An Evolutionary Algorithm for the Vehicle Routing Problem with Drones with Interceptions

Carlos Pambo
Department of Industrial Engineering
Faculty of Engineering
Stellenbosch University
24588628@sun.ac.za

Jacomine Grobler
Department of Industrial Engineering
Faculty of Engineering
Stellenbosch University
jacomine.grobler@sun.ac.za

Abstract

The use of trucks and drones as a solution to address *last-mile* delivery challenges is a new and promising research direction explored in this paper. The variation of the problem where the drone can intercept the truck while in movement or at the customer location is part of an optimisation problem called the vehicle routing problem with drones with interception (VRPDi). This paper proposes an evolutionary algorithm (EA) to solve the VRPDi. The paper demonstrates a metaheuristic strategy by applying an evolution-based algorithm to solve the VRPDi. In this variation of the VRPDi, multiple pairs of trucks and drones need to be scheduled. The pairs leave and return to a depot location together or separately to make deliveries to customer nodes. The drone can intercept the truck after the delivery or meet up with the truck at the following customer location. The algorithm was executed on the travelling salesman problem with drones (TSPD) datasets by Bouman *et al.* (2015), and the performance of the algorithm was compared by benchmarking the results of the VRPDi against the results of the VRP of the same dataset. This comparison showed improvements in total delivery time between 39% and 60%. Further detailed analysis of the algorithm results examined the total delivery time, distance, node delivery scheduling and the degree of diversity during the algorithm execution. This analysis also considered how the algorithm handled the VRPDi constraints. The results of the algorithm were then benchmarked against algorithms in Dillon *et al.* (2023) and Ernst (2024). The latter solved the problem with a maximum drone distance constraint added to the VRPDi. The analysis and benchmarking of the algorithm results showed that the algorithm satisfactorily solved 50 and 100-nodes problems in a reasonable amount of time, and the solutions found were better than those found by the algorithms in Dillon *et al.* (2023) and Ernst (2024) for the same problems. However, the algorithm performance deteriorated considerably as the number of nodes in the problems increased. This deterioration was in terms of the quality of the solution and the computation time required to solve the problem.

1 Introduction

Over the past few years, there has been a massive adoption of online shopping. Several factors contributed to that reality, such as increased access to internet services, the rise in smartphone users, and companies increasingly looking to decrease the time between consumers ordering products and these orders being delivered. These factors have all contributed to the increase in the adoption of online shopping (e-commerce) as a standard method of shopping. This adoption has led to delivery companies expanding their delivery networks and looking for ways to optimise delivering goods from warehouses to customers. This last leg of the delivery network, known as last-mile delivery operations/scheduling, can be characterised as a vehicle routing problem (VRP) optimisation problem.

As a solution to the last-mile scheduling challenge, this paper develops an evolutionary algorithm (EA) to address the vehicle routing problem with drones with interception (VRPDi), where the use of drones is included to support the transportation of goods, where a combination of vehicles and drones can deliver goods, where the drone can intercept the vehicle mid-way through its next delivery node. Goods can be reloaded to the drone, enabling the deliveries to be completed more effectively by the drones and the vehicles. VRPDi introduces new complexities to the VRP, requiring coordination and synchronisation between vehicles to optimise the routing and scheduling of both vehicles and drones, as well as the time and energy needed for intercepting and transferring packages from the vehicle to the drone. The VRPDi has attracted significant research attention in recent years due to such companies' need for faster and more efficient last-mile delivery services.

The VRPDi is a variant of the well-known VRP that includes using drones with interception capabilities, that is, return to the vehicle from where they are launched after the delivery. In contrast, the vehicle is stationary or in transit to the next delivery customer to support transporting goods. The VRP can be categorised as a combinatorial optimisation problem in operations research, where the aim is to minimise an objective function. This objective function could be the total distance travelled, the total delivery time required, and the total cost of the scheduling operation. In the instance of VRPDi, the distance travelled by the drones from the vehicle where it is launched to the customer location and its return to the launching vehicle also counts towards the problem's total cost, subject to various constraints such as time windows and the capacity of the vehicles.

Adding drones to the VRP creates new flexibility and efficiency in the transportation system. Drones can deliver goods directly to customers, bypassing road congestion and thus reducing delivery times. They can also access remote or difficult-to-reach locations, such as islands or mountainous areas, where the accessibility for ground vehicles may be limited. Also, drones are considered a green form of transportation. Solving the VRPDi represents a significant challenge due to the complexity of integrating drones into the existing VRP frameworks. It is a very active research area in engineering, transportation, operations research, and computer science. Researchers have explored various approaches, including heuristic algorithms (Daknama and Kraus, 2017; Schermer *et al.*, 2018; Di Puglia Pugliese *et al.*, 2021; Rios *et al.*, 2021); metaheuristics algorithms (Braekers *et al.*, 2016; Schermer *et al.*, 2019b; Pugliese *et al.*, 2020; Rios *et al.*, 2021; Ernst, 2021); and exact methods (Braekers *et al.*, 2016; Di Puglia Pugliese *et al.*, 2021) to find efficient solutions to the VRPD. Developing efficient and effective VRPD algorithms can significantly improve the performance and cost-effectiveness of *last-mile* delivery operations.

2 Literature Review

The introduction of drones in collaboration with trucks to perform delivery tasks resulted in the creation of the truck drone routing problem (TDRP), which is a variation of the classical VRP, which was first introduced in Dantzig and Ramser (1959) and has been an intensively researched over the decades since. Contrasting both VRP and TDRP is the introduction of drones in the latter and the inherited need for the solution to simultaneously cater to the collaboration between drones and trucks. Early research on the TDRP aimed to address solutions for the travelling salesman problem with drones (TSPD), a simplified version of the TDRP consisting of only one truck. In Murray and Chu (2015), two classes of the TSP were studied: the flying sidekick travelling salesman problem (FSTSP) and the parallel drone scheduling travelling salesman problem (PDSTSP). Both problems assume that a truck or a drone serves a customer. In the FSTSP, a drone collaborates with a truck to serve different customers, is dispatched from the truck to make deliveries and is collected by the truck to restore resources such as batteries and cargo. In the PDSTSP, the truck and drone work independently to serve the customers. Figure 1 illustrates both problems.

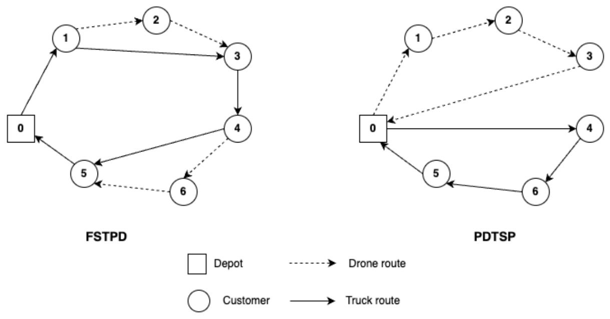


Figure 1: FSTSP and PDSTP Murray and Chu (2015).

