47	452421	0.76
coord200-10-2b	200	10	374280	530.36	376039	0.47
coord200-10-3	200	10	469433	564.20	476023	1.40
coord200-10-3b	200	10	362653	489.34	364637	0.55
		Average	196470.03	219.60	198766.20	0.80

[a] *best*GAP (%): percentage gap is calculated as (SAPSO best objective value - BKS) *100 /BKS.

$T_0 = 40$, $T_f = 0.1$, $K = 1/1.6$, and $\alpha = 0.98$ leads to the best results among all possible combinations. For the second experiment, the selected parameter values are as follows: $I_{iter} = 30\text{L}$, $N_{non-improving} = 50$, $T_0 = 5$, $T_f = 0.01$, $K = 1/1.6$, and $\alpha = 0.99$ for small LRPDR instances; and $I_{iter} = 110\text{L}$, $N_{non-improving} = 150$, $T_0 = 25$, $T_f = 0.01$, $K = 1/1.6$, and $\alpha = 0.98$ for original-size LRPDR instances. Therefore, these combinations are utilized in subsequent studies.

Sensitivity analysis is then performed to provide more insights on the effects of the parameter settings. In the analysis, we change one parameter at a time from the best parameter combination to solve small LRPDR instances to determine the effect. The blue curve and orange curve represent CPU time and objective value, respectively. The initial temperature is increased in increments of 10 starting from 5 to 35. The initial temperature may influence the probability of accepting a worse solution. The higher the initial temperature, the higher the probability.

The final temperature is decreased from 0.1 to 0.001. Generally, the lower the final temperature, the lower the cost, and the higher the computational time; however, according to the results, the computational time is different. This may be because the algorithm is terminated by one of the stopping criteria, i.e. $N_{non-improving}$, thus, the solution stops earlier than usual.

This study increased the number of iterations in increments of $10L$ starting from $20L$ to $50L$. While more iterations improves solution quality, it requires additional computational time. This study increased the value of $N_{non-improving}$ in increments of 50 starting from 50 to 200. As higher

TABLE 6. Comparison of proposed SAPSO and the best known solution (bks) on Tuzun and Burke [14] dataset.

Instance	Customer	Depot	BKS	*bestCPU*	*best*Obj	*bestGap* [a]
coordP111112	100	10	1467.68	259.17	1468.29	0.04
coordP111122	100	20	1449.20	318.02	1449.20	0.00
coordP111212	100	10	1394.80	282.89	1394.80	0.00
coordP111222	100	20	1432.29	321.38	1432.29	0.00
coordP112112	100	10	1167.16	284.81	1167.16	0.00
coordP112122	100	20	1102.24	317.04	1102.24	0.00
coordP112212	100	10	791.66	289.75	791.66	0.00
coordP112222	100	20	728.30	317.65	728.30	0.00
coordP113112	100	10	1238.24	281.34	1238.49	0.02
coordP113122	100	20	1245.31	323.04	1247.27	0.16
coordP113212	100	10	902.26	281.27	902.26	0.00
coordP113222	100	20	1018.29	317.52	1018.29	0.00
coordP131112	150	10	1866.75	502.69	1897.34	1.64
coordP131122	150	20	1823.20	539.73	1831.26	0.44
coordP131212	150	10	1964.30	501.87	1994.01	1.51
coordP131222	150	20	1792.80	544.43	1814.16	1.19
coordP132112	150	10	1443.33	503.33	1447.63	0.30
coordP132122	150	20	1434.60	547.45	1443.05	0.59
coordP132212	150	10	1204.42	503.87	1204.93	0.04
coordP132222	150	20	930.99	552.12	931.28	0.03
coordP133112	150	10	1694.18	503.38	1716.70	1.33
coordP133122	150	20	1392.00	543.18	1405.11	0.94
coordP133212	150	10	1198.20	505.44	1202.25	0.34
coordP133222	150	20	1151.80	542.40	1153.55	0.15
coordP121112	200	10	2243.40	803.14	2249.89	0.29
coordP121122	200	20	2138.40	839.14	2160.48	1.03
coordP121212	200	10	2209.30	785.40	2242.77	1.51
coordP121222	200	20	2222.90	831.07	2243.56	0.93
coordP122112	200	10	2073.70	790.19	2088.88	0.73
coordP122122	200	20	1692.17	834.37	1698.87	0.40
coordP122212	200	10	1453.18	789.96	1463.20	0.69
coordP122222	200	20	1082.46	844.85	1085.27	0.26
coordP123112	200	10	1954.70	788.25	1968.43	0.70
coordP123122	200	20	1918.93	845.14	1923.51	0.24
coordP123212	200	10	1762.00	795.30	1764.69	0.15
coordP123222	200	20	1390.87	839.16	1392.60	0.12
		Average	1499.33	546.38	1507.32	0.44

