A Framework for the Analysis of Metaheuristics for the Travelling Salesman Problem with Drone with Interceptions

Rudolf Ernst[1(✉)], Tsietsi Moremi[1], Jacomine Grobler[1], and Phil M. Kaminsky[2]

[1] Department of Industrial Engineering, Stellenbosch University, Stellenbosch, South Africa
rudolfernst1998@gmail.com

[2] Department of Industrial Engineering and Operations Research, University of California at Berkeley, Berkeley, USA

Abstract. This paper proposes a framework for utilising continuous metaheuristic algorithms for solving drone delivery scheduling problems with interceptions. The use of the framework is illustrated through the application of a particle swarm optimisation-based algorithm, a differential evolution-based algorithm, and a covariance matrix adaptation-evolution strategy for Moremi's travelling salesman problem with drone with interceptions (TSPDi) [1]. A single drone can be launched to make a delivery while a truck is delivering packages to customers. The drone then either intercepts the purpose-built truck after completing a delivery when possible or meets up with the truck at the next customer location. The optimisation algorithms are tested on benchmark datasets and compared against Moremi's ant colony optimisation-based drone scheduling algorithm. Algorithm performance is measured with regards to total delivery time and total truck travel distance. An in-depth analysis of algorithm performance is conducted in the form of an efficiency study. The efficiency study specifically considers the number of duplicate and feasible solutions as the different algorithms handle the TSPDi constraints. It is shown that the metaheuristics are especially useful when a smaller number of deliveries are required, whereas performance deteriorates significantly when the number of nodes to be visited grows.

The contributions of this paper include the only in-depth analysis of the performance of continuous-based metaheuristics in drone scheduling literature, as well as the first framework for utilising continuous-based metaheuristics for solving drone delivery scheduling problems with interceptions. This paper contributes to the computational sciences by using advanced computing capabilities for decision support to improve the performance of an intelligent transport system.

Keywords: Drone delivery · Metaheuristics · Routing and optimisation · TSP with drone with interceptions

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023, LNCS 13957, pp. 351–368, 2023.
https://doi.org/10.1007/978-3-031-36808-0_23

1 Introduction

Last-mile freight is considered the final delivery step of goods from a fixed distribution facility to end-users [2]. The coordination of the "last mile" is a common problem in built-up environments. The closer to the aggregated demand point, the bigger the loss in efficiency and capacity is observed, due to the complexity of sequencing and allocating deliveries in an optimal manner [3]. With the growth of online shopping and accessibility to it, last-mile delivery companies are constantly seeking faster and more efficient ways to deliver products to customers in order to gain a competitive advantage [4].

In recent years, companies have investigated using "innovation vehicles" as well as collaborative or cooperative logistics, among others, to improve the efficiency of last mile deliveries [5]. One of these "innovation vehicles" being considered is a drone delivery vehicle or an unmanned aerial vehicle (UAV). Many practical applications of drones have been referenced in literature. The applications of drones range from the retailing and e-commerce sectors (Amazon, 2013 [6]; Flirtey, 2016 [7]), postal services and package delivery (United Postal Service, 2017 [8]; DHL, 2016 [9]), food and beverage delivery (Flirtey for Domino's Pizza, 2016 [10]; Alphabet for Chipotle, 2016 [11]) as well as healthcare and emergency services (Matternat for blood samples in Lesotho, 2016 [12]; Zipline for blood samples in Rwanda, 2017 [13]; Flirtey for medical supplies in Virginia, 2015 [14]).

Specifically, this paper proposes a framework that considers a problem based on the travelling salesman problem with drone (TSPD), as first presented by Agatz et al. [15], where a single truck and drone are used for pre-determined deliveries. The research involves a truck and drone waiting for one another at customer locations in the process of making deliveries. Agatz' research was expanded on by Moremi [1] to allow the drone to intercept the truck while in motion (TSPDi), on the arc between customer locations. The interceptions improve completion times and decrease waiting time, which is especially undesirable for drones due to security risks.