A variation of TDRP, which has multiple trucks, referred to as a vehicle routing problem with drones (VRPD), was first introduced by Poikonen *et al.* (2017). VRPD consists of a fleet of trucks carrying a certain number of drones to deliver packages to customer nodes. Wang *et al.* (2017) researched using multiple trucks and drones to deliver packages to customers in a VRPD scenario. Each customer could only request one package at a time and could only be visited once. The trucks and drones began and ended their journeys at the depot, and the drone could be launched either from the depot or from the truck. The truck and drone could only meet at the truck's next customer's location, and the truck that launched the drone had to retrieve it. Whichever vehicle arrived at the customer's location first waited for the other. The focus of this paper is on the VRPD, studied in Wang *et al.* (2017); Poikonen *et al.* (2017). In this variation of the problem, a drone is launched from the truck at a customer node and re-joins the same truck it was launched from at the next customer node to be visited by the truck. The VRPD study by Moremi (2022) introduces the ability of the truck to intercept the drone at an intermediate point after the drone has completed its delivery. Different variants of the VRPD exist in the literature. These variants consider different combinations of settings for the truck-drone pair and objectives for the system. Objectives include minimising energy costs, tour time, delivery cost, number of drones in the system, and total travel costs. Different solution methods have addressed these objectives (Wang *et al.*, 2017; Wang and Sheu, 2019; Schermer *et al.*, 2019a; Sacramento *et al.*, 2019).

The VRPDi in this study is defined in a graph $\mathbf{G} = (\mathbf{N}, \mathbf{A})$. Where $\mathbf{N}$ represents the node-set, containing the depot node and a set of customer nodes $\mathbf{l} = \{l_1, l_2, \ldots, l_{n-1}, l_n\}$, and $\mathbf{A} = \{(i,j)|i,j \in N, i \neq j\}$ represents the arc set. For convenience in notation, o^s and o^t, represent the origin and termination depot, respectively. The problem contains a fleet of m homogeneous trucks, each carrying one drone. The trucks travel at a constant speed of v_t and take ω_i time to perform the delivery. The drones travel at a constant speed of v_d and take σ_i time to perform the delivery. The truck and the drone are initially located at the origin depot, ready to serve the customers. For simplicity, this paper does not adopt a docking or transfer hub to land drones, which is preferred in the drone delivery industry (Wang and Sheu, 2019). The station primarily intends to store and maintain backup and support drones for landing purposes. During pilot experiments, drones can take off from any location, but they can only land at designated docking hubs or depots, not at the customer's location. This is because drone landings require specific conditions, such as ample space and a particular docking device that can communicate with the drone to ensure a precise and secure landing. Due to the above conditions for drone landing not being easily reproducible at customer locations, in addition to safety and privacy concerns, this paper opts for drone deliveries to be performed via airdrop instead of landing at the customer node. A truck can be loaded up to its maximum capacity (C) units of customer parcels. Each truck in the system can carry a maximum of one drone. Each drone in the system can fly from the truck carrying it located at any of the locations in **N**, with a customer parcel.

The system does not impose the maximum flying duration of a drone as T^D, meaning that there is no restriction in terms of from which node a drone can fly to perform a delivery as long that specific node has not been visited by a truck or a drone previously. Every time the drone returns from delivery, it is fitted with new batteries, and the next parcel is loaded. These operations are very fast and, therefore, have no impact on the total cost.

It's worth noting that in case a truck finishes its current delivery, and has to drive to the next customer location it will visit, while its drone is also performing a delivery at another node, the truck has two options, taking its speed and drone speed, its current location and the drone location into account. The first is to calculate the time it will require to drive to the position of the possible intermediate interception location, between its current location and its next customer node location, plus the waiting time for the drone to arrive. The second option is to calculate the time it will require to drive to its next customer location, perform the delivery and wait for the drone to arrive from its delivery at that customer location. Whichever of the options requires less time becomes the next action the truck will perform. The mathematical formulation of the above is explained in detail in the next section. The objective of this VRPDi is to minimise the total delivery time. Total delivery time is represented by the maximum amount of time taken by the slowest of the truck-drone pair to finalise all of its customer deliveries. This delivery time of the truck-drone pair is represented by the sum of the time required by the truck and the drone to perform their deliveries. Figure 2 depicts the VRPDi. The example problem consists of three truck-drone pairs, one depot, and eleven nodes to be serviced. The first truck visits nodes 2 and 3, the second one visits nodes 4, 6, and 7, and the third one visits nodes 11 and 9. The first drone visits node 1, the second drone visits node 5, and the third drone visits node 10. Interception points are black circles, solid lines represent truck routes, and dotted lines represent drone routes. VRPDi has forbidden moves such as launching a drone at an intermediate location or only allowing one drone delivery between truck deliveries. The drone can only fly back to the truck from which it was launched.

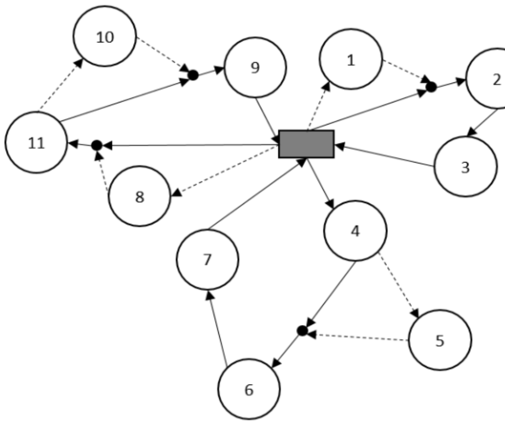


Figure 2: Vehicle routing problem with drones with interceptions

2.1 Mathematical Formulation

The vehicle routing problem (VRP) seeks to find optimal delivery or collection routes from a depot to customers while considering constraints. The problem was first introduced as the truck dispatching problem (TDP) by Dantzig and Ramser (1959). While exact methods are used to solve VRP optimally, heuristic methods like the Clark and Wright savings algorithm are used to solve large VRPs (Clarke and Wright, 1964). Over time, VRPs became more complex, resulting in different VRP variants. Examples of these different variations include the vehicle routing problem with time windows (VRPTW) (Desrochers *et al.*, 1992), the vehicle routing problem with pick-ups and deliveries (VRPPD) (Min, 1989), and the multi-depot vehicle routing problem (Wren and Holliday, 1972). The capacitated vehicle routing problem (CVRP), or traditional VRP, involves vehicles with limited capacities (C) servicing customers scattered across a geographic area from a central depot. Figure 3 shows a CVRP example. The black square represents the depot, while the circle symbols are customer nodes spread across an $x - y$ Cartesian plane. The dashed and solid lines denote the routes of the two vehicles. VRPs aim to determine the minimum distance or time to service all customer nodes while ensuring that each delivery route does not exceed the vehicle's capacity. When the vehicles' capacities differ, the problem becomes a varying fleet VRP.

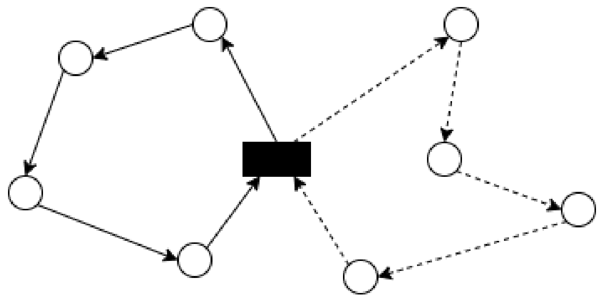


Figure 3: Example of a VRP routing solution

The VRP is defined on a graph (**G**), where $\mathbf{G} = (\mathbf{V}, \mathbf{A})$, and $\mathbf{V} = \{0, ..., n\}$. Node 0 is the depot, and each customer location j has a known demand d_j.

