

Article

A Swarm Intelligence Solution for the Multi-Vehicle Profitable Pickup and Delivery Problem

Abeer I. Alhujaylan ^{1,*} and Manar I. Hosny ² 

¹ Department of Information Technology, College of Computer, Qassim University, Buraydah 51452, Saudi Arabia

² Computer Science Department, College of Computer and Information Sciences, King Saud University, Riyadh 11362, Saudi Arabia; mifawzi@ksu.edu.sa

* Correspondence: a.alhujaylan@qu.edu.sa

Abstract: Delivery apps are experiencing significant growth, requiring efficient algorithms to coordinate transportation and generate profits. One problem that considers the goals of delivery apps is the multi-vehicle profitable pickup and delivery problem (MVPPDP). In this paper, we propose eight new metaheuristics to improve the initial solutions for the MVPPDP based on the well-known swarm intelligence algorithm, Artificial Bee Colony (ABC): K-means-GRASP-ABC(C)S1, K-means-GRASP-ABC(C)S2, Modified K-means-GRASP-ABC(C)S1, Modified K-means-GRASP-ABC(C)S2, ACO-GRASP-ABC(C)S1, ACO-GRASP-ABC(C)S2, ABC(S1), and ABC(S2). All methods achieved superior performance in most instances in terms of processing time. For example, for 250 customers, the average times of the algorithms was 75.9, 72.86, 79.17, 73.85, 76.60, 66.29, 177.07, and 196.09, which were faster than those of the state-of-the-art methods that took 300 s. Moreover, all proposed algorithms performed well on small-size instances in terms of profit by achieving thirteen new best solutions and five equal solutions to the best-known solutions. However, the algorithms slightly lag behind in medium- and large-sized instances due to the greedy randomised strategy and GRASP that have been used in the scout bee phase. Moreover, our algorithms prioritise minimal solutions and iterations for rapid processing time in daily m-commerce apps, while reducing iteration counts and population sizes reduces the likelihood of obtaining good solution quality.

Keywords: multi-vehicle profitable pickup and delivery problem; Artificial Bee Colony; swarm intelligence; metaheuristic algorithms



Citation: Alhujaylan, A.I.; Hosny, M.I. A Swarm Intelligence Solution for the Multi-Vehicle Profitable Pickup and Delivery Problem. *Algorithms* **2024**, *17*, 331. <https://doi.org/10.3390/a17080331>

Academic Editor: Vittorio Maniezzo

Received: 2 July 2024

Revised: 17 July 2024

Accepted: 25 July 2024

Published: 31 July 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Over the past five years, delivery apps have experienced substantial growth, particularly food delivery apps, where COVID-19 propelled the industry forward a few years as millions of individuals in lockdown ordered food and other essential services online. The grocery delivery platform called Instacart reported that its objectives for 2022 were met during the third week of lockdown. In 2022, China emerged as the largest food delivery market, with a staggering market value of USD 42.5 billion. By 2029, the market size of the food delivery app industry as a whole is anticipated to reach USD 165 billion [1]. Moreover, 85% of Americans place online food orders via food ordering applications. Moreover, 80% of restaurants in the United States provide customers with the convenience of mobile ordering [2]. Similarly, food delivery applications in Saudi Arabia are providing instantaneous services and a streamlined user experience for individuals to enjoy food. As a result, new applications that vie to provide services, convenience, and comfort to over 35 million individuals emerge periodically. Examples of these applications that have become very popular in Saudi Arabia are Mrsool, Uber Eats, Talabat, Hunger station, Ninja, Jahez, To You, The Chifz, Shgardi, and NANA [3]. The escalating demand for food delivery applications has concurrently heightened the strain on delivery enterprises, which require

expeditious algorithms to coordinate the transportation of goods in order to generate profits and guarantee client contentment. One of the problems that was presented in the literature that mimics the goal of food delivery apps, increasing profits, achieving user satisfaction, and reducing costs is the multi-vehicle profitable pickup and delivery problem (MVPPDP).

Gansterer et al. [4] introduced the MVPPDP, which is a subtype of the selective pickup and delivery problem (SPDP) and a novel variation of the well-known pickup and delivery problem (PDP). Within the constraints of a one-day planning horizon, the MVPPDP seeks to identify the most profitable routes by minimising travel expenses and maximising revenue collection. The MVPPDP is a static problem characterised by a standardised fleet of vehicles, a central depot, a collection of customers, and a predetermined quantity of requests and products. A customer pair, consisting of the pickup customer and the delivery customer, defines each request. By doing so, the products are conveyed from a pickup customer to a delivery customer in order to generate a revenue stream for the operation. MVPPDP real-world applications include mobile applications for food delivery services and delivery companies.

The MVPPDP is one of the combinatorial optimisation (CO) problem applications that receives particular attention from researchers in computer science and operations research. Solving a CO problem mainly requires searching to find the optimal or best possible solution from a limited set of feasible candidate solutions. Numerous algorithms can be used to solve CO problems, and these algorithms can be classified into two categories: exact algorithms, which are guaranteed to obtain optimal solutions in an adequate amount of time, and approximate algorithms, usually called heuristics or metaheuristics, which are not guaranteed to deliver an optimal solution but provide a good solution in a reasonable amount of time. The words heuristic and meta belong to the Greek language, where heuristic is from *heuriskein*, meaning *‘to find’* or *‘to discover’* new methods and strategies to solve problems, while the prefix meta means *‘beyond’*, in an upper level. Metaheuristic methods can be classified based on the number of solutions used at the same time into two categories: single solution-based metaheuristics (S-metaheuristics) and population-based metaheuristics (P-metaheuristics). In S-metaheuristics, the focus is on modifying and improving a single candidate solution. On the other hand, P-metaheuristics preserve and improve several candidate solutions instead of just a single solution. They are based on iteratively improving solutions in a population and evolving the whole population until a set of good solutions is reached. Exploration-oriented features of P-metaheuristics have made them better than S-metaheuristics at diversifying the search space. The most famous examples of P-metaheuristics are algorithms that are inspired by natural behaviour such as genetic algorithms and swarm intelligence (SI) [5].

Generally, there are two phases that should be applied to obtain a good problem solution: solution construction and solution improvement. Solution construction involves creating one or more initial possible solutions that can be considered as a starting point for exploring the search space. Solution improvement is based on modifying the starting solutions to improve them, according to predefined strategies, until a high-quality solution is reached or stopping criteria have been met (e.g., a maximum number of iterations, or a specific amount of time) [5].

Due to the fact that the MVPPDP is an NP-hard problem (as it is a subtype of the SPDP, which is a special case of PDP) [4], metaheuristic algorithms can be employed to efficiently locate optimal solutions in a reasonable amount of time, particularly for problems of moderate to large sizes. In this paper, we introduce novel algorithms for tackling this problem, with the objective of enhancing the initial solution proposed in [6,7]. Our approach is based on the utilisation of the Artificial Bee Colony (ABC) algorithm, which is one of the widely recognised artificial intelligence algorithms known as SI. Diverse strategies were implemented with ABC, each demonstrating efficacy in accelerating the search procedure and reducing processing time.

The primary goals of this paper are (1) assisting food delivery applications and delivery companies in optimising their services through accelerated customer selection and

route planning in order to maximise profits as well as customer satisfaction; (2) efficient transport planning that can help conserve energy resources by reducing the environmentally damaging effects of gas emissions; and that (3) for some tested instances, the proposed approaches achieve results that surpass the state-of-the-art methods documented in the literature for the MVPPDP.

The rest of this paper is organised as follows: a review of some related work is presented in Section 2. Section 3 introduces the formal problem definition. The proposed method is described in detail in Section 4. Section 5 illustrates the experimental results. Finally, conclusions and future work are presented in Section 6.

2. Related Studies

The MVPPDP is classified as a type of SPDP and its objective is to maximise profits while minimising travel expenses by visiting only profitable customer pairs. Achieving a favourable route that maximises the difference between the total revenue earned and the total travelling cost is the objective of both the MVPPDP and the profitable tour problem (PTP). The primary distinction is that in the PTP, the vehicle route is not subject to any constraints, including but not limited to the maximum trip time, vehicle capacity limits, or precedence constraints [8]. Research focusing on the PTP is in short supply in the scientific literature [9]. Hence, we commence by presenting research pertaining to the MVPPDP.

Gansterer et al. [4] introduced the initial four algorithms. The variable neighbourhood search (VNS) metaheuristic was implemented in [10]. The greedy construction heuristic (C1) and the two-stage cheapest insertion heuristic (C2) were employed during the construction phase of the VNS. During the improvement phase, the subsequent neighbourhood operators were implemented: the pairwise backward exchange, the pairwise forward exchange, forward insertion, relocate pairs, backward insertion, inter-tour exchange, inter-tour insert, inter-dummy insert, inter-dummy exchange and the gravity centre exchange. The authors implemented two approaches for applying the neighbourhood operators: a sequential (Seq) method, which employs a predetermined sequence of neighbourhoods, and a self-adaptive (Sea) method, which adjusts itself based on the search status to determine the sequence of neighbourhoods. In order to compare the proposed method with an alternative algorithm, a guided local search (GLS) metaheuristic-based algorithm was ultimately implemented. An evaluation of the proposed algorithm was conducted on 36 randomised instances of varying sizes. In terms of solution quality, the experimental results demonstrated that both variants of the VNS outperformed the GLS.

Haddad [11] introduced a new algorithm to solve the MVPPDP. Pending a solution to the profitable pickup and delivery problem, they proposed an algorithm known as IPPD, which was derived from a combination of an iterated local search and random variable neighbourhood descent. During the search, the IPPD algorithm was not restricted to accepting only viable solutions. For the construction of the initial solution, the C1 heuristic utilised by Gansterer et al. [4] was implemented. To enhance the solution, several neighbourhood moves were implemented: pair swap or pair shift, block swap or block shift, pickup or delivery shift, inter-block swap or inter-block shift, gravity centre exchange, insert operator and remove operator, and inter-pair swap or inter-pair shift. On the benchmark instances suggested by Gansterer et al. [4], the proposed algorithm was evaluated. It demonstrated its effectiveness in handling small- and medium-sized cases by identifying new optimal solutions for six cases. However, for large instances, the performance of the proposed algorithm was inadequate.

In Alhujaylan and Hosny [6], the construction phase of the widely recognised greedy randomised adaptive search procedure (GRASP) was employed to develop preliminary solutions for the MVPPDP. They used the same dataset that was created by [4]. They conducted a performance comparison between the method and two construction heuristics that had been previously documented in the literature for constructing initial solutions to the MVPPDP. The experimental results demonstrated that the proposed method outperformed

alternative construction heuristics, particularly in instances of a small size, where eight new best initial solutions to the problem were obtained.