In this paper, a self adaptive neighbourhood search differential evolution (SaNSDE), guaranteed convergence particle swarm optimization (GCPSO), and covariance matrix adaptation evolution strategy (CMA-ES)-based algorithms are developed and evaluated as part of the framework. Two alternative repair mechanisms are implemented for the SaNSDE and GCPSO, as explained in Sect. 5. The algorithms are tested on benchmark datasets ranging from 10 to 250 nodes [16] and compared to the ant colony system (ACS) results of Moremi [1]. Furthermore, an in-depth efficiency study is conducted to understand the performance of the different metaheuristics applied to the TSPDi.

The results show that the SaNSDE performs the best on smaller datasets, whereas the CMA-ES performs the best among bigger datasets when only considering the algorithms implemented in this study. The CMA-ES is the most consistent performer, as it also finds good enough solutions for the smaller datasets. The efficiency study highlights the complications of applying metaheuristics to combinatorial optimisation problems such as the TSPDi. The metaheuristics generate large numbers of infeasible solutions but are able to navigate the solution space effectively, having minimal numbers of duplicate solutions.

The main contribution of this paper is that it is, to the best of the authors' knowledge, the first framework developed for utilising continuous-based metaheuristic algorithms, such as PSO, DE, and CMAES, for solving drone delivery scheduling problems with interceptions. The contributions of this paper include the only in-depth analysis of the performance of continuous-based metaheuristics in drone scheduling literature. The paper gives guidance as to applying metaheuristics to the truck drone scheduling problem as well as results to illustrate which metaheuristics perform better on different-sized datasets.

The rest of the paper is organised as follows: Sect. 2 introduces a literature survey of important drone and truck delivery research. Section 3 describes the mathematical model of the TSPDi, followed by the algorithm background in Sect. 4. The framework and experimental setup is presented in Sect. 5, while the empirical evaluation of the algorithms implemented on the TSPDi is done in Sect. 6. An efficiency study is presented in Sect. 7 and a conclusion is given in Sect. 8.

2 Literature Review

A significant amount of research has focused on optimising the last mile of deliveries, specifically with respect to the vehicle routing problem. Solomon and Desrosiers (1988) [17], Laporte et al. (1992) [18], Toth and Vigo (2012) [19], Kumar and Panneerselvam (2012) [20], and Lin et al. (2014) [21], amongst others, published surveys of vehicle routing literature. Extensive surveys, focusing on drones for deliveries, were conducted by Otto et al. (2018) [22], Khoufi et al. (2019) [23], Chung et al. (2020) [24], Poikonen and Campbell (2020) [25]; Moshref-Javadi and Winkenbach (2021) [26], and Liang and Luo (2022) [27].

Single truck and drone problems are mainly categorised and differentiated between a travelling salesman problem with drone (TSP-D) (Agatz and Bouman (2018) [15], Ha et al. (2018) [28]) or flying sidekick travelling salesman problem (FSTSP) (Murray and Chu (2015) [29], Boysen et al. (2018) [30]). TSP-D uses the concept of operations, while FSTSP uses the concept of sorties to model the problem [24]. Most of these problems considered in literature require either the drone or truck to wait for the other at customer locations when making deliveries before synchronising or meeting up again. Lastly, a truck and drone routing problem with synchronisation on arcs (TDRP-SA) (Li et al. 2022 [31]) allows for moving drone launch and retrieval locations. Drones are thus able to be launched to deliver a package and then land on a moving truck when returning. Salama and Srinivas [32] and Thomas et al. [33] also considered models where drones could be dispatched and retrieved on the arc between two customers.

A common extension of the FSTSP is allowing for multiple drones (mFSTSP) (Ferrandez et al. (2016) [34], Chang and Lee (2018) [35], Murray and Raj (2019) [36]). mFSTSP requires the truck to be stationary while the drones make deliveries, only moving to the next location when all drones have returned. On the other hand, the TSP-D is transformed into a TSP with multiple drones (TSP-mD) (Yoon (2018) [37], Tu et al. (2018) [38]), where the truck is allowed to move until the drones complete their deliveries, whereafter it is stationary

until they return. Murray and Chu (2015) [29] also presented the parallel drone scheduling travelling salesman problem (PDSTSP) formulation, where a single truck and multiple identical drones service customers directly from a depot. The next natural step in literature allows multiple trucks and drones to make deliveries (VRP-D), which is not covered in this paper.