Equation 1 represents a set of edges. Each edge connecting nodes i and j has a non-negative cost c_{ij} associated with it, which is the cost of travelling from node i to node j. The asymmetric VRP has directed arcs (**A**), while the symmetric VRP has undirected arcs (**E**). Each customer in a set **S** has a demand $d(\mathbf{S})$ that equals the sum of all customers' demands in **S**, denoted by d_j. K vehicles have a capacity of C. The variable x_{ij} is a binary variable equal to 1 if edge (i, j) is serviced by a vehicle in the solution. Otherwise, x_{ij} is 0.

$$\mathbf{A} = \{(i,j) : i,j \in \mathbf{V}\}, \qquad i \neq j \tag{1}$$

For illustration, Equations 2 to 8 describe a two-index vehicle flow formulation of an asymmetrical VRP with specified assumptions (Laporte, 1992):

- Every vehicle's route begins and concludes at the depot.
- Every customer node, denoted as j in the set **V** except for node 0, is visited only once.
- The total demand of customers visited by each vehicle should not exceed its capacity, denoted by C.

$$\text{Minimise Z} = \sum_{i<j}^{n} c_{ij} x_{ij} \tag{2}$$

Subject to:

$$\sum_{i \in \mathbf{V} \setminus \{j\}} x_{ij} = 1 \quad \forall j \in \mathbf{V} \setminus \{0\} \tag{3}$$

$$\sum_{j \in \mathbf{V} \setminus \{i\}} x_{ij} = 1 \quad \forall i \in \mathbf{V} \setminus \{0\} \tag{4}$$

$$\sum_{i \in \mathbf{V} \setminus \{i\}} x_{i0} = K \tag{5}$$

$$\sum_{j \in \mathbf{V} \setminus \{j\}} x_{0j} = K \tag{6}$$

$$\sum_{i \notin \mathbf{S}} \sum_{j \in \mathbf{S}} x_{ij} \geq r(\mathbf{S}) \quad \forall \mathbf{S} \subseteq \mathbf{V} \setminus \{0\}, \ \mathbf{S} \neq \phi \tag{7}$$

$$x_{ij} \in 0,\ 1 \quad \forall i,j \in \mathbf{V} \tag{8}$$

- The goal of the function labelled as 2 is to minimise the distance or travel time for each vehicle.
- Constraints 3 and 4 are constraints that ensure each vertex has one arc entering and leaving it.
- The requirements for the depot vertex's degree are outlined in constraints 5 and 6.
- Constraints 7 are constraints that eliminate subtours. $r(\mathbf{S})$ is the lower bound on the number of vehicles needed to visit all vertices in **S**.

In Moremi (2022), the vehicle routing problem with drone with interception (VRPDi), was introduced with the following characteristics:

To determine the time and position of the drone and the truck, Equation 10 was utilised. The equation calculates the interception time. This equation calculates the time taken by the drone to move from node j to the truck's location $\mathbf{pI}^j$. The vector $\mathbf{c}^j$ is a 2-dimensional representation of the x and y coordinates of node j, whereas $d(\mathbf{c}^i, \mathbf{c}^j)$ is the distance between node j and the position of a truck at the beginning of a drone launch at node i.

To calculate the time from node j to the interception point $\mathbf{pI}^j$, $\mathbf{TI}^j$ is used, and it is calculated in Equation 9, where $\mathbf{c}^i - \mathbf{c}^j$ represents the difference between the x and y coordinates of nodes j and i as a 2-dimensional vector.

$$\mathbf{TI}^j = \frac{-2(\mathbf{c}^i - \mathbf{c}^j).(v_t^{ik}) \pm \sqrt{(2((\mathbf{c}^i - \mathbf{c}^j).(\mathbf{v}_t^{ik}))^2 - (4(v_d^2 - v_t^2)(-d(\mathbf{c}^i, \mathbf{c}^j)^2)))}}{2(v_d^2 - v_t^2)} \tag{9}$$

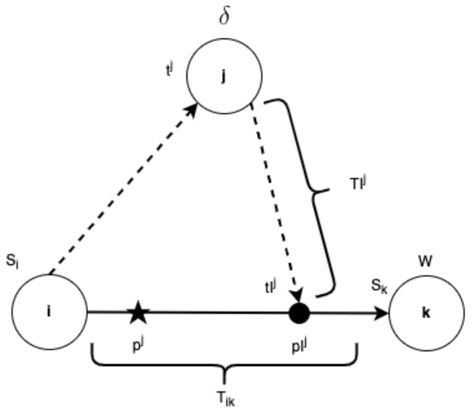


Figure 4: Truck-drone subtour Moremi (2022).

The interception position can then be calculated as follows:

$$\mathbf{pI}^j = \mathbf{c}^i + (\mathbf{v}_t^{ik}).(TI^j + \sigma + (T^j - s_i)) \tag{10}$$

In this situation, the drone has two choices: intercept the truck before it gets to node k or fly straight to node k. If the drone cannot intercept the truck before it reaches node k, it will fly directly to the node's location, and the truck will wait for the drone. The total time it takes for the truck to travel from node i to k is known as T_{ik}. The total time it takes for the truck to travel from node i to the interception point $\mathbf{pI}^j$ is equal to $(tI^j - s_i)$, where tI^j is the time when the drone and the truck meet after the drone has delivered at node j and s_i is the time when the truck arrives at customer i for delivery. This time is equivalent to the drone's travel time from i to the interception point $\mathbf{pI}^j$. If the truck arrives at node location k without intercepting the drone, it must wait for its arrival. The waiting time, W, is calculated as $T_{ijk} - T_{ik}$, where T_{ijk} is the total drone travel time from node i to k and is only relevant when $t_{ijk} > T_{ik}$. The total time to finish the deliveries of nodes i and j and travel to k is $T_{ijk} = max(T_{ik}, t_{ijk})$. The arrival of the last vehicle at node k determines T_{ijk}.

Moremi (2022) proposed the following mathematical models with the assumption that a drone delivery must be scheduled between each truck delivery.The model has the following input parameters:

- n denotes the number of nodes to be serviced, including a depot at node 0
- T_{ijk} denotes the time required to attend to nodes i and j, and travel to k. The truck moves between nodes i and k, while the drone caters to node j in between.

In the above model, the decision variables are:

- The variable u_i is employed to remove sub-tours.

$$x_{ijk} = \begin{cases} 1, & \text{If the truck services node } i \text{ before node } k, \\ & \text{and the drone services node } j \text{ in between} \\ 0, & \text{Otherwise} \end{cases}$$

$$b_j = \begin{cases} 1, & \text{If the drone services node } j \\ 0, & \text{If the truck services node } j \end{cases}$$

$$z_{ik} = \begin{cases} 1, & \text{If the truck travels from node } i \text{ to node } k \\ 0, & \text{Otherwise} \end{cases}$$

$$min \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n} x_{ijk}T_{ijk} \tag{11}$$