In [7], Alhujaylan and Hosny presented new heuristics to construct initial solutions for the MVPPDP. The heuristics are based on first clustering the search space of the MVPPDP using three clustering methods: K-means, adaptive K-means, and ant colony optimisation (ACO). Then, a modified version of the GRASP has been used to construct the initial MVPPDP solution based on the results of each clustering algorithm. The proposed solution was applied in the dataset that was presented by the researchers of [4]. They compared their results with those from the other construction heuristics that have been previously used for the MVPPDP. The experimental results proved the effectiveness of the proposed algorithms in terms of both solution quality and processing time.

In [12], the authors incorporated time windows with uncertain travel times into the MVPPDP. The researchers utilised a subset of the database created by [4], which comprised twenty and fifty requests, corresponding to two and three vehicles, respectively. The objective of the client is to identify a solution that optimises the net profit, which is calculated as the difference between the revenue collected, the cost of the route, and the expense incurred due to time window violations. They took wide time windows and standard time windows into consideration. Under the assumption that travel times are independent Gamma-distributed random variables and with a risk-neutral perspective, they demonstrated that the anticipated costs of punctuality and early arrival can be calculated in a closed form. The authors suggested employing a graph search approach as the solution method. Experiments have demonstrated that precise solutions can be obtained in a reasonable amount of time.

3. Problem Definition

The MVPPDP is a predefined static problem having several constituents. It is postulated that a centralised depot is responsible for receiving customer requests. These requests comprise pickup and delivery customers in pairs. To fulfil these demands, we operate a fleet of identical vehicles that convey a variety of identical products from a subset of pickup customers to their respective delivery customers, thereby generating a specific profit. The following limitations of the MVPPDP must be considered:

- The pairing constraint stipulates that a predefined customer pair is assigned to each request (pickup and delivery).
- Precedence constraint: Priority must be given to visiting the pickup customer over the delivery customer.
- The time limit for journeys: Every vehicle is subject to a daily travel time restriction that must not be surpassed during the service of customers.

One limitation is the capacity of each vehicle, which must not be surpassed during product collection from pickup customers.

- Every vehicle should depart from and arrive at the depot with an empty load.
- A single visit is permitted per customer.

The MVPPDP can be formally defined as follows [10]: Let $G = (V, A)$ be a graph in which $V = \{0, \dots, 2n + 1\}$ defines the vertex set, where the vertex $(0, 2n + 1)$ corresponds with a central depot, while the remaining vertices represent the pickup customers $P = \{1, \dots, n\}$, and delivery customers $D = \{n + 1, \dots, 2n\}$. $A = \{(i, j) : i, j \in V, i \neq j\}$ defines the arc set, where each arc is associated with a non-negative routing cost c_{ij} . Each delivery vertex i has a revenue r_i to be gained when visiting it. Also, each vertex i has a supply q_i (pickup, $q_i > 0$) or demand (delivery, $q_{n+i} = -q_i$). At the start, there is no supply or demand in the depot ($q_0 = 0$). In addition, each vertex i has service duration d_i . There is a set of vehicles $K = \{1, \dots, m\}$, and each vehicle has a load capacity C and maximum tour time T .

The objective function (OF) of the MVPPDP aims to maximise the total profit by subtracting the total travel costs from the total revenues. It can be formulated as follows:

$$\text{Maximize } \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} (r_i - c_{ij}) x_{ijk}, \quad (1)$$

The mathematical model of the MVPPDP can be formulated as in [4,6] as follows:

$$\sum_{i \in V} \sum_{k \in K} x_{ijk} \leq 1 \quad j \in V \quad (2)$$

$$\sum_{j \in V} \sum_{k \in K} x_{ijk} \leq 1 \quad i \in V \quad (3)$$

$$x_{i0k} = 0 \quad i \in V, k \in K \quad (4)$$

$$x_{2n+1,jk} = 0 \quad j \in V, k \in K \quad (5)$$

$$\sum_{i \in V} (x_{ijk} - x_{jik}) = 0 \quad j \in V \setminus \{0, 2n+1\}, k \in K \quad (6)$$

$$\sum_{j \in V} (x_{ijk} - x_{n+i,jk}) = 0 \quad i \in P, k \in K \quad (7)$$

$$\sum_{j \in V} x_{0jk} = \sum_{i \in V} x_{i,2n+1,k} = 1 \quad k \in K \quad (8)$$

$$(x_{ijk} = 1) \Rightarrow Q_{jk} = Q_{ik} + q_j \quad i \in V, j \in V \setminus \{0\}, k \in K \quad (9)$$

$$Q_{ik} \leq C \quad i \in V, k \in K \quad (10)$$

$$Q_{0k} = 0 \quad k \in K \quad (11)$$

$$B_{ik} \leq B_{n+i,k} \quad i \in P \quad (12)$$

$$B_{jk} \geq X_{ijk} (B_{ik} + d_i + t_{ij}) \quad i \in V, j \in V, k \in K \quad (13)$$

$$B_{2n+1,k} \leq T \quad k \in K \quad (14)$$

$$B_{0k} = 0 \quad k \in K \quad (15)$$

$$x_{ijk} \in 0, 1 \quad i \in V, j \in V, k \in K \quad (16)$$

$$Q_{ik}, B_{ik} \geq 0 \quad i \in V, k \in K \quad (17)$$

Constraints (2) and (3) require that each vertex be visited at most once. Constraints (4) and (5) prevent any vehicle from entering the origin depot (beginning point) or leaving the destination depot (end point). Constraint (6) indicates flow conservation. Constraint (7) states that each client pair (pickup and delivery) must be served by the same vehicle. Constraint (8) requires that all vehicles start and return to the depot. Constraint (9) states that exceeding the vehicle capacity is not permitted. Constraint (10) specifies that the load of a vehicle is limited by C . Constraint (11) specifies that all vehicles start with an empty cargo. Constraint (12) indicates that the delivery customer cannot be visited before the pickup customer. Constraint (13) states that the earliest time to begin service at a vertex j is determined by adding the beginning of the service time at the vertex i to the service time at and the travel time between i and j . Constraint (14) indicates that the maximum time for each vehicle is limited by T . Constraint (15) specifies that each vehicle begins at the depot at time zero. Finally, decision variables are defined by (16) and (17).

4. Proposed Method

In the current study, we employed the Artificial Bee Colony (ABC) algorithm that was first proposed by Karaboga [13]. It mimics the collaborative and collective foraging behaviour of a bee colony. In the ABC algorithm, there are three different types of bees:

scout bees, employed bees, and onlooker bees. Each type has a different role in the exploration and exploitation of food sources. A food source represents the solution to a problem, and better solutions belong to food sources with more nectar. The ABC algorithm works as follows: Initially, the scout bees are assigned randomly to the search space to find initial food sources. Then, the employed bees are sent out to exploit the places discovered by scout bees, with each employed bee assigned to one food source. After that, the employed bees start a neighbourhood search to find a better food source nearby. After finding a better food source, the employed bees leave all previous food sources and exploit only the better ones. These employed bees then return to the hive and share their information about discovered food sources with onlooker bees through what is called the waggle dance. The decision of whether the onlooker bees will follow the employed bees depends on the richness of the food sources. If the onlooker bees do decide to follow, they then become employed bees and repeat the procedure of the employed bees. After several iterations, the food source might be exhausted; at that point, the employed bees become scout bees and start searching randomly to find a new food source to replace the previous one. The flowchart of the ABC algorithm is illustrated in Figure 1 [14].

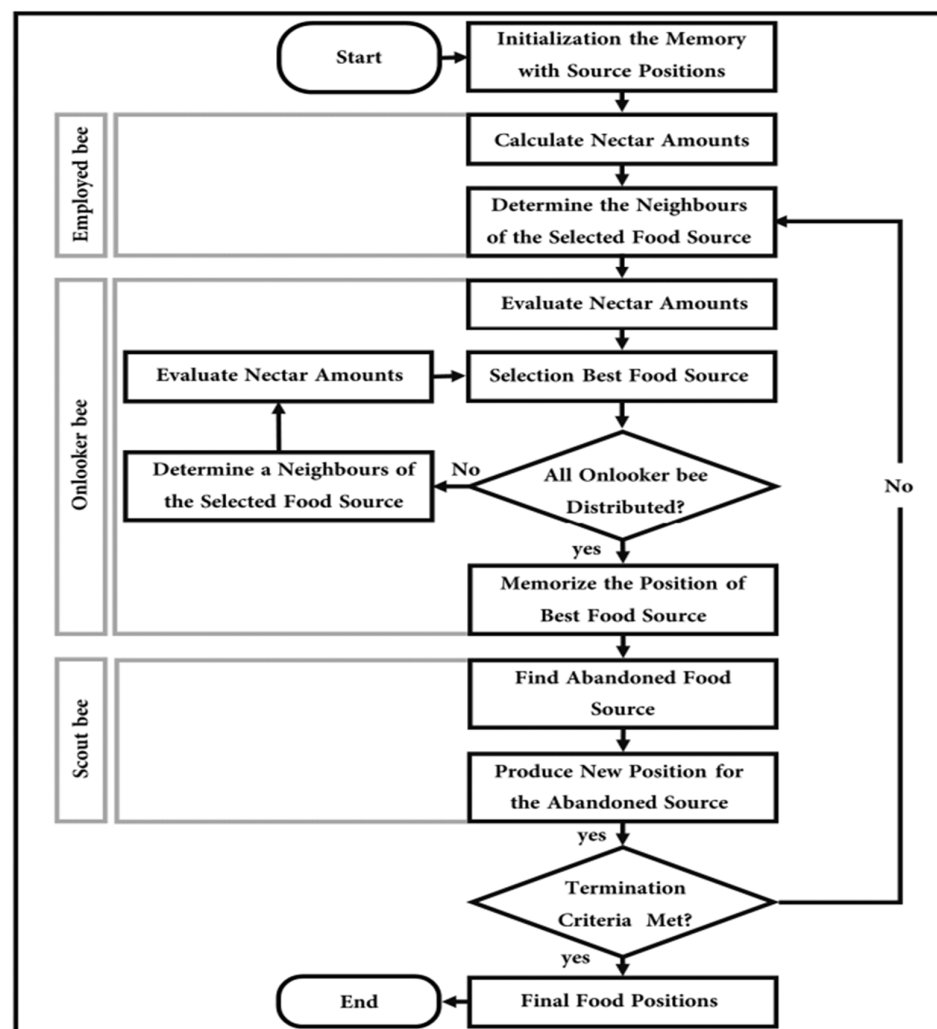


Figure 1. Flowchart of the Artificial Bee Colony algorithm [14].

The ABC heuristic has been used for solving several VRP variants, such as being used by Szeto et al. [15] to solve the capacitated vehicle routing problem, by Wu et al. [16] to solve the VRP with time windows, and by Yao et al. [17] to solve the periodic VRP.