[a] *best*GAP (%): percentage gap is calculated as (SAPSO best objective value - BKS) *100 /BKS.

$N_{non-improving}$ results in higher computational time, when the termination condition is relaxed, a better solution appears.

Regarding the value of K, the tested values are 1/9, 1/3, 1/1.6, and 1. Increasing Boltzmann's constant K raises the computational time, as it affects the probability of accepting a worse solution. This study increased the value of α in increments of 0.01 starting from 0.96 to 0.99, and the increasing trend in both solution quality and computational time is shown. The smaller the alpha, the less iterations are executed, and the faster the convergence velocity, and this explanation is deduced from Fig. 5.

C. COMPUTATIONAL RESULTS

This section shows the effectiveness of the proposed SAPSO heuristic by solving the LRPDR datasets, and then, comparing the results with those obtained by Gurobi and SA heuristic. Gurobi solutions are obtained within a predetermined five-hour limit. In addition, we use SAPSO to solve CLRP,

TABLE 7. Comparison of different methods for the three benchmark sets.

Algorithm	Barreto		Prins et al.		Tuzun and Burke		All	
	CPU	AvgBestGap(%)	CPU	AvgBestGap(%)	CPU	AvgBestGap(%)	avgCPU	*avg*Gap(%)
GRASP+ELS (Duhamel et al., 2010)	**187.62**	**0.08**	258.17	1.11	606.64	1.30	**350.81**	**0.83**
SALRP (Yu et al., 2010)	464.07	0.49	422.43	0.46	826.47	1.49	570.99	0.81
ALNS-500K (Hemmelmayr et al., 2012)	**177.23**	**0.16**	451.10	0.44	829.64	0.44	**485.99**	**0.35**
ALNS-5000K (Hemmelmayr et al., 2012)	**1772**	**0.06**	4221.00	0.27	8103.00	0.18	**4698.67**	**0.17**
GRASP+ILP (Contardo et al., 2014)	**264.38**	**0.14**	1162.93	0.12	2589.53	0.18	**1338.95**	**0.15**
2-Phase HGTS (Escobar et al., 2013)	**105.15**	**0.78**	176.40	0.57	392.33	1.15	**224.63**	**0.83**
MACO (Ting & Chen, 2013)	191.69	0.17	191.43	0.40	202.08	1.24	195.07	0.60
GVTNS (Escobar et al., 2014)	**53**	**0.67**	91.20	0.37	201.22	0.76	**115.14**	**0.60**
HybridGA+ (Lopes et al., 2016)	429.56	0.00	199.05	0.53	363.61	0.71	330.74	0.41
The proposed SAPSO	**151.46**	**0.06**	219.60	0.80	546.38	0.44	**305.81**	**0.43**
	170.95	0.44	219.60	0.80	546.38	0.44	312.31	0.56

[a] Bold numbers indicate that only 13 out of 19 instances are considered.

and compare our results with the best-known CLRP solutions obtained by GRASP x ELS [18], SALRP [19], ALNS-500K and ALNS-5000K [20], 2-Phase HGTS [36], MACO [37], GRASPxILP [38], GVTNS [39], and HybridGA + [40].