3 Problem Background

The problem consists of a truck and drone having to make multiple predetermined deliveries to customers. A truck and a drone that can be launched from the truck leave the depot together when setting out to make deliveries. Whenever the drone makes a delivery, it is launched from the preceding truck node. After the drone completes a delivery, it either intercepts the truck, if possible, while it is in motion on the arc between two customer locations or meets up with the truck at the next customer location. The process is executed until all customers have been visited, whereupon the truck and drone reroute back to the depot. The mathematical formulation of the TSPDi is given below, while the full explanation of the TSPDi can be found in [1].

- n is the number of customers to be serviced.
- T_{ijk} denotes the total time for the scenario where a drone delivers to node j while the truck delivers to nodes i and k. T_{ijk} is calculated based on the drone and truck intercepting each other after the drone delivery. This interception point is calculated using the standard mathematical equations for two objects interceptions explained in [1].
- T_{ik} denotes the total time for scenario where the truck delivers to nodes i and k when there is not a drone delivery in between.
- w is the drone offload time

With the following decision variables:

- u_i is used to eliminate sub-tours.

$$x_{ijk} = \begin{cases} 1 & \text{if the truck services customer } i \text{ before } k \text{ while the drone services } j \text{ in between} \\ 0 & \text{otherwise} \end{cases}$$

$$b_j = \begin{cases} 1 & \text{if the customer } j \text{ is serviced by the drone} \\ 0 & \text{if customer } j \text{ is serviced by the truck} \end{cases}$$

$$z_{ik} = \begin{cases} 1 & \text{if the truck travels between customer } i \text{ and } k \\ 0 & \text{otherwise} \end{cases}$$

$$\min \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n} x_{ijk}T_{ijk} + \sum_{i=1}^{n}\sum_{k=1}^{n} z_{ik}T_{ik}(1 - \sum_{j=1}^{n} x_{ijk}) - w \tag{1}$$

Subject to:

$$z_{ii} = 0 \qquad \forall i \in \{1, ..., n\} \tag{2}$$

$$x_{iii}, x_{iij}, x_{iji}, x_{ijj} = 0 \qquad \forall i, j \in \{1, ..., n\} \tag{3}$$

$$z_{ik} = (\sum_{j=1}^{n} x_{ijk} = 1) => (z_{ik} = 1) \qquad \forall i, k \in \{1, ..., n\} \tag{4}$$

$$b_j = \sum_{i=1}^{n} \sum_{k=1}^{n} x_{ijk} \qquad \forall j \in \{1, ..., n\} \tag{5}$$

$$\sum_{i=1}^{n} z_{iq} - \sum_{k=1}^{n} z_{qk} = 0 \qquad \forall q \in \{1, ..., n\} \tag{6}$$

$$(b_i = 0) => (\sum_{k=1}^{n} z_{ik} = 1) \qquad \forall i \in \{1, ..., n\} \tag{7}$$

$$(b_k = 0) => (\sum_{i=1}^{n} z_{ik} = 1) \qquad \forall k \in \{1, ..., n\} \tag{8}$$

$$\sum_{j=1}^{n} x_{ijk} \leq 1 \qquad \forall i, k \in \{1, ..., n\} \tag{9}$$

$$u_i + z_{ij} \leq u_j + (n-1)(1 - z_{ij}) \qquad \forall i, j, j \neq 1 \in \{1, ..., n\} \tag{10}$$

$$u_1 = 0 \tag{11}$$

$$x_{ijk}, b_i, z_{ik} \in \{0, 1\} \qquad \forall\ i, j, k \in \{1, ..., n\} \tag{12}$$

- The objective function (1) minimises the total time that the drone and truck travel, which includes the travel time of all truck deliveries which do not include a drone release and intercept.
- Constraints (2) and (3) simplify the model by ensuring that the truck and drone cannot travel from the current node back to a previously visited node.
- Constraint (4) links z_{ik} and x_{ijk}.
- Constraint (5) links b_j and x_{ijk}.
- Constraint (6) ensures that if a delivery is conducted at a customer location, the truck or drone leaves that same customer.
- Constraints (7) and (8) ensure that each customer is arrived at from exactly one other customer (by truck), and that from each customer there is a departure to exactly one other customer (by truck). If node j is serviced by a drone, then no trucks enter or exit node j.
- Constraint (9) ensures that only one drone delivery can occur between two truck deliveries.
- Constraints (10) and (11) eliminate any sub-tours.
- Constraint (12) restricts the decision variables to binary values.