In this paper, we aim to improve the initial solutions proposed in [7]. There are many metaheuristics that have been used to enhance the quality of solutions, such as local search (LS), simulated annealing (SA), and VNS. In our study, we also use a metaheuristic that combines greedy and randomised searches and is based on honeybee behaviour, namely the ABC. The main reason for selecting the ABC rather than other honeybee algorithms is its popularity, with the number of studies that have used the ABC compared to other variants reaching 54%, as reported in Karaboga's survey [18]. For tackling the MVPPDP, two versions of the ABC are proposed: ABC without clustering (ABC) and ABC with clustering (ABC(C)), as will be explained next.

4.1. ABC without Clustering (ABC)

In this part of the research, we aimed to improve the initial solutions that have been achieved using the improved GRASP in [7]. The ABC metaheuristic consists of three phases: the employed bee phase, the onlooker bee phase, and the scout bee phase. The main goal of the first two phases is to increase the intensification of the search space to obtain good solutions, while the last phase aims to increase the diversification and prevent getting stuck in local optima. The details of our proposed algorithm are illustrated below.

1. Employed bee phase:

After assigning the initial solutions to the employed bees, they start a neighbourhood search to find better solutions nearby by randomly selecting one of a set of neighbourhood operators. The neighbourhood operators aim to make a small change to the solution to improve it. Thus, for each solution, one route is selected randomly to apply one operator that is also selected randomly. The set of neighbourhood operators is described below:

- Relocate a pickup customer only: Choose one pickup customer randomly and relocate its position, such that the new pickup position must not come after its delivery customer.
- Relocate a delivery customer only: Choose one delivery customer randomly and relocate its position, such that the new delivery position must not come before its pickup customer.
- Relocate a customer pair: Choose a customer pair randomly and relocate their positions, preserving the precedence constraint (i.e., pickup before delivery).
- Swap two customer pairs: Select two customer pairs randomly and swap their positions.
- Two-Opt operator: Select two consecutive pairs of customers (a consecutive pair is one in which a pickup customer is followed directly by its delivery customer) and reorder the edges between pairs while considering the precedence constraint.
- Remove one customer pair randomly and insert an unvisited one: Remove a customer pair randomly, select an unvisited customer pair that has a high insertion ratio (IR) using Equation (18), and insert it into the best position in the route.

$$IR_{customerpair} = \frac{r_{customerpair}}{c_{(depot,PC)}}, \quad (18)$$

where r is the revenue of the customer pair and $c_{(depot,PC)}$ is the distance between the depot and the pickup customer of that pair.

- Remove a customer pair with a low IR and insert an unvisited pair with a high IR: Compute the IR for all visited pairs using Equation (18), and remove a customer pair with the lowest IR value. Then, recompute the IR for all unvisited customer pairs, select one with the highest IR value, and insert it into the best position in the route.
- Insert a customer pair with a high IR: Compute the IR for all unvisited customer pairs, select one with the highest IR value, and insert it into the best position in the route.

After applying the selected neighbourhood operator, if the new solution S' is feasible, the fitness of S' is computed using Equation (1) and compared with the fitness of the old solution S . To escape from local optima, we decided to use the demon algorithm (DA) rather

than a simple local search algorithm, since the DA has the ability to accept a feasible non-improved new solution. The DA is a variant of the SA that was presented by Creutz [19]. The acceptance function of the DA is based on the value of the demon (credit) that is initialised at the beginning with a given value D . In the case where the new solution S' is not an improvement, the difference (ΔE) between the fitness of S and S' is compared with the demon value, and S' is accepted if this difference is less than or equal to D . Then, if S' is accepted, it is memorised, and the value of D is decreased; otherwise, the old solution S is memorised, the D value is increased, and a non-improvement counter for this solution is increased by one. The DA is simpler and faster than the SA because the latter requires more computational time for generating a random number and an exponential function. Also, it contains only one parameter that needs to be tuned. The DA proved to be an effective algorithm when it was used to solve several real-life problems, achieving good results in terms of the quality of solution and time, when compared with the SA [19].

2. Onlooker bee phase:

Only the solutions that have good OF values have a high probability of being assigned to onlooker agents. Several strategies have been proposed for selecting solutions, such as roulette wheel selection, tournament selection, stochastic universal sampling, and rank-based selection [20]. We adopted the most popular one, which is the roulette wheel selection strategy, in which the probability of selection P_i is computed as follows:

$$P_i = \frac{OF_i}{\sum_{j=1}^n OF_j}, \quad (19)$$

where OF_i is the objective function for a solution i , and $\sum_{j=1}^n OF_j$ is the sum of objective function values for all solutions in a population. After a solution is selected, one of the solution routes is selected randomly and all the neighbourhood operators used in the employed bee phase are applied sequentially to it, and then the fitness of the new solution is measured using Equation (1). To increase the intensification process and focus only on promising areas of the search space, the acceptance criterion is based on a straightforward LS (local search) algorithm. That is, if S' is better than S , memorise S' , and there is no need to complete the remaining operators; otherwise, try the next operator. If the old solution is not improved by any operator, the old solution is memorised, and a non-improvement counter for this solution is increased by one.

3. Scout bee phase:

The scout bee phase aims to increase the diversification and expand the solution space, thus preventing getting stuck in local optima. The scout bee phase starts when a non-improvement counter for a solution S_i reaches a predefined number of trials, called a limit. The limit refers to the number of attempts that were made to improve the solution S_i , in both employed and onlooker bee phases, respectively. Three important actions should be performed when a non-improvement counter for solution S reaches the limit: (1) clear the non-improvement counter, (2) clear the solution S_i and (3) reconstruct the solution S_i . A suitable reconstruction strategy has an important role in the scout bee phase, where it either contributes to enhancing the quality of the solution or leaves the solution unchanged. Thus, we proposed two strategies to reconstruct the solution S_i . The details of these reconstruction heuristics are presented as follows:

- First strategy (ABC(S1)): One of the following three heuristics is selected randomly:

1. Random insertion heuristic:

While the time constraint is not violated, the customer pairs are selected randomly and inserted directly into the route. Each pickup customer is followed by its delivery customer; therefore, there is no need to check the vehicle's capacity and precedence constraints.

2. Greedy and randomised heuristics:

While the time constraint is not violated, the customer pairs are selected randomly and inserted into the best positions in the route. In addition to the time constraint, the vehicle capacity and precedence constraints must not be violated.

3. Greedy insertion heuristic:

The insertion ratio is computed for all customer pairs using Equation (18). The customer pair that has the highest IR value is then inserted into the route in the best position. Time, vehicle capacity, and precedence constraints must not be violated.

- Second strategy (ABC(S2)): This strategy works exactly like the GRASP in [7].

After the solution S_i is reconstructed, the OF is computed using Equation (1), and S_i is memorised as a new solution and used to start again with the improvement process using the employed and onlooker bee phases.

The outline of the routing phase of our ABC is presented in Algorithm 1, where each notation is defined as follows: Pop Size represents the population size; Max Iter: the maximum number of iterations; Num Vehicles: the quantity of vehicles; US: pairs of customers who have not been served; ΔE : the difference between OF values for an old and new solution; D : demon value; and *Non-improvement*: a counter to count the number of unsuccessful attempts to improve the solution.

Algorithm 1: The pseudocode of the proposed ABC.

```

Pop = { $S_1, S_2, S_3, \dots, S_n$ }; /* Initial solutions that have been generated using GRASP(V2) */
Limit = initial limit value,  $D$  = initial demon value, Non-improvement = 0, Global_Best = Best
solution in Pop;
For  $M = 1$  to  $Max\_Iter$ 
  /** Employed bee phase **/
  For  $i = 1$  to Pop_Size
    Select randomly one route of solution  $S_i$ ;
    Apply one randomly selected neighbourhood operator to the current route of solution
     $S_i$  to obtain a new solution  $S'_i$ ;
    If  $S'_i$  is feasible Then Compute the OF of  $S'_i$ ;
    End If
    If ( $OF(S'_i) > OF(S_i)$ )
      Memorise the new solution  $S'_i$ ;
    Else if ( $\Delta E \leq D_i$ )
      Memorise the new solution  $S'_i$ ;
      update the demon value:  $D_i = D_i - \Delta E$ ;
    Else
      Memorise the old solution  $S_i$ ;
      update the demon value:  $D_i = D_i + \Delta E$ ;
      Non-improvement( $i$ ) = Non-improvement( $i$ ) + 1;
    End If
  End For
  /** Onlooker bee phase **/
  Select  $N$  solutions from the employed bees using roulette wheel selection strategy;
  For  $i = 1$  to  $N$ 
    Select randomly one route of solution  $S_i$ ;
    For  $j = 1$  to number of neighbourhood operators
      Apply a selected neighbourhood operator ( $j$ ) to the current route of solution
       $S_i$  to obtain a new solution  $S'_i$ ;
      If  $S'_i$  is feasible Then Compute the OF of  $S'_i$ ;
      End If
      If ( $OF(S'_i) > OF(S_i)$ )
        Memorise the new solution  $S'_i$ ;
        Break;
      End if
    End For
  End For

```

```

If the solution  $S_i$  has not improved after applying the neighbourhood operators
Non-improvement( $i$ ) = Non-improvement( $i$ ) + 1;
End If
End For
/*** Scout bee phase ***/
For  $i = 1$  to  $N$ 
If Non-improvement( $i$ ) > Limit
Non-improvement( $i$ ) = 0;
 $S_i = 0$ ;
Reconstruct  $S_i$  by using either the S1 or S2 strategy of scout bee phase; Compute the OF for  $S_i$ ;
End if
End For
Assign the solution that has maximum OF to the Best_Solution;
If (Global_Best < Best_Solution)
Global_Best = Best_Solution;
End If
End For
Output: the Global_Best solution

```

4.2. ABC with Clustering (ABC(C))

In this part of the study, the ABC method described in Section 4.1 was used to improve the initial solutions that were constructed using the various GRASP(C) methods reported in [7], namely K-means-GRASP(C), modified-K-means GRASP(C), and ACO-GRASP(C). Again, there are some modifications that must be considered when using the ABC with those algorithms. When using the following operators, the insertion process should be restricted to the customer pairs inside the same cluster only: remove one customer pair randomly and insert an unvisited one, remove a customer pair with a low IR and insert an unvisited pair with a high IR, and insert a customer pair with a high IR. Also, the insertion restriction should be applied in the scout bee phase in both strategies (S1 and S2). We used the same parameter values that have been used in the ABC (without clustering) to improve the initial solutions in this part of the study. Moreover, the same strategies from the scout bee phase (S1 and S2) have been used in the ABC(C) algorithms. Therefore, overall, we have six algorithms that use the ABC(C): (1) K-means-GRASP(C)-ABC(C)S1, (2) K-means-GRASP(C)-ABC(C)S2, (3) modified K-means-GRASP(C)-ABC(C)S1, (4) modified K-means-GRASP(C)-ABC(C)S2, (5) ACO-GRASP(C)-ABC(C)S1, and 6) ACO-GRASP(C)-ABC(C)S2. For example, the first algorithm refers to generating the initial solution using the GRASP after clustering the search space by using the K-means algorithm and then improving it using the ABC with reconstruction strategy S1.