1) RESULTS FOR THE CLRP BENCHMARK DATASET

The comparison results between the CLRP solutions obtained by the proposed SAPSO and the best-known LRP solutions are shown in Tables 4, 5, and 6. The first three columns describe the name of the instances and the numbers of customers and depots, respectively. The following three columns present the best-known CLRP solution (BKS) values, the computational time (CPU) of obtaining the best solutions, and the best objective value (*best*Obj) of ten runs. The last column demonstrates the percentage gap between BKS and the best objective value. The smaller the gap, the better the performance of the proposed SAPSO. The proposed SAPSO is tested ten times for each instance.

Table 4 summarizes the comparison results of solving the Barreto [34] dataset. The average CPU time is 170.95s. When comparing the best solution obtained by SAPSO with BKS, the average of the gaps to the best-known results is 0.44%. BKSs to 12 of 19 instances are obtained by SAPSO. Regarding the Prins *et al.* [35] dataset, Table 5 presents the comparison results, where the average computing time is 219.6s. The average of *best* Gap (%) is 0.80%, and the *best* Gap (%) is lower than 1% for 23 (out of 30) instances. Table 6 shows that the average computational time is 546.38s when solving the Tuzun and Burke [14] dataset, and the average gap of the best results is 0.44%. Moreover, 30 of the 36 solutions solved by the proposed SAPSO are close to BKS, and the gaps are all less than 1%. Overall, the proposed SAPSO, with the parameters described in the last section, is tested on 85 LRP benchmark instances. The results show that the *best* Gap (%) is lower than 1% for 82% of instances, and the obtained solution is the best-known solution for 44% of the instances.

Table 7 summarizes the comparison between the proposed SAPSO and the aforementioned state-of-the-art algorithms. CPU refers to the average computational time in seconds, and *AvgBestGap* (%) refers to the average percentage gap between the best obtained result and the best-known result. Note that, in order to compare with some of the methods, there are two values provided for both CPU and *AvgBestGap* (%) for the Barreto [34] dataset. The additional value only considers 13 out of 19 instances in comparison.

Based on Table 7, when compared with the other six methods that only solved 13 of 19 instances in the Barreto [34] dataset, the performance of the proposed SAPSO is one of the best two methods. For the Tuzun and Burke [14] dataset, SAPSO outperforms all the other heuristics, except for ALNS-5000K and GRASP+ ILP. Although computational time varies with machine capabilities, the computational time of the proposed SAPSO is much faster than the other two methods. The average results of the three benchmark datasets are indicated in the last two columns, where *avg*CPU and *avg*Gap are the average of the three values of CPU and *AvgBestGap*, respectively. All things considered, the proposed SAPSO seems suitable to solve the location-routing problem.

2) RESULTS FOR THE LRPDR DATASET

Tables 8 and 9 report the computational results, best objective value (*best* Obj), average objective value (*avg* Obj), and average computational time in seconds (*avg*CPU) of small and original size LRPDR instances, respectively. The proposed algorithms are run 30 times for each instance.

TABLE 8. Computational results for the small LRPDR instances.