4 Algorithm Background

The SaNSDE, GCPSO and CMA-ES algorithms are explained in this section along with their relevant background and implementation.

4.1 Differential Evolution

The *differential evolution* (DE) algorithm is a population-based, stochastic optimisation algorithm. The DE is successful due to the difference vectors that determine the step sizes applied to the individuals in the population. The difference between two solutions is the core driver of the algorithm toward better solutions [39]. A differential mutation operator allows for the step sizes of the algorithm to automatically adapt to the objective function landscape [39]. Before a population converges around a specific optimum, the randomly sampled individuals are usually far apart in different regions of the search space. Larger step sizes will then be allowed when greater exploration of the search space is desired. These larger steps also make it possible to escape local optima. In the later iterations, when convergence around a good solution is desired, smaller step sizes are automatically applied due to the close proximity of all individuals in the search space. This strategy allows the DE algorithm to exploit the region around the optimum in search of even better solutions.

The *Self-adaptive DE with Neighbourhood Search* (SaNSDE) [40] algorithm applied in this paper has been developed by combining the best features of the *Neighbourhood Search DE* (NSDE) [40] algorithm and the *Self-adaptive DE* (SaDE) [41] algorithm.

4.2 Guaranteed Convergence Particle Swarm Optimisation

Particle swarm optimization (PSO) is a random population-based optimisation algorithm first implemented by Kennedy and Eberhart [42]. The PSO algorithm represents each candidate solution by the position a particle resides in within a multi-dimensional hyperspace. New candidate solutions, or positions, are determined through displacement and velocity updates, which move particles to new positions, allowing for exploration of the search space [43]. The magnitude of any particle's velocity at time t is the result of three vectors; the particle velocity at time $t-1$, a vector representing the best solution or position found up to time t by the specific particle, and the vector representing the overall best solution or position found up to time t by any particle.

The *guaranteed convergence PSO* (GCPSO) algorithm ensures that a random search around the global best particle position is executed by the *gbest* particle to avoid stagnation of the algorithm. The search space adjusts according to the number of successive successes or failures of the particle in improving the objective function. A success is recorded when there is an improvement in the objective function value.

4.3 Covariance Matrix Adaptation Evolution Strategy

The *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) is an algorithm operating in the continuous domain that has been successfully applied to many unconstrained or bounded constraint optimisation problems. The CMA-ES works in three phases, namely selection and recombination, covariance matrix adaptation, and step size control.

The CMA-ES algorithm is a combination of an evolutionary approach and a fitness landscape model [44]. A set of parameters describing an individual model, called sample points, are used to generate a new set or population of points, where the subset of points with the best fitness values survives to the next generation [44]. Specifically in the CMA-ES, the differences in the average fitness as well as the covariance between subsequent populations are used to move the population towards the optimum. The knowledge gathered from the interactions between parameters is iteratively saved in a covariance matrix [44] and used to allocate new sampling points, which drives the search process. Unfavourable premature convergence is avoided by the variable property of the step size.

5 Framework Description

Initially, a random population of N real-valued vectors is created, meaning N solutions are evaluated per iteration. The dimensions of the vectors are $2*(n-1)$ with decision variables between the bounds of l_b and u_b. These decision variables change throughout the algorithm's execution and are assigned to each operation. The activity diagram in Fig. 1 shows the framework, with the implementation of the metaheuristics, which is explained next.