5. Computational Experiments

All algorithms have been implemented in MATLAB (R2017b) on a laptop powered by an Intel Core i7-4510U processor operating at 2.00 GHz (2601 MHz, two cores, four logical processors). The following subsections provide descriptions of the datasets utilised and the specifics of parameter tuning prior to presenting the experimental results.

5.1. Test Instances

To test the performance of our proposed methods, we used the same instances used by [4]. The dataset comprises 36 instances, which have been categorised into three distinct groups: small size (consisting of 20 and 50 customers, respectively, served by two and three vehicles), medium size (comprising 100 and 250 customers, served by 4 and 5 vehicles, respectively), and large size (500 and 1000 customers served by 6 and 8 vehicles, respectively). Moreover, twelve instances comprise each group. These occurrences exhibit variability in terms of both time and revenue. To generate short and long routes, the total time limit is either small or large, with a range of 2500 to 15,000 s.

Furthermore, the revenue amounts are established in one of three ways: at random, proportional to demand, or equal for all customers. Revenue is acquired from the delivery customer subsequent to the shipment of merchandise originating from the pickup customer. The quantity of goods is specified as a positive integer between 1 and 50.

5.2. Parameter Tuning

5.2.1. Parameter Tuning of ABC without Clustering

The ABC algorithm has three parameters: the number of iterations (Max_Iter), the size of the population (Pop_Size), and the initial value of the limit. Additionally, since we used the DA in the employed bee phase, the initial value of the demon (D) needs to be tuned. The details of testing each parameter are presented below.

Number of Iterations

Each data instance of 100 customers was tested with different numbers of iterations: 10, 50, 100, 500, and 750. As seen in Table 1, the results of testing showed that there was no enhancement in the objective function values after 500 iterations. Thus, the maximum number of iterations was taken to be 500.

Table 1. Results of tuning the number of iterations of the ABC.

Dataset Instances	Number of Iterations				
	10	50	100	500	750
13-100-F-S	32,274.04	35,432.98	39,437.98	39,438.58	39,438.58
14-100-F-L	63,311.86	63,311.86	63,311.86	63,455.13	63,455.13
15-100-P-S	68,625.48	68,625.48	71,185.28	72,032.22	72,032.22
16-100-P-L	115,792.8	115,792.8	124,608.2	131,404.9	131,404.9
17-100-R-S	67,974.56	74,246.27	74,246.27	74,330.95	74,330.95
18-100-R-L	112,986.9	112,986.9	116,775.2	119,991.9	119,991.9

Size of Population

The same dataset instances were tested again to select the appropriate number of solutions in the population. At the beginning, the population size was set to 30, because it provided the best results given in [7]. Then, we increased the population to different sizes: 40, 50, and 60 solutions. Table 2 presents the values of the OF after setting the number of iterations to 500. The results showed that increasing the population size to more than 50 did not lead to an improvement in the objective function value. Thus, the population size was taken to be 50.

Table 2. Results of tuning the population size of the ABC.

Dataset Instances	Population Size			
	30	40	50	60
13-100-F-S	39,438.58	39,438.58	41,472.86	41,472.86
14-100-F-L	63,455.13	66,453.66	66,453.66	66,453.66
15-100-P-S	72,032.22	72,032.22	72,170.54	72,170.54
16-100-P-L	131,404.9	131,404.9	131,404.9	131,404.9
17-100-R-S	74,330.95	77,484.06	77,484.06	77,484.06
18-100-R-L	119,991.9	120,249.7	126,504.3	126,504.3

Limit Value

The limit value of the non-improvement counter is another important parameter of the ABC algorithm because it affects the conversion of the search process. Thus, the value selected for the limit should be suitable for the number of iterations. Therefore, we tested the limit using two values: 100 and 200. The results in Table 3 show that increasing the

limit value to more than 100 did not contribute to an improvement in the objective function value. Thus, the limit value was taken to be 100.

Table 3. Results of tuning the limit value of the ABC.

Dataset Instances	Limit	
	100	200
13-100-F-S	41,472.86	41,472.86
14-100-F-L	66,453.66	66,453.66
15-100-P-S	72,170.54	72,170.54
16-100-P-L	131,404.9	131,404.9
17-100-R-S	77,484.06	77,484.06
18-100-R-L	126,504.3	126,504.3

Demon Value

Several values have been tested to select the best initial value of the demon (D): 1000, 2500, 5000, and 7500. The results in Table 4 indicated no improvement in objective function if the demon value was greater than 5000. Thus, the demon value D was taken to be 5000.

Table 4. Results of tuning the demon value of the ABC.

Dataset Instances	Demon Value			
	1000	2500	5000	7500
13-100-F-S	41,472.86	41,596.33	41,954.86	41,954.86
14-100-F-L	66,453.66	66,683.84	66,683.84	66,683.84
15-100-P-S	72,170.54	72,449.56	72,449.56	72,449.56
16-100-P-L	131,404.9	131,404.9	131,404.9	131,404.9
17-100-R-S	77,484.06	77,484.06	77,484.06	77,484.06
18-100-R-L	126,504.3	126,504.3	126,504.3	126,504.3

5.3. Parameter Tuning of ABC(C)

Since we used the same ABC algorithm, with little modification, in the ABC with the clustering approach, we adopted the same parameter values that have been used in the ABC for our ABC(C).

5.4. Experimental Results

As previously mentioned, we proposed two versions of the ABC to improve the initial solutions: the ABC used with GRASP, and the ABC(C) used with the clustering methods, i.e., K-means-GRASP(C), modified K-means-GRASP(C), and ACO-GRASP(C). We also have two separate strategies for the scout bee phase. S1 constructs the solution by randomly selecting one of three strategies: random insertion, greedy, and randomised insertion or greedy insertion. After testing this strategy, we found the random insertion heuristic did not improve the results compared to others, so we eliminated this choice and kept the other two. The second strategy, S2, works like the GRASP in [7]. Therefore, we have eight overall proposed methods named as follows: (1) ABC(S1), (2) ABC(S2), (3) K-means-GRASP(C)-ABC(C)S1, (4) K-means-GRASP(C)-ABC(C)S2, (5) modified K-means-GRASP(C)-ABC(C)S1, (6) modified K-means-GRASP(C)-ABC(C)S2, (7) ACO-GRASP(C)-ABC(C)S1, and (8) ACO-GRASP(C)-ABC(C)S2. All proposed algorithms were run five times, as performed with the previous algorithms in [6] and [7]. We compared our proposed algorithms and the state-of-the-art algorithms presented in [10,11], which are C1-Seq VNS, C2-Seq VNS, C1-Self Adaptive VNS, C2-Self Adaptive VNS and IPPD, for solving the MVPPDP in terms of the processing time and profits gained.

First, with respect to the processing time, all previous algorithms have been run for 5 min for each instance. We note, though, that this time is not considered acceptable for the m-commerce apps (e.g., Hungerstation, MRSOOL, etc.) that are used daily and rely on the

speed of responding, so in our case, we used the stopping condition of 500 iterations to measure the processing time. Table 5 presents the comparison results between our proposed algorithms and state-of-the-art algorithms in terms of time (in seconds). Also, the average of each category (highlighted rows in Table 5) has been used to compare the performance of the algorithms as shown in Figure 2. From Table 5 and Figure 2, it is evident that our proposed clustering-based algorithms outperform all previous algorithms with respect to processing time in all instances, except for instances involving 1000 customers. Also, the other proposed algorithms that are not based on clustering (ABC(S1) and ABC(S2)) were faster than the previous algorithms for small- and medium-sized instances. The reason why our algorithms do not excel in large-sized instances is probably due to the scout bee strategies that have been used, namely the greedy method in the first strategy and the GRASP in the second. In the greedy method, the customer pairs are selected based on a specific measure and then inserted into the best position that leads to decreasing the travel cost. This approach gives good results but consumes time, especially in large-sized instances. In the second strategy, the unvisited customer pairs are selected based on computing the IR, which also leads to consuming time in large-sized instances. Overall, though, the clustering of the search space clearly helped to process large-sized instances and made the processing time faster compared to other algorithms that deal with all customers without clustering.

Table 5. Comparison between state-of-the-art algorithms and the proposed algorithms based on time (seconds).

Dataset	Previous Algo- rithms(5 min Runtime)	K-Means- GRASP- ABC(C)S1	K-Means- GRASP- ABC(C)S2	Modified K-Means- GRASP- ABC(C)S1	Modified K-Means- GRASP- ABC(C)S2	ACO- GRASP- ABC(C)S1	ACO- GRASP- ABC(C)S2	ABC(S1)	ABC(S2)
1-20-F-S	300	14.48	14.47	15.20	14.58	16.78	12.12	31.04	30.65
2-20-F-L	300	12.13	12.42	13.82	13.17	13.51	11.45	30.20	30.56
3-20-P-S	300	16.62	17.48	17.83	17.17	18.72	15.85	33.77	39.28
4-20-P-L	300	12	12.01	12.61	12.24	15.63	12.98	24.43	28.45
5-20-R-S	300	14.87	15.21	14.97	14.57	16.67	14.57	30.79	31.17
6-20-R-L	300	11.32	12.38	13.47	13.71	13.63	12.25	22.62	21.56
Average-20	300	13.57	14	14.65	14.24	15.82	13.20	28.81	30.28
7-50-F-S	300	23.24	24.29	25.67	25.49	25.45	22.03	50.10	54.80
8-50-F-L	300	19.79	21.24	21.53	21.45	22.18	20.65	43.81	46.63
9-50-P-S	300	25.67	24.64	26.58	25.15	25.87	20.49	50.55	54.72
10-50-P-L	300	21.63	22.64	24.41	22.96	23.34	20.90	44.77	48.77
11-50-R-S	300	26.31	26.58	26.86	26.05	27.49	20.23	48.90	51.94
12-50-R-L	300	22.73	20.93	23.37	21.87	22.77	20.91	45.94	48.34
Average-50	300	23.23	23.39	24.74	23.83	24.52	20.87	47.34	50.86
13-100-F-S	300	37.87	33.57	36.58	35.10	36.89	33.36	80.65	83.81
14-100-F-L	300	31.42	33.67	34.59	34.38	35.55	33.14	73.90	81.74
15-100-P-S	300	31.03	32.03	36.98	35.55	36.75	33.23	76.39	82.69
16-100-P-L	300	33.54	34.33	36.50	34.57	36.20	32.41	74.19	82.93
17-100-R-S	300	38	35.92	37	35.81	38.19	34.03	82.59	86.96
18-100-R-L	300	36.19	35.37	34.38	33.34	37.49	33.63	77.02	81.01
Average-100	300	34.68	34.15	36.01	34.79	36.85	33.30	77.46	83.19
19-250-F-S	300	73.50	72.85	78.10	73.30	74.19	67.79	176.81	204.37
20-250-F-L	300	79.65	73.28	80.50	76.28	77.66	65.10	181.10	195
21-250-P-S	300	74.20	72.09	75.23	72.88	74.05	63.51	169.26	195.17
22-250-P-L	300	73.82	72.718	79.53	74.40	77.86	65.44	179	193.60
23-250-R-S	300	74.67	72.99	77.63	72.38	76.21	66.44	177.25	193.93
24-250-R-L	300	79.55	73.21	84.04	73.84	79.62	69.46	178.99	194.47
Average-250	300	75.90	72.86	79.17	73.85	76.60	66.29	177.07	196.09
25-500-F-S	300	163.89	155.32	172.74	158.36	162.15	142.52	388.62	474.16
26-500-F-L	300	167.03	155.73	177.18	160.25	170.25	148.98	415.25	466.38
27-500-P-S	300	160.91	146.27	172.04	158.45	162.76	140.62	386.55	465.87
28-500-P-L	300	168.60	144.23	180.26	158.74	169.89	141.34	435.34	464.62

Table 5. Cont.