Instance	Gurobi				SA					SAPSO				
	BKS	*Obj*	*RPD*	*CPU*	*best*Obj	*Min. RPD*	*avg*Obj	*Mean RPD*	avgCPU	*best*Obj	*Min. RPD*	*avg*Obj	*Mean RPD*	avgCPU
ScoordMin27	1487.77	**1487.77**	0.00	18000.00	**1487.77***	0.00	1487.88	0.01	5.51	**1487.77**	0.00	1505.24	1.17	2.28
ScoordGaspelle1	144.23	**144.23**	0.00	293.30	144.32	0.06	145.32	0.76	9.70	**144.23**	0.00	149.89	3.92	0.32
ScoordGaspelle2	289.06	**289.06**	0.00	286.86	**289.06**	0.00	289.06	0.00	3.22	**289.06**	0.00	289.06	0.00	1.57
ScoordGaspelle3	258.09	**258.09**	0.00	249.56	**258.09**	0.00	258.09	0.00	3.90	**258.09**	0.00	258.09	0.00	0.71
ScoordGaspelle4	324.76	**324.76**	0.00	4733.88	**324.76**	0.00	324.76	0.00	7.38	**324.76**	0.00	324.76	0.00	1.37
ScoordGaspelle5	276.71	**276.71**	0.00	211.12	277.97	0.46	292.19	5.59	11.61	**276.71**	0.00	277.58	0.31	1.40
ScoordGaspelle6	240.03	**240.03**	0.00	207.65	241.21	0.49	241.37	0.56	0.48	**240.03**	0.00	241.41	0.57	0.83
ScoordChrist50	360.28	362.27	0.55	18000.00	363.80	0.98	368.64	2.32	0.84	**360.28**	0.00	372.10	3.28	1.25
Average	422.62	422.86	0.07	5247.79	423.37	0.25	425.91	1.15	5.33	422.61	0.00	427.26	1.16	1.22

* Bold number denotes that the obtained solution is equal to the best-known solution.

TABLE 9. Computational results for the original sized LRPDR instances.

Instance	Gurobi				SA					SAPSO				
	BKS	*Obj*	*RPD*	*CPU*	*best*Obj	*Min. RPD*	*avg*Obj	*Mean RPD*	avgCPU	*best*Obj	*Min. RPD*	*avg*Obj	*Mean RPD*	avgCPU
ScoordMin27	2952.40	2960.01	0.26	18000.00	2990.29	1.28	3337.74	13.05	5.51	**2952.40***	0.00	3251.21	10.12	19.65
ScoordGaspelle1	345.89	**345.89**	0.00	18000.00	**345.89**	0.00	347.36	0.42	9.70	**345.89**	0.00	348.30	0.70	61.66
ScoordGaspelle2	536.21	**536.21**	0.00	6840.50	**536.21**	0.00	538.00	0.33	3.22	**536.21**	0.00	**536.21**	0.00	30.65
ScoordGaspelle3	437.25	437.74	0.11	18000.00	438.41	0.27	453.46	3.71	3.90	**437.25**	0.00	449.44	2.79	70.54
ScoordGaspelle4	428.43	475.21	10.92	18000.00	451.87	5.47	457.18	6.71	7.38	**428.43**	0.00	447.76	4.51	91.42
ScoordGaspelle5	453.23	**453.23**	0.00	18000.00	462.43	2.03	463.41	2.25	11.61	**453.23**	0.00	462.43	2.03	157.48
ScoordGaspelle6	438.18	438.18	0.00	18000.00	458.07	4.54	463.13	5.69	0.48	**453.99**	3.61	466.19	6.39	158.96
ScoordChrist50	557.07	642.21	15.28	18000.00	563.57	1.17	583.20	4.69	0.84	**557.07**	0.00	586.41	5.27	240.86
Average	770.56	786.08	3.32	16605.06	780.84	1.84	830.43	4.61	5.33	770.56	0.45	818.49	3.98	103.90

* Bold number denotes that the obtained solution is equal to the best-known solution.

In order to assess the performance of the proposed SAPSO heuristic in solving LRPDR, the relative percentage deviation (*RPD*) value for each benchmark instance is computed as $RDP = (Obj_{a\lg} - OBJ_{BKS})/OBJ_{BKS} \times 100\%$, where $Obj_{a\lg}$ is the objective function value (OBJ_{alg}) of the solution obtained using a given algorithm or Gurobi, and OBJ_{BKS} is the best *OBJ* among the solutions obtained using SAPSO, SA, and Gurobi. The best and average of the solutions to each test problem, based on 30 runs, obtained using SAPSO, SA and Gurobi, were used to compute the *RPD* values, which are denoted as *Min.RPD* and *MeanRPD*.