To evaluate the fitness of the population in each iteration, the continuous values are converted into a schedule or route to follow by making use of priority-based mapping [45]. The first n values are sorted in descending order to find the permutation of nodes to be visited. The second n allocation variables are used to determine whether a node is visited by a truck or a drone, where a value smaller than zero is a drone node and a value bigger than or equal to zero is a truck node, in effect discretising the search space. For example, the following ten hypothetical decision variables are to be used to determine the sequence that **five** customers will be visited, as well as by which vehicle:

$$[6\ 71\ -99\ 62\ -20\ 13\ 59\ -5\ -20\ 3]$$

The first five priority values are re-sorted in descending order by the function, leading to the following order of nodes to be visited being obtained:

$$\text{Depot} \rightarrow \text{Node 2} \rightarrow \text{Node 4} \rightarrow \text{Node 1} \rightarrow \text{Node 5} \rightarrow \text{Node 3} \rightarrow \text{Depot}$$

The priority values are altered throughout the algorithm execution to guide the process of searching the solution space and are reassigned at each operation. The second n allocation variables are then used to determine which vehicle visits each of the n customers. The vehicle completing each of the deliveries, indicated by a D for drone and a T for truck, are now added in round brackets, with the allocation variable value.

$$\text{Depot} \rightarrow \text{Node 2 } [59](\text{T}) \rightarrow \text{Node 4 } [-20](\text{D}) \rightarrow \text{Node 1 } [13](\text{T}) \rightarrow \text{Node 5 } [3](\text{T}) \rightarrow \text{Node 3 } [-5](\text{D}) \rightarrow \text{Depot}$$

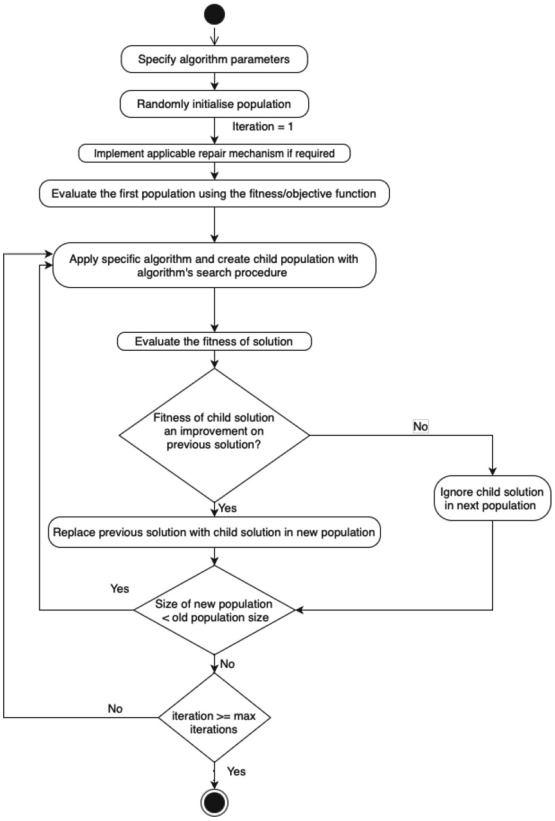

Fig. 1. Activity diagram showing framework

A repair process is required in cases where two consecutive drone nodes are scheduled, which does not conform to the TSPDi constraints. Two different procedures are developed for repairing the infeasible solutions, and both result sets are documented in an effort to understand the effect of applying metaheuristics to solve combinatorial optimisation problems. The first repair mechanism changes the second consecutive drone node to a truck node without altering the allocation variable. In the first instance, the algorithm is thus finding a fitness function for a "different solution" than it actually calculates. The second repair procedure changes the allocation variable of the second consecutive drone node to $u_b/2$, to ensure that there are no consecutive drone nodes. The value of $u_b/2$ is chosen, as this ensures that a truck node is scheduled, but that there is still room for the algorithm to move through solutions and change the value towards a negative number (and thus a drone node). The second repair process changes the mechanism the algorithm uses to move through the search space. Only the first repair process is implemented on the CMA-ES.

The framework and metaheuristics are evaluated on 13 datasets from Bouman and Schmidt [16] with varying sizes and coordinate spreads. It is assumed that the truck travels 10 distance units per time unit, whereas the drone travels 20 distance units per time unit. 0.1 time units are added per drone and truck delivery. N is chosen as 100 individuals per population, while u_b and l_b values of -100 and 100 are initialised.