Dataset	Previous Algorithms(5 min Runtime)	K-Means-GRASP-ABC(C)S1	K-Means-GRASP-ABC(C)S2	Modified K-Means-GRASP-ABC(C)S1	Modified K-Means-GRASP-ABC(C)S2	ACO-GRASP-ABC(C)S1	ACO-GRASP-ABC(C)S2	ABC(S1)	ABC(S2)
29-500-R-S	300	170.15	140.16	170.50	158.58	162.42	139.80	383.89	468.15
30-500-R-L	300	169.78	141.24	174.22	161.07	169.30	143.36	428.65	464.89
Average-500	300	166.73	147.16	174.49	159.24	166.13	142.77	406.38	467.35
31-1000-F-S	300	432.51	354.73	448.29	408.67	426.32	397.22	1040.22	1406.22
32-1000-F-L	300	439.85	360.11	447.72	423.01	461.18	430.54	1197.38	1438.43
33-1000-P-S	300	434.34	364.52	432.93	418.51	391.01	416.76	1128.14	1398.49
34-1000-P-L	300	448.06	378.57	440.70	435.69	389.93	430.39	1204.97	1416.64
35-1000-R-S	300	451.79	404.20	425.83	421.47	381.24	411.31	1110.80	1296.40
36-1000-R-L	300	420.15	410.26	446.23	418.03	393.29	420.10	1222.33	1286.07
Average-1000	300	437.78	378.73	440.28	420.90	407.16	417.72	1150.64	1373.71

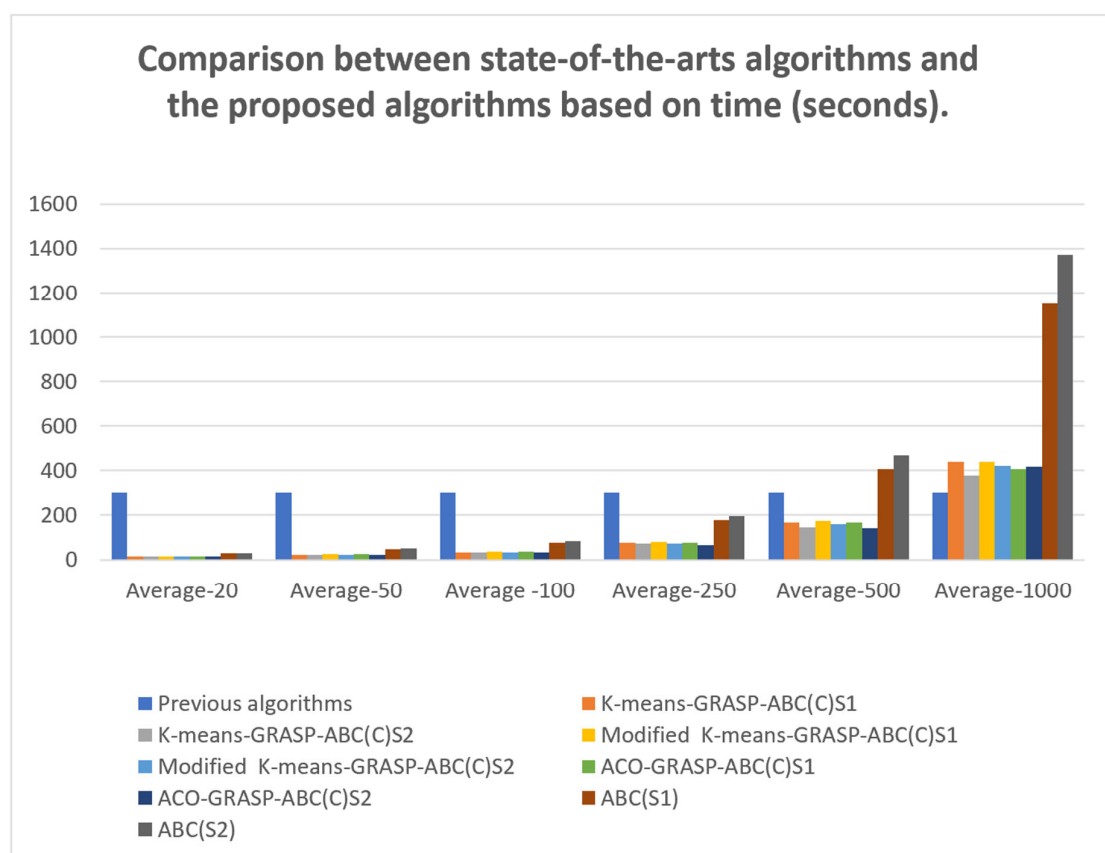


Figure 2. Comparison between state-of-the-arts algorithms and the proposed algorithms based on time (seconds).

On the other hand, we compared our proposed algorithms and the previous algorithms in terms of profit as computed in Equation (1). Table 6 presents the average profit for five runs, while Table 7 presents the best profit obtained over five runs and the gap between the best profit and the best known profit using the following equation:

$$Gap = \frac{Best\ profit\ over\ five\ runs - Best\ known\ profit}{Best\ known\ profit} * 100 \quad (20)$$

Table 6. Comparison of our proposed algorithms with previous methods based on average profit.

Dataset	<i>C1—Seq VNS [9]</i>	<i>C2—Seq VND [9]</i>	<i>C1—Self Adaptive VND [9]</i>	<i>C2—Self Adaptive VND [9]</i>	<i>IPPD [10]</i>	<i>K-Means- GRASP- ABC(C)S1</i>	<i>K-Means- GRASP- ABC(C)S2</i>	<i>Modified K-Means- GRASP- ABC(C)S1</i>	<i>Modified K-Means- GRASP- ABC(C)S2</i>	<i>ACO- GRASP- ABC(C)S1</i>	<i>ACO- GRASP- ABC(C)S2</i>	<i>ABC(S1)</i>	<i>ABC(S2)</i>
1-20-F-S	30,315.5	30,315.5	30,315.5	30,315.5	30,315.5	30,267.75	30,427.58	27,007.58	27,007.58	25,490.76	25,490.76	30,395.92	30,294.77
2-20-F-L	38,765.6	38,765.6	37,866.3	38,765.6	39,179.66	37,642.38	37,468.82	39,773.34	39,240.95	36,342.53	36,322.33	39,459.15	36,153.93
3-20-P-S	51,473.4	52,128.4	52,565.1	52,565.1	52,566	51,481.8	43,340.71	51,481.8	43,009.55	42,917.34	34,107.64	52,684.98	52,565.1
4-20-P-L	58,365.6	58,365.6	58,365.6	58,365.6	58,777	57,350.35	57,086.36	57,350.35	57,301.88	56,523.32	47,367.54	58,888.64	56,808.21
5-20-R-S	36,469.5	36,469.5	36,469.5	36,469.5	36,469.5	34,090.58	34,090.58	30,237.02	29,843.58	31,741.08	31,741.08	36,469.52	36,469.52
6-20-R-L	43,368.6	43,368.6	43,368.6	43,368.6	43,781.81	42,215.53	42,086.44	44,401.45	43,736.53	40,769.67	40,450.55	44,244.27	42,143.26
Average-20	43,126.37	43,235.53	43,158.43	43,308.32	43,514.91	42,174.73	40,750.08	41,708.59	40,023.35	38,964.12	35,913.32	43,690.41	42,405.8
Standard Deviation	10,289.94	10,399.10	10,559.14	10,475.04	10,560.13	10,436.42	9349.23	11,822.81	10,907.82	10,656.36	7500.01	10,573.40	10,312.25
7-50-F-S	22,306	22,306	22,306	22,306	22,307.15	18,662.83	21,603.83	18,662.83	21,603.83	21,281.88	25,123.15	24,975.25	24,998.45
8-50-F-L	58,050.8	58,050.8	57,303.4	57,996.7	58,050.8	50,772.64	50,873.68	50,881.55	50,773.54	46,699.31	47,534.79	53,555.25	53,268.82
9-50-P-S	57,651.7	57,651.7	57,651.7	57,651.7	57,651.7	58,996.17	53,258.83	58,996.17	53,064.14	55,643.25	55,643.25	57,651.71	57,651.71
10-50-P-L	132,648	132,648.0	128,353	128,334.8	129,640.3	109,726.8	117,770.8	116,372	116,149.7	97,740.02	106,491.9	124,819.3	128,477.1
11-50-R-S	33,340.4	33,340.4	33,340.4	33,340.4	33,340.4	28,865.48	28,907.33	28,865.48	28,907.33	31,251.34	29,969.16	33,054.57	33,098.58
12-50-R-L	79,350.3	79,350.3	77,655.5	79,350.3	77,832.56	76,422.62	76,503.28	76,309.46	76,408.91	64,359.55	64,519.79	76,582.15	75,732.86
Average-50	63,891.2	50,139.84	62,768.33	63,163.32	63,137.16	57,241.09	58,152.95	58,347.92	57,817.91	52,829.22	54,880.35	61,773.03	62,204.58
Standard Deviation	39,248.45	22,523.36	37,635.69	37,749.54	38,077.61	33,048.63	35,097.31	35,184.37	34,547.72	27,041.68	29,384.58	35,937.41	37,167.44
13-100-F-S	41,464.84	41,955.2	41,523.5	42,174	41,896.86	35,386.24	36,055.01	35,303.67	36,625.9	33,147.92	33,187.57	40,596.77	40,543.56
14-100-F-L	81,271.2	81,971	80,080.4	80,500.2	79,479.88	58,704.7	60,780.74	57,607.07	60,035.43	51,106.58	47,074.98	64,937.47	67,237.63
15-100-P-S	70,387.4	70,398.5	71,113.8	71,082.8	69,436.25	54,197	57,206.6	49,566.36	59,481.84	60,614.78	45,995.09	68,874.02	71,290.33
16-100-P-L	124,288	123,931	122,691.8	122,183.2	125,057	98,140.3	104,031.6	100,122.4	110,099	92,681.38	96,365.75	112,690.5	124,487.1
17-100-R-S	76,885.4	78,418.8	76,954.4	79,651.2	77,012.75	61,579.81	61,579.81	61,960.78	61,698.84	52,689.7	52,163.72	77,334.8	75,925.47
18-100-R-L	131,052.6	130,670.4	130,403.4	130,366.6	131,058.6	98,522.81	100,291.3	104,771.9	109,450.2	92,423.54	93,867.6	115,108.7	122,320.4
Average-100	87,558.24	87,890.82	87,127.88	87,659.67	87,323.56	67,755.14	69,990.84	68,222.03	72,898.54	63,777.32	61,442.45	79,923.7	83,634.09
Standard Deviation	34,099	33,674.36	33,546.8	33,090.45	34,316.01	25,383.15	26,636.59	28,060.86	30,021.67	24,032.04	26,834.7	29,022.89	33,175.38
19-250-F-S	73,259.94	72,445.3	72,703	72,287.9	70,799.19	36,045.94	32,341.35	43,729.29	38,692.81	34,038.31	33,942.57	48,988.62	46,216.19
20-250-F-L	132,996	132,438	132,394.4	133,404	129,485.3	54,144.64	63,245.12	70,325.74	81,370.79	51,239.48	57,273.9	71,397.91	81,288.1
21-250-P-S	111,334.4	111,817.8	109,449.4	109,355.8	110,440.3	62,245.05	71,066.44	74,931.5	83,992.55	72,198.2	69,410.97	86,624.32	93,568.56
22-250-P-L	197,899.8	197,214.2	194,131.2	194,549	194,338.6	112,454.5	131,917.5	133,567.4	155,750.3	95,906.32	127,789.5	146,104	160,301.3
23-250-R-S	162,410.8	162,448.2	158,164.4	159,129.6	158,414.6	99,900	90,042.33	109,717.8	93,812.47	102,500.1	99,734.68	125,201.2	120,776
24-250-R-L	269,072.8	268,713.8	266,559	267,291.4	266,204.2	148,156.7	176,579.9	180,807.7	177,336	146,270.4	164,012.2	204,037.4	203,859.8
Average-250	157,829	157,512.9	155,566.9	156,003	154,947	85,491.14	94,198.78	102,179.9	105,159.2	83,692.14	92,027.3	113,725.6	117,668.3
Standard Deviation	69,184.1	69,165.71	68,327.76	68,666.69	68,777.74	42,022.43	52,050.37	49,795.51	51,632.15	40,174	48,193.98	56,662.62	57,036.7