Table 8 presents the computational results of solving the small LRPDR instances adapted from the Barreto [34] dataset. It can be seen that six of eight instances are optimally solved by Gurobi within five hours. Gurobi was able to find feasible solutions to the other two instances in five hours. The proposed SAPSO obtained all of the optimal solutions provided by Gurobi within 1.6 seconds, and solutions to the other two instances with the same or better objective value than Gurobi in 2.5 seconds. Furthermore, the average *Min. RPD* and average *Mean RPD* of SAPSO are 0.00% and 1.15%, respectively, while the average *RPD* of Gurobi is 0.07%. On the other hand, SA can only obtain three optimal solutions. Table 8 also presents that the average *Min. RPD* and average *Mean RPD* of SA are 0.25% and 1.15%, respectively. Furthermore, SAPSO's computational time is less than SA.

The computational results for the original sized LRPDR instances adapted from the Barreto [34] dataset are presented in Table 9. Gurobi found a feasible solution to each of the eight instances in five hours, including one optimal solution. The proposed SAPSO not only found the optimal solution obtained by Gurobi, but also found better solutions to the other instances than those found by Gurobi, except for the coordGaspelle6 instance. Furthermore, the average *Min. RPD* and average *Mean RPD* of SAPSO are 0.45% and 3.98%, respectively, while the average *RPD* of Gurobi is 3.32%. On the other hand, SA obtained only three optimal solutions. Table 9 also presents that the average *Min. RPD* and average *Mean RPD* of SA are 1.84% and 4.61%, respectively. Furthermore, SAPSO's computational time is less than SA.

To determine whether the proposed SAPSO heuristic outperformed SA and Gurobi, Wilcoxon signed rank tests in terms of *Min. RPD* and *MeanRPD* were conducted. The analytical results, as shown in Table 10, reveal that, at the confidence level of $\alpha = 0.05$, the proposed SAPSO heuristic significantly outperformed SA and Gurobi in terms of *Min.*

TABLE 10. Results of Wilcoxon signed rank tests on min. RPD and mean RPD.

	SAPSO vs.	SA	Gurobi solver
Dataset 1	Test on Min. *RPD*		
	W	48	36
	P-value	0.01624*	0.1908
	Test on Mean *RPD*		
	W	30.5	14.5
	P-value	0.5854	0.9851
Dataset 2	Test on Min. *RPD*		
	W	51	44
	P-value	0.01614*	0.071
	Test on Mean *RPD*		
	W	35	20
	P-value	0.3992	0.9087
All datasets	Test on Min. *RPD*		
	W	197	159.5
	P-value	0.001154*	0.04314*
	Test on Mean *RPD*		
	W	133	67
	P-value	0.4323	0.9928

*denotes that significant difference exists.

RPD. These statistical results indicate that SAPOS significantly improved the performance of SA in solving LRPDR.

VII. CONCLUSION AND FUTURE RESEARCH

The location-routing problem with demand range is presented in this study. The problem extends LRP by considering a range of potential demands. This study developed a mathematical model for this new LRP variant, and proposed a hybrid metaheuristic, SAPSO, which hybridizes simulated annealing and particle swarm optimization algorithms to solve LRPDR.

In order to ensure the proposed approach is effective in solving LRP-related problems, we test it on three LRP benchmark datasets. The numerical study results show that the percentage gap between BKS and the best objective value obtained by the proposed SAPSO are all less than 1% for the three LRP benchmark datasets. For most of the instances, SAPSO's outperforms SA and Gurobi, thus, further experiments for solving LRPDR were conducted using SAPSO.

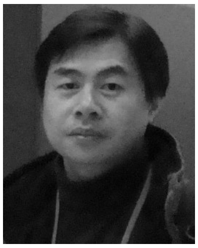
This study applied the proposed SAPSO with specific parameter setting to solve the new dataset of LRPDR, and the results show that the proposed SAPSO outperforms Gurobi both in solution quality and computational time, except for one instance. On the other hand, all things considered, the proposed SAPSO is better than SA. According to the comparison between LRP and LRPDR, this research found that adding the flexibility of delivery quantity could reduce the total cost.