6 Empirical Evaluation

The means of the delivery times and truck distances with their standard deviations over 30 optimisation runs are presented in Table 1 for all algorithms, as well for an ant colony system (ACS-DT2) of Moremi [1]. The first and second executions of repair mechanisms are shown separately for the SaNSDE and GCPSO. The performance indicator shows the % deterioration or improvement in the algorithm result when compared against the ACS-DT2. Optimal results for the TSPDi model by Knoetze [46] are presented for the small datasets. The time measurement is the most important factor and objective of the problem, but the distance results are included as a secondary measurement. Throughout the rest of the paper, any mention of results is related to the total time measurement, while small datasets are those with 20 nodes or less and large datasets are those with more than 20 nodes. The framework is able to provide final, feasible solutions for the TSPDi.

The framework and most algorithms instantiations are able to improve on the ACS-DT2 result for small datasets, while only the CMA-ES improves on the ACS-DT2 for several larger datasets. The SaNSDE-1 is able to improve up to 16.4% on the smaller datasets while performing poorly with large datasets, obtaining results that are up to 65.8% worse than the ACS-DT2. The SaNSDE-2 is able to improve up to 15.6% on the small datasets, while having a 67.1% worse result on the large datasets. The GCPSO algorithm has the worst results of the candidates. The GCPSO-1 is able to obtain up to 8.2% better results than the ACS-DT2 while having up to 75.9% decline on larger datasets. The GCPSO-2 algorithm obtains a 7.9% improvement at best, having up to 103.1% worse average results on larger datasets.

Most of the algorithms averaged solutions that were equal to or close to the optimal results obtained by Knoetze. These results indicate how well the metaheuristics perform on the TSPDi for small datasets. Problems larger than 20 can currently not be solved optimally, which highlights the benefit of using metaheuristics that find "good enough" solutions in a reasonable amount of time.

The standard deviation is relatively small for the metaheuristics on smaller datasets, but the large standard deviations on larger datasets highlights erratic performance, especially for the GCPSO. The SaNSDE-1 algorithm is the best candidate for smaller datasets, as it is able to gain the best results of any of the algorithm executions. The SaNSDE is underperforming on 100-node or larger datasets. The CMA-ES improves on the results of the ACS-DT2 for the 50 node datasets and one of the 100 node datasets. The CMA-ES is thus considered the best algorithm from this study for larger datasets.