Table 7. Cont.

Dataset	Best Known Profit	C1—Seq VNS [4]		C2—Seq VND [4]		C1—Self Adaptive VND [4]		C2—Self Adaptive VND [4]		IPPD [10]		K-Means-GRASP-ABC(C)S1	
		Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap
9-50-P-S	57,651.7	57,651.7	0	57,651.7	0	57,651.7	0	57,651.7	0	57,651.7	0	58,996.17	2.3
10-50-P-L	132,648	132,648	0	132,648	0	132,648	0	132,648	0	132,648	0	114,962.9	−13.3
11-50-R-S	33,340.4	33,340.4	0	33,340.4	0	33,340.4	0	33,340.4	0	33,340.4	0	28,907.33	−13.2
12-50-R-L	79,350.3	79,350.3	0	79,350.3	0	79,350.3	0	79,350.3	0	77,832.56	−1.95	76,479.84	−3.6
Average-50	63,913.7	63,891.20	−0.1	63,891.20	−0.10	63,891.20	−0.10	63,891.20	−0.10	63,638.44	−0.43	58,174.87	−8.9
13-100-F-S	44,402.3	42,174.4	−5.3	42,174.4	−5.3	42,174.4	−5.3	42,215.5	−5.2	41,956.25	−5.83	36,322.94	−18.1
14-100-F-L	81,999.4	81,999.4	0	81,999.4	0	81,621	−0.5	80,848.1	−0.2	88,891.4	0	60,767.96	−25.8
15-100-P-S	71,512.4	70,836	−1	71,326.4	−0.3	71,399.2	−0.2	71,399.2	−0.2	71,234.58	−0.39	54,428.34	−23.8
16-100-P-L	125,032	124,770	−0.2	124,770	−0.2	123,766	−1	124,910	−0.1	125,786.7	−0.6	99,860.9	−20.1
17-100-R-S	80,878.8	80,878.8	0	78,818	−2.6	80,878.8	0	80,878.8	0	80,878.8	0	61,647.98	−23.7
18-100-R-L	131,845	131,473	−0.3	131,845	0	131,339	−0.4	131,339	−0.4	131,845	0	102,403.8	−22.3
Average-100	89,278.32	97,991.44	−1.13	88,488.87	−1.40	88,529.73	−1.23	88,598.43	−1.02	90,098.79	−1.14	69,238.65	−22.4
19-250-F-S	75,309.1	73,610.9	−2.3	73,575.7	−2.4	73,835.5	2	73,544.7	−2.4	72,038.6	−4.54	39,672.94	−47.3
20-250-F-L	136,594	134,889	−1.3	134,186	−1.8	132,870	−2.8	135,622	−0.7	131,012.9	−4.26	54,144.64	−60.3
21-250-P-S	113,886	112,427	−1.3	112,480	−1.3	112,428	−1.3	110,062	−3.5	112,747.3	−1.01	69,574.44	−38.9
22-250-P-L	200,402	200,402	0	198,611	−0.9	195,135	−2.7	196,524	−2	198,241.2	−1.09	112,454.5	−43.8
23-250-R-S	164,070	162,960	−0.7	164,070	0	162,115	−1.2	162,115	−1.2	161,804.7	−1.4	103,456	−36.9
24-250-R-L	274,563	271,185	−1.2	271,193	−1.2	272,957	−0.6	269,790	−1.8	269,734.7	−1.79	148,156.7	−46
Average-250	160,804	159,245.65	−1.13	159,019.28	−1.27	158,223.42	−1.10	174,822.60	−1.93	157,596.57	−2.35	87,909.87	−45.3
25-500-F-S	167,080	161,063	−3.7	161,040	−3.8	159,064	−5	161,375	−3.5	160,592.1	−4.04	59,737.27	−64.2
26-500-F-L	259,319	249,171	−4.1	249,661	−3.9	247,986	−4.6	249,312	−4	249,105.7	−4.1	107,852.1	−58.4
27-500-P-S	206,788	203,832	−1.5	205,480	−0.6	200,764	−3	201,590	−2.6	208,287.7	0.72	105,678.8	−48.8
28-500-P-L	331,291	321,471	−3.1	325,522	−1.8	320,872	−3.2	321,155	−3.2	337,358.9	1.82	229,525.6	−30.7
29-500-R-S	281,711	279,105	−0.9	278,233	−1.3	274,419	−2.7	276,392	−1.9	265,040	−6.29	126,298.6	−55.1
30-500-R-L	417,282	407,689	−2.4	409,755	−1.8	407,849	−2.3	409,262	−2	411,967.6	−1.29	259,683.6	−37.7
Average-500	277,245.2	270,388.50	−2.61	271,615.17	−2.20	268,492.33	−3.47	269,847.67	−2.87	272,058.67	−2.20	165,807.7	−40.1
31-1000-F-S	173,826	154,196	−12.7	151,943.2	−12.2	154,357	−12.6	156,106	−11.4	152,358.7	−11.8	33,184.03	−80.9
32-1000-F-L	247,385	221,958	−11.5	221,322	−11.8	224,999	−9.9	226,913	−9	222,089.1	−10.1	92,970.35	−62.4
33-1000-P-S	464,580	450,349	−3.2	450,337	−3.2	442,311	−5	442,218	−5.1	457,759.4	1.47	272,126.9	−41.4
34-1000-P-L	712,918	693,407	−2.8	672,910	−5.9	684,121	−4.2	683,211	−4.3	681,370.5	−3.26	498,835.3	−30
35-1000-R-S	492,083	463,625	−6.1	472,626	−4.1	465,022	−5.8	469,162	−4.9	461,919.6	−4.90	207,134	−57.9
36-1000-R-L	705,167	673,951	−4.6	669,926	−5.3	676,720	−4.4	684,217	−3.1	676,159.7	−2.90	464,549.3	−34.1
Average-1000	465,993.2	405,958	−6.8	439,844.03	−7.08	441,255	−6.98	443,638	−6.3	441,942.83	−5.2	261,466.65	−43.8

Table 7. Cont.