For future studies, other practical considerations, such as multi-period, time windows, inter-depot routes, and split delivery, could be integrated into LRPDR to make it closer to reality. In addition, exact methods and different metaheuristics could be developed for SAPSO to effectively solve the problem.

REFERENCES

[1] O. Dukkanci, B. Y. Kara, and T. Bektaş, "The green location-routing problem," *Comput. Oper. Res.*, vol. 105, pp. 187–202, May 2019.
[2] C. E. Schmidt, A. L. Silva, M. Darvish, and L. C. Coelho, "The time-dependent location-routing problem," *Transp. Res. E, Logistics Transp. Rev.*, vol. 128, pp. 293–315, Aug. 2019.
[3] C. Prodhon, "A hybrid evolutionary algorithm for the periodic location-routing problem," *Eur. J. Oper. Res.*, vol. 210, no. 2, pp. 204–212, 2011.
[4] V. C. Hemmelmayr, "Sequential and parallel large neighborhood search algorithms for the periodic location routing problem," *Eur. J. Oper. Res.*, vol. 243, no. 1, pp. 52–60, 2015.
[5] Ç. Koç, "A unified-adaptive large neighborhood search metaheuristic for periodic location-routing problems," *Transp. Res. C, Emerg. Technol.*, vol. 68, pp. 265–284, Jul. 2016.
[6] Y. Wang, K. Assogba, Y. Liu, X. Ma, M. Xu, and Y. Wang, "Two-echelon location-routing optimization with time windows based on customer clustering," *Expert Syst. Appl*, vol. 104, pp. 244–260, Aug. 2018.
[7] K. Pichka, A. H. Bajgiran, M. E. H. Petering, J. Jang, and X. Yue, "The two echelon open location routing problem: Mathematical model and hybrid heuristic," *Comput. Ind. Eng.*, vol. 121, pp. 97–112, Jul. 2018.
[8] M. Vidović, B. Ratković, N. Bjelić, and D. Popović, "A two-echelon location-routing model for designing recycling logistics networks with profit: MILP and heuristic approach," *Expert Syst. Appl.*, vol. 51, pp. 34–48, Jun. 2016.
[9] V. F. Yu and S.-W. Lin, "Multi-start simulated annealing heuristic for the location routing problem with simultaneous pickup and delivery," *Appl. Soft Comput.*, vol. 24, no. 1, pp. 284–290, Nov. 2014.
[10] T. Capelle, C. E. Cortés, M. Gendreau, P. A. Rey, and L.-M. Rousseau, "A column generation approach for location-routing problems with pickup and delivery," *Eur. J. Oper. Res.*, vol. 272, no. 1, pp. 121–131, 2019.
[11] A. M. Campbell, "The vehicle routing problem with demand range," *Ann. Oper. Res.*, vol. 144, no. 1, pp. 99–110, 2006.
[12] S. Çetinkaya and C.-Y. Lee, "Stock replenishment and shipment scheduling for vendor-managed inventory systems," *Manage. Sci.*, vol. 46, no. 2, pp. 217–232, 2000.
[13] C. Archetti, E. Fernández, and D. L. Huerta-Muñoz, "The flexible periodic vehicle routing problem," *Comput., Oper. Res.*, vol. 85, pp. 58–70, Sep. 2017.
[14] D. Tuzun and L. I. Burke, "A two-phase tabu search approach to the location routing problem," *Eur. J. Oper. Res.*, vol. 116, no. 1, pp. 87–99, Jul. 1999.
[15] G. Nagy and S. Salhi, "Location-routing: Issues, models and methods," *Eur. J. Oper. Res.*, vol. 177, no. 2, pp. 649–672, 2007.
[16] J.-M. Belenguer, E. Benavent, C. Prins, C. Prodhon, and R. W. Calvo, "A branch-and-cut method for the capacitated location-routing problem," *Comput. Oper. Res.*, vol. 38, no. 6, pp. 931–941, 2011.