Table 1. Algorithm Results

Datasets	Nodes	Optimal Time	ACS-DT2		SaNSDE-1			SaNSDE-2			GCPSO-1			GCPSO-2			CMA-ES		
			Time	Std Dev	Time	Std Dev	Perf	Time	Std Dev	Perf	Time	Std Dev	Perf	Time	Std Dev	Perf	Time	Std Dev	Perf
Uniform-51-n10	10	25.47	25.87	0.25	**25.47**	0.00	1.5%	**25.47**	0.00	1.5%	25.61	0.24	1%	25.76	0.43	0.4%	25.76	0.88	4.2%
Uniform-52-n10	10	20.3	20.71	0.20	**20.35**	0.07	1.7%	20.36	0.08	1.7%	20.63	0.29	0.4%	20.61	0.01	0.5%	20.94	0.54	−1.1%
Uniform-53-n10	10	20.2	20.98	0.10	**20.20**	0.00	3.7%	**20.20**	0.00	3.7%	20.59	0.49	1.9%	20.70	0.52	1.3%	21.21	0.56	−1.1%
Uniform-61-n20	20	25.49	32.67	0.63	**27.31**	0.94	16.4%	27.57	1.28	15.6%	29.98	2.43	8.2%	30.08	2.25	7.9%	30.92	1.51	5.3%
Uniform-62-n20	20	28.87	33.87	0.33	**30.05**	0.69	11.3%	30.24	0.62	10.7%	32.67	1.59	3.5%	32.67	1.59	3.5%	35.93	2.03	−6%
Uniform-63-n20	20	28.62	33.95	0.94	**28.79**	0.32	15.2%	28.83	0.34	15.1%	33.38	3.28	1.7%	33.66	2.56	0.9%	35.34	3.62	−4.3%
Uniform-71-n50	50		83.34	3.17	85.90	5.55	−3.1%	85.54	4.25	−2.6%	89.75	8.34	−7.7%	93.64	10.54	−12.4%	**81.18**	5.30	2.6%
Uniform-72-n50	50		85.56	1.49	95.45	3.77	−11.6%	92.79	4.29	−8.5%	94.44	9.13	−10.4%	99.46	9.56	−16.3%	**83.99**	5.55	1.8%
Uniform-73-n50	50		83.56	1.22	89.30	4.04	−6.8%	88.51	3.60	−5.9%	90.15	9.56	−7.9%	95.41	10.21	−14.2%	**88.36**	7.13	−5.7%
Uniform-91-n100	100		161.09	3.08	202.34	5.41	−25.6%	202.62	6.60	−25.8%	209.22	10.35	−29.9%	221.02	14.66	−37.2%	**157.05**	8.61	2.5%
Uniform-92-n100	100		149.18	2.30	200.27	6.32	−34.2%	201.59	5.41	−35.1%	211.25	17.00	−41.6%	213.97	12.93	−43.4%	**159.44**	8.71	−6.9%
Uniform-1-n250	250		382.98	5.30	634.89	9.30	−65.8%	640.00	9.90	−67.1%	673.62	32.00	−75.9%	778.14	65.43	−103.2%	**422.48**	20.99	−10.3%
Uniform-2-n250	250		385.04	7.62	626.88	7.27	−62.8%	633.17	11.15	−64.4%	659.68	30.99	−71.3%	754.33	53.64	−95.9%	**409.17**	15.55	−6.3%
Datasets	Nodes	Optimal Dist	ACS-DT2		SaNSDE-1			SaNSDE-2			GCPSO-1			GCPSO-2			CMA-ES		
			Dist	Std Dev	Dist	Std Dev	Perf	Dist	Std Dev	Perf	Dist	Std Dev	Perf	Dist	Std Dev	Perf	Dist	Std Dev	Perf
Uniform-51-n10	10		250.98	2.18	**248.31**	0.00	1.1%	248.31	0.00	1.1%	248.42	0.68	1%	250.30	4.24	0.3%	249.47	4.59	0.6%
Uniform-52-n10	10		195.22	3.32	**191.79**	1.39	1.8%	191.93	1.39	1.7%	194.88	4.14	0.2%	194.08	4.31	0.6%	193.24	5.54	1%
Uniform-53-n10	10		182.92	6.85	**190.48**	0.00	−4.1%	**190.48**	0.00	−4.1%	192.11	3.19	−5%	193.12	3.88	−5.6%	193.01	7.01	−5.5%
Uniform-61-n20	20		287.03	14.98	**235.63**	10.35	17.9%	235.85	17.78	17.8%	258.85	27.14	9.8%	262.63	29.40	8.5%	247.33	23.84	13.8%
Uniform-62-n20	20		315.17	7.83	**275.98**	6.84	12.4%	277.56	6.42	11.9%	301.20	15.42	4.4%	301.20	15.42	4.4%	308.84	36.81	2%
Uniform-63-n20	20		311.44	13.09	**274.00**	5.93	12%	274.43	4.10	11.9%	307.23	31.60	1.4%	311.88	26.82	−0.1%	317.18	27.97	−1.8%
Uniform-71-n50	50		732.46	59.76	720.41	55.54	1.6%	701.47	55.72	4.2%	765.80	80.07	−4.6%	813.53	117.95	−11.1%	**680.45**	65.53	7.1%
Uniform-72-n50	50		741.82	47.97	802.25	54.41	−8.1%	782.66	54.01	−5.5%	809.91	82.78	−9.2%	878.29	97.49	−18.4%	**720.96**	55.94	2.8%
Uniform-73-n50	50		730.77	25.68	751.33	54.33	−2.8%	**747.67**	53.90	−2.3%	781.56	89.60	−7%	845.42	96.08	−15.7%	754.39	58.85	−3.2%
Uniform-91-n100	100		1400.48	34.08	1635.97	97.17	−16.8%	1642.26	99.42	−17.3%	1751.24	128.38	−25%	1908.00	175.52	−36.2%	**1263.12**	84.34	9.8%
Uniform-92-n100	100		1294.68	68.68	1652.34	82.41	−27.6%	1663.78	59.89	−28.5%	1774.03	160.17	−37%	1808.59	154.58	−39.7%	**1301.16**	85.21	−0.5%
Uniform-1-n250	250		3093.95	153.67	5217.9	198.87	−68.6%	5250.31	164.57	−69.7%	5610.62	343.99	−81.3%	6909.78	619.82	−123.3%	**3377.06**	186.62	−9.2%
Uniform-2-n250	250		3234.35	74.98	5145.57	143.81	−59.1%	5209.33	147.41	−61.1%	5523.64	336.98	−70.8%	6646.18	554.17	−105.5%	**3297.91**	129.58	−2%