Subject to:

$$z_{ii} = 0 \quad \forall i \in \{1, ..., n\} \tag{12}$$

$$x_{iii} = 0 \quad \forall i \in \{1, ..., n\} \tag{13}$$

$$z_{ik} = \sum_{j=1}^{n} x_{ijk} \quad \forall i, k \in \{1, ..., n\} \tag{14}$$

$$b_j = \sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} \quad \forall j \in \{1, ..., n\} \tag{15}$$

$$\sum_{i=1}^{n} z_{iq} - \sum_{k=1}^{n} z_{qk} = 0 \quad \forall q \in \{1, ..., n\} \tag{16}$$

$$\sum_{k=1}^{n} z_{ik} = 1 - b_i \quad \forall i \in \{1, ..., n\} \tag{17}$$

$$\sum_{k=1}^{n} z_{ik} = 1 - b_k \quad \forall k \in \{1, ..., n\} \tag{18}$$

$$\sum_{j=1}^{n} x_{ijk} \leq 1 \quad \forall i, k \in \{1, ..., n\} \tag{19}$$

$$\begin{gathered} u_i + z_{ij} \leq u_j + (n-1)(1 - z_{ij}) \\ \forall i, j \neq 1 \in \{1, ..., n\} \end{gathered} \tag{20}$$

$$u_1 = 0 \tag{21}$$

$$x_{ijk}, b_i, z_{ik} \in \{0, 1\} \quad xi, j, k \in \{1, ..., n\} \tag{22}$$

- The objective of Equation 11 is to reduce the drone's and truck's overall travel time.
- Constraints 12 and 13 simplify the model by ensuring that the truck and drone cannot return to a previously visited node.
- Constraint 14 links z_{ik} and x_{ijk}.
- Constraint 15 links b_j and x_{ijk}.
- Constraint 16 ensures that if a delivery is made at a node, the vehicle used for delivery must depart from that same node.
- Constraint 19 ensures that only one drone delivery can occur between two truck deliveries.
- Constraints 20 and 21 eliminate any sub-tours.
- Constraint 22 restricts the decision variables to binary values.

3 Algorithm Description

Algorithm 3.1 provides a pseudo-code description of the algorithm used to solve the VRPDi. The algorithm requires the following input control parameters: the number of nodes (n) and the number of iterations the algorithm runs. These iterations are also called generations (g), the size of the population (P_{size}), a list of a 2-dimensional vector of the x and y coordinates of the nodes (**L**), a matrix of the Euclidean distances between the nodes (**D**), obtained using Equation 23, for the distance between nodes i and j, that contain the Cartesian coordinates (c_x^i, c_y^i) and (c_x^j, c_y^j), respectively:

$$d(\mathbf{c}^i, \mathbf{c}^j) = \sqrt{(c_x^i - c_x^j)^2 + (c_x^i - c_y^j)^2}\,. \tag{23}$$

The population's fitness in each iteration is evaluated by converting the genotype solution into a route

```
Algorithm 3.1: Evolutionary Algorithm to solve the VRPDi
Input  : Size of population ($P_{size}$); Number of elite individuals ($E$); Probability of mutation ($\gamma$);
         Number of iterations ($g$); Number of truck-drone pairs ($N_{td}$); Number of nodes ($n$);
         Nodes location coordinates (L); Nodes distances matrix (D)
Output: Candidate solution X
1  Generate random $P_{size}$ number of solutions and save them in P;
2  Evaluate the fitness of each solution in P;
3  Save the best solution in P into X;
4  for $i = 1$ to $g$ do
5      Select the best $E$ solutions in P and save them in $\mathbf{P}_s$;
6      Initialise $\mathbf{P}_n$ as empty list;
7      for $j = 1$ to $P_{size}$ do
8          Select a random solution in $\mathbf{P}_s$ and save it in $\mathbf{X}_A$;
9          Select a random solution in $\mathbf{P}_s \notin \mathbf{X}_A$ and save it in $\mathbf{X}_B$;
10         Perform crossover of $\mathbf{X}_A$ and $\mathbf{X}_B$ and save it in $\mathbf{X}_C$;
11         Perform mutation of $\mathbf{X}_C$ based on $\gamma$ and save it in $\mathbf{X}_C^{'}$;
12         if $X_C^{'}$ is infeasible then
13             Update $\mathbf{X}_C^{'}$ with repaired $\mathbf{X}_C^{'}$;
14         Save $\mathbf{X}_C^{'}$ into $\mathbf{P}_n$
15     Evaluate the fitness of each solution in $\mathbf{P}_n$;
16     if Fitness of X < Fitness of best solution in $\mathbf{P}_n$ then
17         Update X with best solution found in $\mathbf{P}_n$;
18     Update population list P with the current generation of solutions in $\mathbf{P}_n$;
19 return X
```

to follow using *genotype-phenotype mapping*, as described in Holland (1973) and Mathew (2012). For minimisation problems, as is the case with the proposed EA in this paper, to generate non-negative values in all the cases and to reflect the relative fitness of individual solutions, it is necessary to map the underlying objective function of the problem to a fitness function form. One such transformation commonly adopted for fitness function mapping is to use the inverse of objective function value, thus resulting in $F(i) = \frac{1}{z(i)}$ to get the fitness value of $F(i)$. The transformation in the equation does not alter the location of the minimum. However, it converts a minimisation problem to an equivalent maximisation problem, where the greater the value of $F(i)$, the more chances that the solution has to be an optimal solution to the problem. In the equation above, $z(i)$ is the objective function value of the i^{th} solution. This fitness procedure implementation is described in Algorithm 3.2.

```
Algorithm 3.2: Fitness evaluation procedure
Input  : List of population (P)
Output: List of population with fitness values ($\mathbf{P}^{'}$)
1  Initialise $\mathbf{P}^{'}$ with P;
2  for $i = 1$ to Length of P do
3      Update the Fitness of $\mathbf{P}_i^{'}$ with ($1/Total\ time$ of $\mathbf{P}_i$);
4  Sort $\mathbf{P}^{'}$ by fitness in ascending order;
5  return $\mathbf{P}^{'}$
```

The algorithm utilises the concept of elitism in selecting individual solutions for the next generation in the evolutionary process, as described in Laumanns *et al.* (2000). Algorithm 3.3 describes this selection process. The primary purpose of using elitism in an EA is to keep the reference for promising areas of the search space across the generations. In practical terms, this enables the continuous exploitation of these promising areas, where the local or global optimum can be found. Elitism also ensures the best individual solutions are found, considering the entire processing of an EA in the last generation created, which would be the final solution.

```
Algorithm 3.3: Population selection procedure
Input  : List of population (P); Number of elite individuals (E)
Output: List of individuals to be used for breeding (S)
1  Initialise $\mathbf{P}^{'}$ with empty list;
2  Initialise $l$ with the length of P;
3  for $i = 1$ to E do
4      Insert $\mathbf{P}_i$ to the list $\mathbf{P}^{'}$;
5  for $i = 1$ to $(l - E)$ do
6      Initialise $R$ with a random decimal value (between 0 and 100);
7      Initialise $T$ with 0;
8      for $j = 1$ to $l$ do
9          $T$ equals to $\sum_{j=1}^{i}$(Fitness of $\mathbf{P}_j$);
10         if $R < T$ then
11             Insert $\mathbf{P}_j$ to the list $\mathbf{P}^{'}$;
12 return $\mathbf{P}^{'}$
```

The crossover operator used by the algorithm is described in Algorithm 3.4. As the EA regards the visited nodes sequence and the nodes delivery type as two different components of its genotype, as described in Goldberg and Lingle (2014), a partially mapped crossover (PMX) based operator is adopted as the crossover operator in this algorithm implementation. The mutation operator adopted for this implementation is described in Algorithm 3.5.