[17] C. Prins, C. Prodhon, and R. W. Calvo, "Solving the capacitated location-routing problem by a GRASP complemented by a learning process and a path relinking," *4OR*, vol. 4, no. 3, pp. 221–238, 2006.
[18] C. Duhamel, P. Lacomme, C. Prins, and C. Prodhon, "A GRASP×ELS approach for the capacitated location-routing problem," *Comput. Oper. Res.*, vol. 37, no. 11, pp. 1912–1923, 2010.
[19] V. F. Yu, S.-W. Lin, W. Lee, and C.-J. Ting, "A simulated annealing heuristic for the capacitated location routing problem," *Comput. Ind. Eng.*, vol. 58, no. 2, pp. 288–299, Mar. 2010.
[20] V. C. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic, "An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics," *Comput. Oper. Res.*, vol. 39, no. 12, pp. 3215–3228, 2012.
[21] P. Francis, K. Smilowitz, and M. Tzur, "The period vehicle routing problem with service choice," *Transp. Sci.*, vol. 40, no. 4, pp. 439–454, 2006.
[22] C. Wang, D. Mu, F. Zhao, and J. W. Sutherland, "A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows," *Comput. Ind. Eng.*, vol. 83, pp. 111–122, May 2015.
[23] V. F. Yu, P. Jewpanya, and A. A. N. P. Redi, "Open vehicle routing problem with cross-docking," *Comput. Ind. Eng.*, vol. 94, pp. 6–17, Apr. 2016.
[24] V. F. Yu, A. A. N. P. Redi, Y. A. Hidayat, and O. J. Wibowo, "A simulated annealing heuristic for the hybrid vehicle routing problem," *Appl. Soft Comput.*, vol. 53, pp. 119–132, Apr. 2017.
[25] K. M. Ferreira and T. A. de Queiroz, "Two effective simulated annealing algorithms for the location-routing problem," *Appl. Soft Comput.*, vol. 70, pp. 389–422, Sep. 2018.
[26] V. F. Yu and S.-Y. Lin, "A simulated annealing heuristic for the open location-routing problem," *Comput. Oper. Res.*, vol. 62, pp. 184–196, Oct. 2015.
[27] R. Eberhart and J. Kennedy, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1–59.
[28] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem," *Appl. Soft Comput.*, vol. 30, pp. 484–490, May 2015.
[29] Y. Marinakis, G.-R. Iordanidou, and M. Marinaki, "Particle swarm optimization for the vehicle routing problem with stochastic demands," *Appl. Soft Comput.*, vol. 13, no. 4, pp. 1693–1704, 2013.
[30] F. P. Goksal, I. Karaoglan, and F. Altiparmak, "A hybrid discrete particle swarm optimization for vehicle routing problem with simultaneous pickup and delivery," *Comput. Ind. Eng.*, vol. 65, no. 1, pp. 39–53, 2013.
[31] C.-J. Ting, K.-C. Wu, and H. Chou, "Particle swarm optimization algorithm for the berth allocation problem," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1543–1550, 2014.
[32] V. F. Yu, P. Jewpanya, and V. Kachitvichyanukul, "Particle swarm optimization for the multi-period cross-docking distribution problem with time windows," *Int. J. Prod. Res.*, vol. 54, no. 2, pp. 509–525, 2016.
[33] V. F. Yu, P. Jewpanya, C.-J. Ting, and A. A. N. P. Redi, "Two-level particle swarm optimization for the multi-modal team orienteering problem with time windows," *Appl. Soft Comput.*, vol. 61, pp. 1022–1040, Dec. 2017.
[34] S. S. Barreto, "Analysis and modelling of location-routing problems," Ph.D. dissertation, Univ. Aveiro, Aveiro, Portugal, 2004.
[35] C. Prins, C. Prodhon, and R. Wolfler-Calvo, "Nouveaux algorithmes pour le problème de localisation et routage sous contraintes de capacité," in *Proc. MOSIM*, 2004, pp. 1115–1122.
[36] J. W. Escobar, R. Linfati, and P. Toth, "A two-phase hybrid heuristic algorithm for the capacitated location-routing problem," *Comput. Oper. Res.*, vol. 40, no. 1, pp. 70–79, 2013.
[37] C.-J. Ting and C.-H. Chen, "A multiple ant colony optimization algorithm for the capacitated location routing problem," *Int. J. Prod. Econ.*, vol. 141, no. 1, pp. 34–44, 2013.
[38] C. Contardo, J.-F. Cordeau, and B. Gendron, "A GRASP + ILP-based metaheuristic for the capacitated location-routing problem," *J. Heuristics*, vol. 20, no. 1, pp. 1–38, 2014.
[39] J. W. Escobar, R. Linfati, M. G. Baldoquin, and P. Toth, "A granular variable tabu neighborhood search for the capacitated location-routing problem," *Transp. Res. B, Methodol.*, vol. 67, pp. 344–356, Sep. 2014.
[40] R. B. Lopes, C. Ferreira, and B. S. Santos, "A simple and effective evolutionary algorithm for the capacitated location–routing problem," *Comput. Oper. Res.*, vol. 70, pp. 155–162, Jun. 2016.