Dataset	K-Means-GRASP-ABC(C)S2		Modified K-Means-GRASP-ABC(C)S1		Modified K-Means-GRASP-ABC(C)S2		ACO-GRASP-ABC(C)S1		ACO-GRASP-ABC(C)S2		ABC(S1)		ABC(S2)	
	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap
1-20-F-S	30,427.58	0.3	27,007.58	−10.9	27,007.58	−10.9	25,490.76	−15.9	25,490.76	−15.9	30,554.62	0.78	30,315.52	0
2-20-F-L	37,556.07	−4.5	40,104.23	1.9	39,240.95	−0.2	36,423.3	−7.4	36,322.33	−7.6	40,104.23	1.9	36,234.48	−7.8
3-20-P-S	44,334.2	−16.1	51,515.53	−2.5	44,334.2	−16.1	42,944.78	−18.7	34,107.64	−35.4	53,164.53	0.58	52,565.1	−0.5
4-20-P-L	57,350.35	−1.7	57,350.35	−1.7	57,350.35	−1.7	56,603.63	−3	47,402.91	−18.7	59,140.98	1.32	57,286.89	−1.8
5-20-R-S	34,090.58	−6.5	30,827.18	−15.4	29,843.58	−18.1	31,741.08	−12.9	31,741.08	−12.9	<u>36,469.52</u>	0	<u>36,469.52</u>	0
6-20-R-L	42,169.13	−3.6	44,764.07	2.2	43,780.72	0.04	41,071.62	−6.1	40,600.21	−7.2	44,707.23	2.1	42,873.71	−2
Average-20	40,987.99	−5.8	41,928.16	−3.6	40,259.56	−7.4	39,045.86	−10.2	35,944.16	−17.4	44,023.52	1.16	42,624.2	−2
7-50-F-S	21,603.83	−3.7	21,603.83	−3.7	21,603.83	−3.7	21,281.88	−5.1	25,123.15	11.9	24,975.25	11.2	24,998.45	11.3
8-50-F-L	51,001.04	−12.1	51,001.04	−12.1	50,896.3	−12.3	49,128.45	−15.3	50,061.24	−13.7	53,710.99	−7.4	54,288.08	−6.4
9-50-P-S	54,165.39	−6	54,165.39	−6	53,064.14	−7.9	55,643.25	−3.4	55,643.25	−3.4	<u>57,651.71</u>	0	<u>57,651.71</u>	0
10-50-P-L	119,099.6	−10.2	119,099.6	−10.2	116,868.6	−11.8	100,393.3	−24.3	106,544.8	−19.6	<u>126,360</u>	−4.7	133,743.2	0.82
11-50-R-S	28,907.33	−13.2	28,907.33	−13.2	28,907.33	−13.2	31,251.34	−6.2	29,969.16	−10.1	33,058.21	−0.8	33,159.12	−0.5
12-50-R-L	76,697.65	−3.3	76,697.65	−3.3	76,604.84	−3.4	65,792.21	−17	65,752.8	−17.1	77,467.34	−2.3	76,517.36	−3.5
Average-50	58,579.14	−8.3	58,579.14	−8.3	57,990.84	−9.2	53,915.07	−15.6	55,515.73	−13.1	62,203.92	−2.6	63,392.99	−0.8
13-100-F-S	36,188.32	−18.4	35,937.96	−19	38,323.6	−13.6	33,565.46	−24.4	33,566.06	−24.4	41,812.08	−5.8	41,636.8	−6.2
14-100-F-L	63,488.24	−22.5	60,851.27	−25.7	61,351.99	−25.1	52,328.66	−36.1	49,789.52	−39.2	66,375.92	−19	72,595.41	−11.4
15-100-P-S	60,815.37	−14.9	54,190.18	−24.2	60,554.49	−15.3	60,614.78	−15.2	47,498.28	−33.5	69,275.79	−3.1	71,400.63	−0.1
16-100-P-L	104,786.4	−16.1	102,121.7	−18.3	111,728.2	−10.6	95,836.25	−23.3	101,142.5	−19.1	115,579.2	−7.5	128,473.2	2.7
17-100-R-S	61,647.98	−23.7	61,962.79	−23.3	61,962.79	−23.3	52,689.7	−34.8	52,689.7	−34.8	77,484.06	−4.1	77,484.06	−4.1
18-100-R-L	101,013.6	−23.3	105,175.3	−20.2	110,683.8	−16	95,363.69	−27.6	97,568.94	−25.9	119,554.3	−9.3	126,871.5	−3.7
Average-100	71,323.32	−20.1	70,039.87	−21.5	74,100.81	−17	65,066.42	−27.1	63,709.17	−28.6	81,680.23	−8.5	86,410.27	−3.2
19-250-F-S	35,443.33	−52.9	48,079.56	−36.1	39,848.03	−47	36,309.86	−51.7	36,612.19	−51.3	36,309.86	−51.7	36,612.19	−51.3
20-250-F-L	67,014.68	−50.9	70,325.74	−48.5	87,575.52	−35.8	51,239.48	−62.4	62,831.21	−54	51,239.48	−62.4	62,831.21	−54
21-250-P-S	77,941.82	−31.5	81,510.78	−28.4	92,124.73	−19.1	73,654.78	−35.3	73,143.68	−35.7	73,654.78	−35.3	73,143.68	−35.7
22-250-P-L	135,164.3	−32.5	133,567.4	−33.3	161,377.2	−19.4	95,906.32	−52.1	131,668.8	−34.2	95,906.32	−52.1	131,668.8	−34.2
23-250-R-S	97,231.36	−40.7	113,547.4	−30.7	95,471.16	−41.8	107,628.9	−34.4	101,863.1	−37.9	107,628.9	−34.4	101,863.1	−37.9
24-250-R-L	187,891.9	−31.5	180,807.7	−34.1	180,803	−34.1	146,270.4	−46.7	166,523.5	−39.3	146,270.4	−46.7	166,523.5	−39.3
Average-250	100,114.6	−37.7	104,639.8	−34.9	109,533.3	−31.8	85,168.2	−47	95,440.4	−40.6	85,168.2	−47	95,440.4	−40.6
25-500-F-S	74,692.46	−55.2	51,581.38	−69.1	76,481.48	−54.2	61,718.84	−63	54,247.37	−67.5	78,432.78	−53	86,250.21	−48.3
26-500-F-L	124,478.4	−51.9	106,505.3	−58.9	131,572.4	−49.2	72,388.37	−72	93,308.66	−64	112,172	−56.7	126,203.2	−51.3
27-500-P-S	152,566.6	−26.2	107,797.6	−47.8	150,645.3	−27.1	109,238	−47.1	112,111.9	−45.7	136,168.2	−34.1	148,384.5	−28.2

Table 7. Cont.

Dataset	K-Means-GRASP-ABC(C)S2		Modified K-Means-GRASP-ABC(C)S1		Modified K-Means-GRASP-ABC(C)S2		ACO-GRASP-ABC(C)S1		ACO-GRASP-ABC(C)S2		ABC(S1)		ABC(S2)	
	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap	Best Profit	Gap
28-500-P-L	249,759.7	−24.6	225,812.4	−31.8	247,180.6	−25.3	152,573.5	−53.9	196,969.2	−40.5	201,973.8	−39	225,040.5	−32
29-500-R-S	172,397.2	−38.8	108,998.5	−61.3	175,267.1	−37.7	141,743.7	−49.6	128,570.6	−54.3	178,693.9	−36.5	173,003.3	−38.5
30-500-R-L	275,488.2	−33.9	241,454	−42.1	294,424.7	−29.4	167,819.5	−59.7	214,081.3	−48.6	239,675.7	−42.5	261,517.2	−37.3
Average-500	174,897.1	−36.9	140,358.2	−49.3	179,261.9	−35.3	117,580.3	−57.5	133,214.8	−51.9	157,852.7	−43	170,066.5	−38.6
31-1000-F-S	51,188.29	−70.5	39,530.22	−77.2	51,585.38	−70.3	23,266.5	−86.6	25,563.9	−85.2	33,125.19	−80.9	42,197.08	−75.7
32-1000-F-L	97,093.51	−60.7	92,142.15	−62.7	97,772.35	−60.4	24,110.85	−90.2	47,620.03	−80.7	57,137.24	−76.9	73,126.9	−70.4
33-1000-P-S	293,900.2	−36.7	236,797.1	−49	287,774.9	−38	178,654.3	−61.5	209,372.1	−54.9	240,944.9	−48.1	267,814.2	−42.3
34-1000-P-L	536,199.8	−24.7	492,753.1	−30.8	534,689.9	−24.9	346,244.9	−51.4	381,116.4	−46.5	397,930	−44.1	430,894.4	−39.5
35-1000-R-S	264,295.2	−46.2	192,030.8	−60.9	249,078.6	−49.3	124,538.1	−74.6	181,237	−63.1	212,979.1	−56.7	232,298.7	−52.7
36-1000-R-L	458,126.7	−35	407,703.7	−42.1	463,915.7	−34.2	297,212.8	−57.8	335,935.6	−52.3	364,643.1	−48.2	380,268.9	−46
Average-1000	276,004.15	−40.7	243,492.8	−47.7	280,802.8	−39.7	165,671.2	−64.4	196,807.5	−57.7	217,793.3	−53.2	237,766.7	−48.9

The best-known profit was computed in [9] by running each instance for 20 h, and then the best profit was saved as the best-known profit. As can be seen from Tables 6 and 7, our proposed algorithms perform well in terms of profit for small-size instances, with the ABC(S1) returning six new best solutions and two other solutions that were equal to the best-known solutions. The ABC(S2) also achieved three new best solutions and three solutions that are equal to the best-known solution. Also, the modified K-means-GRASP(C)-ABC(C)S1 achieved two new best solutions. In addition, the K-means-GRASP(C)-ABC(C)S1, K-means-GRASP(C)-ABC(C)S2, modified K-means-GRASP(C)-ABC(C)S2, and ACO-GRASP(C)-ABC(C)S2 each produced one new best solution. Overall, the general performance for all proposed algorithms was acceptable and close to the previous algorithms in terms of average profit on these instances.

For the medium- and large-sized instances, though, our proposed algorithms did not do well compared to previous methods, as evident by the large average gap percentage compared to the best-known results shown in Table 6. There are many possible reasons for this:

- Again, the methods used in the scout phase of the ABC seem to have an effect on the performance of the proposed algorithms. The first is the greedy randomised strategy, which selects a random customer pair, meaning the worst pair (either with low revenue or separated by a great distance) may be selected, resulting in poor solution quality. Moreover, in some cases, the neighbourhood operators cannot sufficiently enhance the quality of the solution because of time violations or load constraints. The second method uses the GRASP, which constructs solutions by randomly selecting from the Restricted Candidate List (RCL), which is half the size of the Candidate Solution Set (CSS). Thus, selecting unvisited customers is not a problem for small-sized instances because the number of customer pairs is small, which means the chance of selecting the best customer pair (i.e., that with the highest IR2 value) is high. In contrast, in medium- and large-sized instances, where the size of the RCL is large, the chance of selecting the best customer pair decreases, which affects the quality of the solutions.
- The number of solutions in a population, the number of iterations, and the limit value play important roles in the algorithms' performance. In our algorithms, we rely on small numbers of both solutions and iterations because a fast processing time is a critical demand for daily m-commerce apps. These decreased iteration numbers and population sizes, however, reduce the chances of achieving satisfactory solution quality.
- All previous algorithms were based on VNS, which has the ability to change the size of the neighbourhood search each time the solution quality is not enhanced. Thus, expanding the neighbourhood structure in a systematic way helps the VNS fully explore the neighbourhood and obtain good solutions. In contrast, the diversification feature in our approach is based either on the greedy feature (inserting the customer pair with the highest IR into the best position) or on a combination of greedy and randomised features (inserting a random customer pair into the best position or inserting a random customer pair from RCL into the best position). However, these strategies seem less successful at diversification than the VNS, which relies on an organised neighbourhood exploration mechanism.

Figure 3 compares our proposed algorithms based on the average of the best solutions for each category of instances (highlighted rows in Table 6). It is clear that the ABC in both versions (S1 and S2) still has the preference over all the other algorithms for the instances comprising 20, 50, 100, 250, and 500 customers. In contrast, both versions of K-means-GRASP(C)-ABC(C) and modified K-means-GRASP(C)-ABC(C) outperform the GRASP(V2)-ABC in both versions (S1 and S2) for the 1000-customer instances. The main reasons for the poor performance of clustering algorithms in small- and medium-sized instances is the distribution of customer pairs in K-means and modified K-means, where the clustering process may result in putting a small number of customers within a cluster, which means that the vehicle still has remaining time to serve, but no more customers exist

within that cluster. Thus, we conclude that the clustering algorithms (except ACO) do well in large-sized instances only.