Algorithm 3.4: Crossover procedure

```
Input  : Parent A (XA); Parent B (XB); Number of nodes ($n$);
Output: Resulting child from the crossover operation (X)
1 Initialise X with empty solution genotype;
2 Initialise $\mathbf{G}_A$ with a random decimal value (between 0 and 1) $* n$;
3 Initialise $\mathbf{G}_B$ with a random decimal value (between 0 and 1) $* n$;
4 for $i = min(\boldsymbol{G}_a, \boldsymbol{G}_b)$ to $max(\boldsymbol{G}_a, \boldsymbol{G}_b)$ do
5     Insert $\mathbf{XA}_i$ bit into $X$;
6 for $i = 1$ to Length of XB) do
7     if bit $i \in$ XB AND bit $i \notin X$ then
8         Insert $\mathbf{XA}_i$ bit into X;
9 return X
```

Algorithm 3.5: Mutation procedure

```
Input  : Solution (X); Change of mutation ($\gamma$); Number of nodes ($n$);
Output: Mutated solution ($\mathbf{X}'$)
1 Initialise $\mathbf{X}'$ with X;
2 for $i = 1$ to $n$ do
3     Initialise $R$ with a random decimal value (between 0 and 1);
4     if $R < \gamma$ then
5         Initialise $S$ with value of $i$;
6         while $S == i$ do
7             Update $S$ with a random decimal value (between 0 and 1) $* n$;
8         Initialise $\mathbf{G}_A$ with $\mathbf{X}_i^i$;
9         Initialise $\mathbf{G}_B$ with $\mathbf{X}_S^i$;
10        Update $\mathbf{X}'$ at index $S$ with $\mathbf{G}\S_A$;
11        Update $\mathbf{X}'$ at index $i$ $\mathbf{G}_B$;
12 return $\mathbf{X}'$
```

The repair mechanism adopted for this implementation is described in Algorithm 3.6 for instances where two consecutive drone nodes are scheduled, which does not conform to the VRPDi model constraints. The procedure was developed for repairing infeasible solutions based on what Mitchell *et al.* (2003), FitzGerald and O'Donoghue (2008) and Ernst *et al.* (2023b) proposed. The repair mechanism changes the second consecutive drone node to a truck node without modifying the delivery sequence, meaning that after the repair procedure, the algorithm has found a solution different from the one it initially found. The resulting repaired solution may or may not already be part of the list of candidate individual solutions in that generation of the algorithm.

Algorithm 3.6: Solution repair procedure

```
Input  : Solution (X)
Output: Repaired solution ($\mathbf{X}'$)
1 Initialise $\mathbf{X}'$ with X;
2 for $i = 1$ to (length of X) $- 1$ do
3     if Delivery of $X_i' ==$ Drone AND Delivery of $X_{i+1}' ==$ Drone then
4         Update Delivery of $\mathbf{X}_i'$ to Truck;
5 return $\mathbf{X}'$
```

Equation 24 is used to compute the number of truck-drone pairs (N_{td}) for a given dataset **S**. Where $\theta(\mathbf{S})$ represents the total demand for the nodes in dataset **S**, the value of $\theta(\mathbf{S})$ is given by $\theta(\mathbf{S}) = \sum_{i\in\mathbf{S}} \theta_i$, for each node i in the dataset **S**. C denotes vehicle capacity. If the computed value of N_{td} is not an integer, the value is rounded up to the nearest integer. After the N_{td} is computed, each truck-drone pair is assigned several nodes to be delivered equal to or smaller than the vehicle capacity C.

$$N_{td}(\mathbf{S}) = \frac{\theta(\mathbf{S})}{C} \; N_{td} \in \mathbb{Z}^+ \tag{24}$$

4 Results

The work presented in this paper investigated the feasibility of applying an evolutionary algorithm to solve the vehicle routing problem with drones with interceptions (VRPDi). The algorithm is evaluated on ten datasets from Bouman *et al.* (2015). The selected datasets feature customer nodes that are uniformly distributed across the $x - y$ Cartesian plane. These datasets are summarised in Table 1.

The algorithm parameters were set as follows:

- Algorithm-specific parameters include the population size, the elitism rate, the mutation rate, and the maximum number of generations.
- Problem-specific parameters include the number of nodes (n), the truck speed (v_t), the drone speed (v_d), the truck delivery time (ω_i), the drone delivery time (σ_i) and the number of truck-drones pairs to be scheduled (N_{td}). Table 2 summarises

Datasets	**Nodes**
Uniform-71-n50	50
Uniform-72-n50	50
Uniform-73-n50	50
Uniform-91-n100	100
Uniform-92-n100	100
Uniform-93-n100	100
Uniform-1-n250	250
Uniform-2-n250	250
Uniform-5-n500	500
Uniform-6-n500	500

Table 1: Datasets for algorithm evaluation from Bouman *et al.* (2015)

the values assumed for these variables. The truck speed (v_t) and the drone speed (v_d) parameters for each of the problems are set based on values from Bouman *et al.* (2015).

- Truck capacity (C) was set to 40 for datasets with less than or equal to 100 nodes and 100 for datasets with more than 100 customer nodes. The dataset's demand for each node i was set to 1.

Parameters	**Values**
Crossover	PMX
Elitism rate (%)	15
Encoding	Metameric
Fitness function	Linear ranking
Max. generations	1000
Mutation operation	Bit inversion
Mutation rate (%)	30
Reinsertion	Fitness-based
Population size	150
Selection	Elitism

Table 2: Algorithm parameters

Data	**VRP**			**VRPDi**			**Performance**		
	Time	**Distance**	**CPU**	**Time**	**Distance**	**CPU**	**Time**	**Distance**	**CPU**
Uniform-71-n50	58.61	712.31	**5.02**	**37.63**	**662.34**	7.53	55%	7%	-33%
Uniform-72-n50	60.46	819.87	**4.66**	**43.19**	**562.46**	10.75	39%	45%	-56%
Uniform-73-n50	58.40	770.00	**5.13**	**39.03**	**509.72**	8.00	49%	51%	-35%
Uniform-91-n100	82.03	1683.95	**10.34**	**52.75**	**1156.19**	11.08	55%	45%	-6%
Uniform-92-n100	80.44	1634.95	**6.92**	**51.69**	**1107.40**	12.40	55%	47%	-44%
Uniform-93-n100	81.98	1681.81	**7.13**	**51.17**	**1082.89**	11.72	60%	55%	-39%
Uniform-1-n250	220.34	6566.14	**27.86**	**144.00**	**3933.12**	29.14	53%	66%	-4%
Uniform-2-n250	224.39	6691.53	38.61	**149.70**	**4013.30**	**24.75**	46%	69%	24%
Uniform-5-n500	260.68	9730.62	149.37	**154.23**	**6662.66**	**119.68**	45%	68%	33%
Uniform-6-n500	245.24	8842.61	168.39	**149.23**	**6382.95**	**136.56**	64%	38%	23%

Table 3: VRP vs VRPDi performance summary

4.1 Algorithm Benchmarking

The results of the algorithm developed in this paper were compared to the nearest neighbour heuristic for initial solutions self-adaptive neighbourhood search differential evolution (NNHis SaNSDE) algorithms in Dillon *et al.* (2023) and Ernst (2024) for scheduling multiple truck and drone algorithms. The NNHis SaNSDE algorithm is a population-based metaheuristic EA. The algorithm was introduced in Ernst *et al.* (2023a). The algorithm is built on the SaNSDE-based algorithm introduced in Yang *et al.* (2008), combined with problem-specific nearest neighbour heuristics that introduce good solutions into the initial population to guide the search process. The delivery completion time for the VRPDi solutions of the algorithm in this paper against solutions from the NNHis SaNSDE I algorithm is summarised in Table 4. The VRPDi solutions with a maximum drone distance constraint of this paper against those of the NNHis SaNSDE II algorithm, are summarised in Table 5. The algorithms were compared in datasets with three distribution types: double-centred, single-centred and uniformly distributed. This distribution refers to the placement of the node $x - y$ coordinates along the Cartesian plane.