VINCENT F. YU received the Ph.D. degree in industrial and operations engineering from the University of Michigan, Ann Arbor. He is currently a Professor and the Chair of Industrial Management with the National Taiwan University of Science and Technology. He has published articles in *Applied Mathematical Modelling*, *Applied Soft Computing*, *Computers and Industrial Engineering*, *Computers and Operations Research*, *European Journal of Operational Research*, *Industrial Marketing Management*, *International Journal of Production Research*, *Journal of Cleaner Production*, *Journal of Intelligent Manufacturing*, and *Omega*. His current research interests include operations research, logistics management, soft computing, and artificial intelligence.

PANCA JODIAWAN received the B.Eng. degree in industrial engineering from Bunda Mulia University, Jakarta, Indonesia, in 2016, and the M.B.A. degree in industrial management from the National Taiwan University of Science and Technology, Taipei, Taiwan, in 2019, where he is currently pursuing the Ph.D. degree in industrial management. From 2016 to 2017, he was a Lecturer Assistant with the Industrial Engineering Department, Bunda Mulia University. He has already published his previous works in several conferences, e.g., the International Congress of Logistics and SCM Systems 2019 and the International Conference on Industrial Engineering and Engineering Management 2019. His research interests include combinatorial optimization in logistics and metaheuristics design.

YI-HSUAN HO received the bachelor's degree in logistic management from the National Kaohsiung First University of Science and Technology (NKFUST), Kaohsiung, Taiwan, and the master's degree in industrial management from the National Taiwan University of Science and Technology (NTUST). She has an Expertise in supply chain management, especially in logistic network design and planning. She applied the knowledge to solve a practical location routing problem when she was the Solution Design Intern of DHL Supply Chain, Taiwan. After receiving her master's degree, she has been working as a Transportation Specialist for the company for about one and half years. She plans to build up her on-the-job experiences and continues her involvement in supply chain industry.

SHIH-WEI LIN received the bachelor's, master's, and Ph.D. degrees in industrial management from the National Taiwan University of Science and Technology, Taiwan, in 1996, 1998, and 2000, respectively. He is currently a Professor with the Department of Information Management, Chang Gung University, Taiwan. He is also with the Department of Neurology, Linkou Chang Gung Memorial Hospital, Taoyuan, Taiwan, and the Department of Industrial Engineering and Management, Ming Chi University of Technology, Taipei, Taiwan. His articles have appeared in *Computers and Operations Research*, the *European Journal of Operational Research*, the *Journal of the Operational Research Society*, the *European Journal of Industrial Engineering*, the *International Journal of Production Research*, *The International Journal of Advanced Manufacturing Technology*, *Knowledge and Information Systems*, *Applied Soft Computing*, *Applied Intelligence*, *Expert Systems with Applications*, and so on. His current research interests include meta-heuristics and data mining.