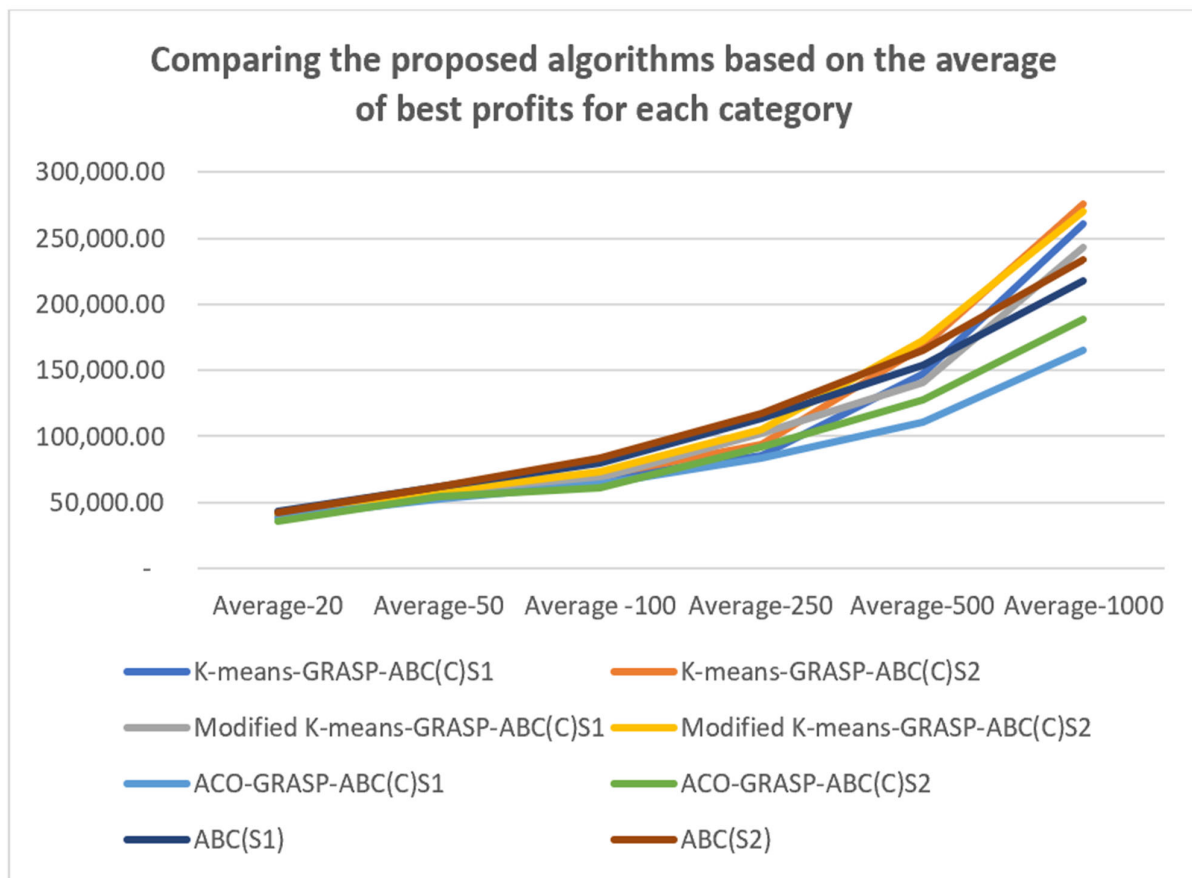


Figure 3. Comparison the proposed algorithms based on the average of best profits for each category.

Regarding the ACO-GRASP(C)-ABC(C)(S1 and S2), it is clear that it has the worst performance among all algorithms. Again, the clustering method of ACO has a significant effect on results. Specifically, the randomness feature of ACO contributes to an inaccurate clustering of customer pairs.

In addition, as seen in the highlighted rows in Table 7, we noticed that the second strategy for scout bees outperforms the first one in many instances because the first strategy includes selecting either the best customer pairs or random pairs, and the random selection option may affect the quality of the solution, while the second strategy selects customer pairs randomly from the RCL, which contains the best customer pairs in the CSS.

6. Conclusions

In this paper, we present eight new metaheuristics to improve the initial solutions for the multi-vehicle profitable pickup and delivery problem (MVPPDP). We proposed two versions of the ABC (Artificial Bee Colony) to improve the initial solutions: the ABC with GRASP and the ABC (C) with GRASP and clustering, i.e., K-means-GRASP (C), modified K-means-GRASP (C), and ACO-GRASP (C). We also have two separate strategies for the scout bee phase. The first strategy, S1, constructs the solution by randomly selecting one of two strategies: greedy and randomised insertion or just greedy insertion. The second strategy, S2, is based on the GRASP. Thus, we have eight different proposed methods. When compared to the state-of-the-art methods in terms of processing time, all eight methods have preference in all instances (except for the 1000-customer instances). Moreover, all proposed algorithms performed well for small-sized instances in terms of profits by achieving thirteen

new best solutions and five equal solutions to the best-known solutions. However, the algorithms slightly lag behind in medium- and large-sized instances due to the greedy randomised strategy and the GRASP that have been used in the scout bee phase. Moreover, our algorithms prioritise minimal solutions and iterations for rapid processing times in daily m-commerce apps, while reducing iteration counts and population sizes reduces the likelihood of obtaining good solution quality.

For future research, more neighbourhood operators will be carefully investigated in the scout bee phase to be applicable within the constraints of the MVPPDP, thus increasing the chances of producing high-quality solutions. Moreover, we will try other population-based metaheuristics, such as GAs, evolutionary algorithms, or other swarm-based metaheuristics that may contribute to new solutions for the MVPPDP.

Author Contributions: Conceptualization, A.I.A.; methodology, A.I.A.; formal analysis, A.I.A.; data curation, A.I.A.; Writing—original draft, A.I.A.; funding acquisition, A.I.A.; Supervision, M.I.H.; writing—review and editing, M.I.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: <http://prolog.univie.ac.at/research/MVPPDP/instances.zip> (accessed on 17 July 2024).

Acknowledgments: The Researchers would like to thank the Deanship of Graduate Studies and Scientific Research at Qassim University for financial support (QU-APC-2024-9/1).

Conflicts of Interest: The authors declare no conflict of interest.

References

- Curry, D. Food Delivery App Revenue and Usage Statistics. 2024. Available online: <https://www.businessofapps.com/data/food-delivery-app-market/> (accessed on 24 January 2024).
- Khalid, H. Benefits of Food Delivery App for Restaurants and Customers. Available online: <https://enatega.com/benefits-of-food-delivery-app/> (accessed on 1 February 2024).
- Team, D.J. Top Food Delivery Apps in Saudi Arabia 2023. Available online: <https://www.digitalgravity.ae/blog/top-food-delivery-apps-in-saudi-arabia/> (accessed on 2 January 2024).
- Gansterer, M.; Küçüktepe, M.; Hartl, R.F. The multi-vehicle profitable pickup and delivery problem. *OR Spectr.* **2017**, *39*, 303–319. [CrossRef]
- Talbi, E.G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: Hoboken, NJ, USA, 2009; ISBN 978-0-470-27858-1.
- Alhujaylan, A.I.; Hosny, M.I. A GRASP-based solution construction approach for the multi-vehicle profitable pickup and delivery problem. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 111–120. [CrossRef]
- Alhujaylan, A.I.; Hosny, M.I. Hybrid Clustering Algorithms with GRASP to Construct an Initial Solution for the MVPPDP. *Comput. Mater. Contin.* **2020**, *62*, 1025–1051. [CrossRef]
- Archetti, C.; Speranza, M.G.; Vigo, D. Vehicle routing problems with profits. In *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed.; SIAM: Philadelphia, PA, USA, 2014; Chapter 10; pp. 273–297. ISBN 978-1-61197-358-7.
- Toth, P.; Vigo, D. *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed.; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2015; ISBN 1611973589.
- Küçüktepe, M. A General Variable Neighbourhood Search Algorithm for the Multi-Vehicle Profitable Pickup and Delivery Problem. University of Vienna 2014. Available online: https://scholar.google.com/scholar?hl=en&as_sdt=0,5&q=A+General+Variable+Neighbourhood+Search+Algorithm+for+the+Multi-Vehicle+Profitable+Pickup+and+Delivery+Problem.+&btnG= (accessed on 22 June 2024).
- Haddad, M.N. An Efficient Heuristic for One-to-One Pickup and Delivery Problems. Ph.D. Thesis, IC/UFF, Niterói, Brazil, 2017.
- Bruni, M.E.; Toan, D.Q. The multi-vehicle profitable pick up and delivery routing problem with uncertain travel times. *Transp. Res. Procedia* **2021**, *52*, 509–516. [CrossRef]
- Karaboga, D. An Idea based on Honey Bee Swarm for Numerical Optimization. Technical Report-tr06, Erciyes University, Engineering Faculty, Computer Engineering Department. 2005. Available online: https://scholar.google.com/scholar?hl=en&as_sdt=0,5&q=An+idea+based+on+honey+bee+swarm+for+numerical+optimization&btnG= (accessed on 22 June 2024).
- Alazzawi, A.K.; Rais, H.M.; Basri, S.; Alsariera, Y.A. PhABC: A Hybrid Artificial Bee Colony Strategy for Pairwise test suite Generation with Constraints Support. In Proceedings of the 2019 IEEE Student Conference on Research and Development (SCORED), Bandar Seri Iskandar, Malaysia, 15–17 October 2019; pp. 106–111, ISBN 9781728126135.
- Szeto, W.Y.; Wu, Y.; Ho, S.C. An artificial bee colony algorithm for the capacitated vehicle routing problem. *Eur. J. Oper. Res.* **2011**, *215*, 126–135. [CrossRef]

16. Wu, B.; Shi, Z. A clustering algorithm based on swarm intelligence. In Proceedings of the 2001 International Conferences on Info-Tech and Info-Net. Proceedings (Cat. No.01EX479), Beijing, China, 29 October–1 November 2001; pp. 58–66. ISBN 0-7803-7010-4.
17. Yao, B.; Hu, P.; Zhang, M.; Wang, S. Artificial bee colony algorithm with scanning strategy for the periodic vehicle routing problem. *Simulation* **2013**, *89*, 762–770. [[CrossRef](#)]
18. Karaboga, D.; Akay, B. A survey: Algorithms simulating bee swarm intelligence. *Artif. Intell. Rev.* **2009**, *31*, 61–85. [[CrossRef](#)]
19. Creutz, M. Microcanonical monte carlo simulation. *Phys. Rev. Lett.* **1983**, *50*, 1411. [[CrossRef](#)]
20. Bansal, J.C.; Sharma, H.; Jadon, S.S. Artificial bee colony algorithm: A survey. *Int. J. Adv. Intell. Paradig.* **2013**, *5*, 123–159. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.