Data	**Nodes**	**No. trucks / Clusters**	**EA**	**NNHis SaNSDE I**
Doublecenter-61-n20	20	2	**40.09**	57.29
Doublecenter-61-n20	20	2	**40.09**	57.29
Doublecenter-91-n100	100	2	**91.78**	94.52
Doublecenter-3-n250	250	2	188.205	**166.12**
Doublecenter-3-n375	375	2	262.71	**214.52**
Doublecenter-5-n500	500	2	309.49	**240.86**
Singlecenter-61-n20	20	4	**15.13**	24.61
Singlecenter-100-n100	100	4	**31.78**	36.84
Singlecenter-1-n250	250	4	88.055	**59.46**
Singlecenter-1-n375	375	4	92.49	**62.17**
Singlecenter-6-n500	500	5	117.95	**77.49**
Uniform-61-n20	20	4	**10.76**	15.80
Uniform-91-n100	100	4	**47.66**	48.08
Uniform-8-n250	250	4	65.32	**48.09**
Uniform-1-n375	375	4	71.68	**57.43**
Uniform-8-n500	500	4	77.54	**66.62**

Table 4: VRPDi benchmarking of the EA vs NNHis SaNSDE I Dillon *et al.* (2023)

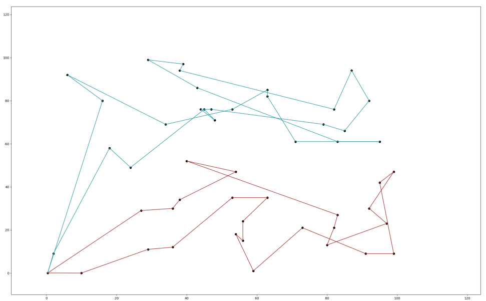
Data	**Nodes**	**Max. distance**	**Max. drone distance**	**No. trucks / clusters**	**EA**	**NNHis SaNSDE II**
Doublecenter-71-n50	50	822.02	616.51	2	**77.15**	87.08
Doublecenter-91-n100	100	751.58	563.69	3	**79.34**	86.56
Doublecenter-1-n250	250	772.27	579.20	3	228.43	**132.71**
Doublecenter-5-n500	500	896.52	672.39	5	221.60	**162.01**
Singlecenter-71-n50	50	327.35	245.52	2	51.51	**35.22**
Singlecenter-91-n100	100	451.54	338.66	3	81.01	**65.63**
Singlecenter-1-n250	250	478.34	358.75	3	161.91	**97.43**
Singlecenter-5-n500	500	546.21	409.66	5	181.81	**101.70**
Uniform-71-n50	50	249.41	187.06	2	**36.70**	38.17
Uniform-72-n50	50	256.03	192.02	2	40.96	**39.78**
Uniform-73-n50	50	253.04	189.78	2	**40.12**	41.48
Uniform-91-n100	100	263.07	197.30	3	**43.70**	46.84
Uniform-92-n100	100	270.17	202.62	3	**40.09**	40.66
Uniform-1-n250	250	266.60	199.95	3	80.38	**67.56**
Uniform-2-n250	250	259.27	194.45	3	79.09	**54.52**
Uniform-5-n500	500	276.50	207.37	5	87.43	**55.38**
Uniform-6-n500	500	74.57	205.93	5	79.31	**52.49**

Table 5: VRPDi benchmarking of the EA vs NNHis SaNSDE II Ernst (2024)

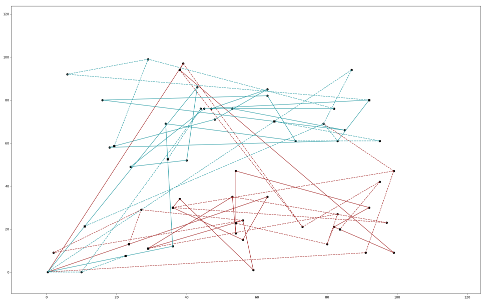
5 Discussion

The algorithm is evaluated over 30 independent runs on the datasets. This section only contains graphical representations of *Uniform-71-n50* dataset deliveries for the

VRP and VRPDi. The delivery tours for each of the trucks of the VRP and the trucks and drones of the VRPDi for the dataset are shown in Figure 5. The figure's continuous solid lines represent truck paths, the dotted lines represent drone paths, the black circles represent customer node locations, and the black squares represent interception points between trucks and drones. The above figure clearly shows that the solution obtained by the algorithm is feasible and complies with the problem constraints.

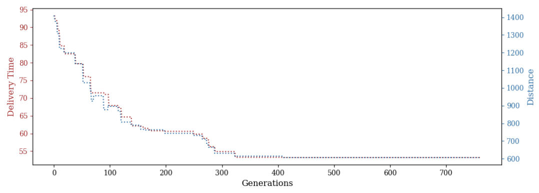


(a) VRP

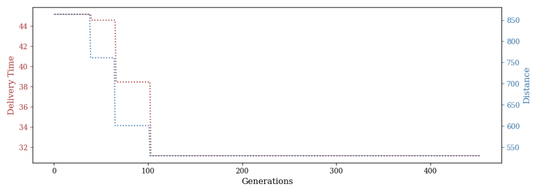


(b) VRPDi

Figure 5: Deliveries for the VRP and the VRPDi (Uniform-71-n50)



(a) VRP



(b) VRPDi

Figure 6: Best solutions for 50 nodes deliveries for VRP and VRPDi (Uniform-71-n50)

The algorithm parameter settings balance exploration and exploitation throughout the search process. The algorithm's mean and standard deviations for the delivery time and the distance are shown in Table 3. *Time* refers to the total time for completing all deliveries. *Distance* is the total distance travelled by trucks. *CPU* refers to the time in minutes the algorithm took to be completed for each of the runs. The VRPDi and VRP systems are compared using the same algorithm. Node deliveries are scheduled by a truck-drone assigning system, where nodes are randomly assigned to trucks and drones at first. As the algorithm progresses through the generations, nodes closer to each other are set using the Euclidean distance matrix mentioned in Section 3, satisfying the problem's above-mentioned requirements. After that, the deliveries for the same truck-drone pair are scheduled truck-only.

Comparing the results of the VRP to those of the VRPDi, it is noticeable that the latter yielded positive results when compared with the VRP system in all datasets. There was an improvement in delivery time ranging between 39% and 60%. Meanwhile, the distance improvements were between 7% and 69%. The executions also registered a deterioration in the algorithm running time (CPU) as the number of nodes in the problems increased. This deterioration ranged between 4% and 56%, with the most significant deterioration registered in the 50 and 100-node datasets. A complete summary of the results comparison of running the EA on the datasets in Table 1 for both VRP and VRPDi are shown in Table 3.

EAs often suffer from loss of diversity through premature convergence of the population, causing the search to be trapped in local optima (Zhu, 2003, 2004). The phenotype measure describes the unique fitness values in a given population ($\mathbf{P}$), divided by the population size (P_{size}); the standard deviation of the fitness value is given by the equation $stddev(\mathbf{P}) = \sqrt{\frac{\sum_{i=1}^{N}(f_i - \bar{f})^2}{N-1}}$, over the generations of the algorithm. Genetic diversity represents a crucial part of evolutionary exploration since an EA can only search the space offered by the genes available in the population. Figure 7 depicts a typical fitness environment that contains a converged population, the global optimum and several local optima. The location of the hypothetical average individual in the population is also illustrated in the figure (Ursem, 2002; Zhu, 2004). The unique fitness value and the standard deviation value can be used to determine the diversity variation in the solutions in the algorithm search space. The above standard deviation equation can also be employed to determine the mutation and crossover rates of the EA. Thus, the search diversity shown in Figures 8 confirms that sufficient diversity is maintained for the search process and that the algorithm does not converge to a single solution early during the search.

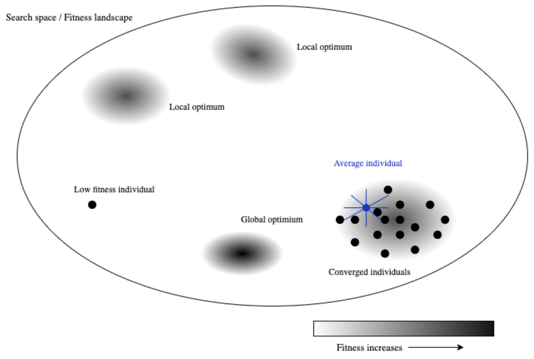
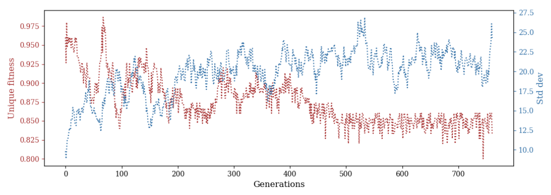
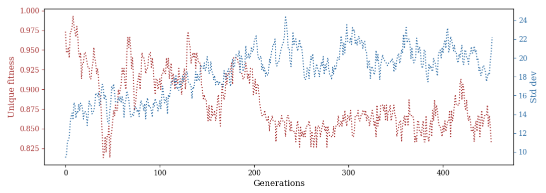


Figure 7: Search space depicting a converged population, global optimum, local optima, and the average individual's position (McGinley *et al.*, 2011)



(a) VRP



(b) VRPDi

Figure 8: Solutions search diversity for 50 nodes for VRP and VRPDi (Uniform-71-n50)

Results of non-parametric statistical tests to evaluate the significance of results obtained from the algorithm executions are summarised in Table 6. Each problem variation was compared to the effects of the other problem variation (VRP vs VRPDi) using a Mann-Whitney U test at 5% significance. Wins, draws, and losses were recorded for all comparisons of the evaluated problem variation. A win is recorded if the first combination significantly outperforms the second. A draw is recorded if no statistical difference is observed. A loss is recorded for the first problem variation if the second variation outperforms it. These tests show that the VRP statistically underperformed compared to the VRPDi ten times when using delivery time as a performance metric, with no draws and ten losses recorded. Similar results were obtained when total distance was used as the performance metric for the evaluation.

5.1 Benchmarking

The algorithm in Dillon *et al.* (2023) (referred to as NNHis SaNSDE I algorithm), and the algorithm in Ernst (2024) (referred to as NNHis SaNSDE II algorithm), use a cluster-first, route-second approach to solve the VRPDi. A maximum drone distance constraint is also used in the VRPDi variation in Ernst (2024). A drone can only travel a maximum distance equal to 75% of the maximum distance between two nodes in the dataset. This means that if a drone travels from node i to node k, to perform a delivery at node k, and the Euclidean distance between i and k is greater than the maximum drone distance of the dataset, the solution is deemed infeasible. Therefore, the repair procedure is triggered. This repair procedure involves changing the delivery type for node k from a drone to a truck (Mitchell *et al.*, 2003; Ernst *et al.*, 2023b). For comparison purposes, the number of truck-drones pairs to be scheduled (N_{td}) was set according to the number of clusters (k) for each of the problems being compared. These results show that the algorithm developed in this paper performed better than both NNHis SaNSDE algorithms for problems with 100 or fewer nodes, regardless of node uniformity distribution. The algorithm developed in this paper did not outperform either of the algorithms in datasets with more than 100 nodes.

	Time			Distance		
	Win	**Draw**	**Loss**	**Win**	**Draw**	**Loss**
VRP	0	0	**10**	0	0	**10**
VRPDi	**10**	0	0	**10**	0	0

Table 6: Hypothesis tests regarding delivery time and distance of the VRP and the VRPDi

6 Conclusions

This paper proposes an evolutionary algorithm to solve the vehicle routing problem with drones with interceptions (VRPDi). The literature has shown the TSPD and the VRPD, including all its different variations, to be complex problems to solve, specifically as the number of nodes increases. This paper then discussed the different mathematical formulations of the VRPD, showing that solving the model proposed in this is more complex than unravelling a VRP. There is additional complexity in adding drones to assist the trucks in completing the deliveries. This paper demonstrated the performance of the proposed algorithm to solve such problems. The evaluation of the results of the proposed algorithm included solutions for the VRP (truck-only) and the VRPDi (truck-drone pairing) variations of the vehicle routing problem, which were done so that they could be compared. The paper also shows that the total delivery time improvement from a VRP to a VRPDi solution was between 39% and 64%. The improvement in total distance was between 7% and 69%, from VRP to VRPDi. The paper then compared the performance of the algorithm developed to the NNHis SaNSDE I algorithm from Dillon *et al.* (2023) and the NNHis SaNSDE II algorithm from Ernst (2024).

The algorithm developed in this paper outperforms the NNHis SaNSDE I algorithm in datasets with 50 and 100 nodes. For the same datasets with more than 100 nodes, the algorithm of this paper is outperformed by the NNHis SaNSDE I algorithm. The comparison also showed that the NNHis SaNSDE II algorithm outperformed the algorithm developed in this paper in datasets with more than 100 nodes for the VRPDi problem variation with a maximum drone distance constraint. For the same problem variation, the algorithm developed in this paper outperformed the NNHis SaNSDE II algorithm in datasets with 100 nodes or fewer nodes. Through the above benchmarking, it becomes evident that the algorithm in this paper performed better than the other algorithms in all variations of the VRPDi reviewed in this paper for 50 and 100-node datasets regarding the quality of the solutions and the computation effort required to arrive at the solution. It also became evident that the algorithm developed in this paper became more computationally expensive as the number of nodes in the problem increased, requiring an average of 128 minutes to execute each run of the 500 node dataset, highlighting the fact that its solution quality decreased as the number of nodes in the datasets increased.

Given the increase in the demand for *last-mile* deliveries from central logistic locations and their customers, enterprises aim to optimise these scheduling and routing activities to be as efficient as possible. The scheduling and routing algorithm developed from this paper can be utilised in any environment where deliveries must be scheduled from a central location.

The contribution of this work includes the development of the first GA-based algorithm to solve this multi-vehicle, namely trucks and drones, scheduling and routing problem, the VRPDi. The paper contributes to the metaheuristics optimisation field. These contributions include applying evolutionary algorithm concepts to solve scheduling and routing optimisation problems. The paper also contributes to evolutionary computing, namely, the genetic algorithm, which has been tested for solving this single-objective routing problem. Given the emergence of the fourth industrial revolution, companies operating in the transportation industry can benefit immensely from optimisation research into scheduling and routing optimisation. The VRPDi problem is characterised by a delivery system comprised of trucks and drones. An EA was developed to solve this scheduling and routing optimisation problem, leading to several areas being identified for future research. These opportunities are aimed at improving the approach used by the algorithm to solve the problem, thus improving the algorithm's efficiency. In future work, we aim to utilise self-adaptive parameters in the algorithm to optimise control parameters for specific problem variations and increase the algorithm's efficiency by assigning nodes to trucks and drones using a cluster-first approach.

Abbreviations

The following abbreviations are used in this manuscript:

CVRP	Capacitated vehicle routing problem
EA	Evolutionary algorithm
FSTSP	Flying sidekick travelling salesman problem
GA	Genetic algorithm
NNHis	Nearest neighbour heuristic for initial solutions
PDSTSP	Parallel drone scheduling travelling salesman problem
PMX	Partially mapped crossover
SaNSDE	Self-adaptive neighbourhood search differential
TSP	Travelling salesman problem
VRP	Vehicle routing problem
VRPDi	Vehicle routing problem with drones with interceptions
VRPPD	Vehicle routing problem with pick-up and delivery
VRPTW	Vehicle routing problem with time windows

Author Contributions

Conceptualization: Pambo & Grobler; Methodology: Pambo; Resources: Pambo & Grobler; Writing - original draft: Pambo; Writing - review and editing: Pambo & Grobler. Both authors have read and agreed to the published version of the manuscript.

References

Bouman, P., Agatz, N. & Schmidt, M. (2015). Instances for the tsp with drone.

Braekers, K., Ramaekers, K. & Van Nieuwenhuyse, I. (2016). The vehicle routing problem: State of the art classification and review. *Computers & industrial engineering*, **99**, 300–313.

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

Daknama, R. & Kraus, E. (2017). Vehicle routing with drones. *arXiv preprint arXiv:1705.06431*.

Dantzig, G.B. & Ramser, J.H. (1959). The truck dispatching problem. *Management science*, **6**, 80–91.

Desrochers, M., Desrosiers, J. & Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations research*, **40**, 342–354.

Di Puglia Pugliese, L., Macrina, G. & Guerriero, F. (2021). Trucks and drones cooperation in the last-mile delivery process. *Networks*, **78**, 371–399.

Dillon, S., Ernst, R. & Grobler, J. (2023). Investigating alternative clustering algorithms for a cluster-first, route-second truck and drone scheduling algorithm. In *International Conference on Computational Science and Its Applications*, 147–162, Springer.

Ernst, R. (2021). *Efficiency study of various algorithms on the travelling salesperson problem with drone and with interceptions*. Master's thesis, Stellenbosch: University of Stellenbosch.

Ernst, R. (2024). *Evolutionary algorithms for routing problems with drones with interceptions*. Ph.D. thesis, Stellenbosch: Stellenbosch University.

Ernst, R., Grobler, J., Moremi, T. & Knoetze, F. (2023a). Differential evolution algorithms for the travelling salesman problem with drones with interceptions. *Submitted Comp. Ind. Eng.*.

Ernst, R., Moremi, T., Grobler, J. & Kaminsky, P.M. (2023b). A framework for the analysis of metaheuristics for the travelling salesman problem with drone with interceptions. In *International Conference on Computational Science and Its Applications*, 351–368, Springer.

FitzGerald, A. & O'Donoghue, D.P. (2008). Genetic repair for optimization under constraints inspired by arabidopsis thaliana. In *Parallel Problem Solving from Nature – PPSN X*, 399–408, Springer Berlin Heidelberg, Berlin, Heidelberg.

Goldberg, D.E. & Lingle, R. (2014). Alleles, loci, and the traveling salesman problem. In *Proceedings of the first international conference on genetic algorithms and their applications*, 154–159, Psychology Press.

Holland, J.H. (1973). Genetic algorithms and the optimal allocation of trials. *SIAM journal on computing*, **2**, 88–105.

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, **59**, 345–358.

Laumanns, M., Zitzler, E. & Thiele, L. (2000). A unified model for multi-objective evolutionary algorithms with elitism. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, vol. 1, 46–53, IEEE.

Mathew, T.V. (2012). Genetic algorithm. *Report submitted at IIT Bombay*, 53.

McGinley, B., Maher, J., O'Riordan, C. & Morgan, F. (2011). Maintaining healthy population diversity using adaptive crossover, mutation, and selection. *IEEE Transactions on Evolutionary Computation*, **15**, 692–714.

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

Mitchell, G.G., O'Donoghue, D., Barnes, D. & McCarville, M. (2003). Generepair-a repair operator for genetic algorithms.

Moremi, T.J. (2022). *An ant colony optimisation approach to scheduling truck and drone delivery systems*. Ph.D. thesis, Stellenbosch: Stellenbosch University.

Murray, C.C. & Chu, A.G. (2015). The flying sidekick travelling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, **54**, 86–109.

Poikonen, S., Wang, X. & Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks*, **70**, 34–43.

Pugliese, L.D.P., Guerriero, F. & Macrina, G. (2020). Using drones for parcels delivery process. *Procedia Manufacturing*, **42**, 488–497.

Rios, B.H.O., Xavier, E.C., Miyazawa, F.K., Amorim, P., Curcio, E. & Santos, M.J. (2021). Recent dynamic vehicle routing problems: A survey. *Computers & Industrial Engineering*, **160**, 107604.

Sacramento, D., Pisinger, D. & Ropke, S. (2019). An adaptive large neighbourhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, **102**, 289–315.

Schermer, D., Moeini, M. & Wendt, O. (2018). Algorithms for solving the vehicle routing problem with drones. In *Intelligent Information and Database Systems: 10th Asian Conference, ACIIDS 2018, Dong Hoi City, Vietnam, March 19-21, 2018, Proceedings, Part I 10*, 352–361, Springer.

Schermer, D., Moeini, M. & Wendt, O. (2019a). A hybrid vns/tabu search algorithm for solving the vehicle routing problem with drones and en route operations. *Computers & Operations Research*, **109**, 134–158.

Schermer, D., Moeini, M. & Wendt, O. (2019b). A metaheuristic for the vehicle routing problem with drones and its variants. *Transportation Research Part C: Emerging Technologies*, **106**, 166–204.

Ursem, R.K. (2002). Diversity-guided evolutionary algorithms. In *International Conference on Parallel Problem Solving from Nature*, 462–471, Springer.

Wang, X., Poikonen, S. & Golden, B. (2017). The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, **11**, 679–697.

Wang, Z. & Sheu, J.B. (2019). Vehicle routing problem with drones. *Transportation research part B: methodological*, **122**, 350–364.

Wren, A. & Holliday, A. (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Journal of the Operational Research Society*, **23**, 333–344.

Yang, Z., Tang, K. & Yao, X. (2008). Self-adaptive differential evolution with neighbourhood search. In *2008 IEEE congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 1110–1116, IEEE.

Zhu, K.Q. (2003). A diversity-controlling adaptive genetic algorithm for the vehicle routing problem with time windows. In *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, 176–183, IEEE.

Zhu, K.Q. (2004). Population diversity in genetic algorithm for vehicle routing problem with time windows. In *European Conference on Machine Learning. Pisa, Italy*, 537–547.