The convergence graphs for all algorithm executions on the uniform-52-n10 and uniform-2-n250 are presented in Fig. 2 and Fig. 3. The convergence graphs show how the algorithm improves on its best objective function value. The convergence around optimal solutions is much faster on smaller datasets, whilst the algorithms progressively improve on the fitness function over the iterations for larger datasets.

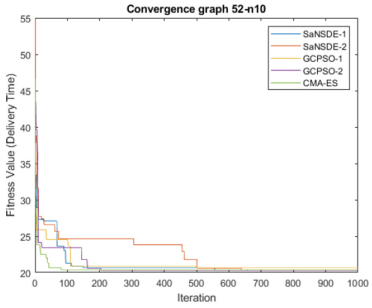

Fig. 2. Convergence graph for Uniform-52-n10 dataset

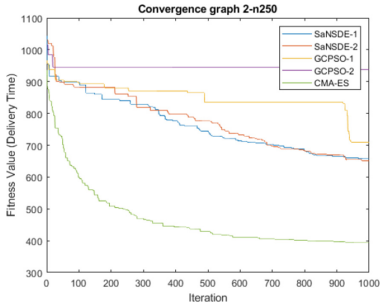

Fig. 3. Convergence graph for Uniform-2-n250 dataset

The SaNSDE-1 algorithm shows similar performance to the SaNSDE-2, indicating that the repair mechanism does not have a significant influence on the SaNSDE. The performance difference is below the 1 percentile point for most datasets, with a maximum difference of 1.3%. The results indicate that the GCPSO performs better when allowed to naturally search the solution space without feeding the repaired decision variable values to the algorithm for determining the next population of solutions. For smaller datasets, the repair mechanism does not significantly influence performance, but for the larger datasets, the performance difference is up to 15.5% in favour of the GCPSO-1.

Non-parametric Mann-Whitney U statistical tests are performed for each of the five algorithm combinations. The results are presented in Table 2. For each entry, the algorithm in row i is measured against the algorithm in column j. The entries are in the form Wins-Draws-Losses, indicating the number of times (out of the 13 datasets) each algorithm statistically significantly differs in average delivery time, at a significance of 0.05. The SaNSDE-1 algorithm outperforms the SaNSDE-2 algorithm twice while drawing on 10 occasions and losing once. The GCPSO-1 outperforms the GCPSO-2 on three datasets, drawing on 10 occasions. The SaNSDE-1 is superior to the GCPSO-1 on 11 occasions, drawing in 2 comparisons. Lastly, the SaNSDE combinations draw six times against the CMA-ES, while drawing once and losing on six of the datasets.

Table 2. Hypothesis Test Results

	SaNSDE-1	SaNSDE-2	GCPSO-1	GCPSO-2	CMA-ES
SaNSDE-1	X	2-10-1	11-2-0	12-1-0	6-1-6
SaNSDE-2	1-10-2	X	11-2-0	13-0-0	6-1-6
GCPSO-1	0-2-11	0-2-11	X	3-10-0	4-3-6
GCPSO-2	0-1-12	0-0-13	0-10-3	X	4-2-7
CMAE-ES	6-1-6	6-1-6	6-3-4	7-2-4	X

7 Efficiency and Feasibility Study

Further analysis in the form of an efficiency and feasibility study is executed to analyse the algorithm results on the TSPDi. Different key performance indicators (KPIs), specifically the duplicate solutions, inherent duplicates, algorithm efficiency, and infeasible solutions, are analysed for the algorithm-repair mechanism combinations and presented in graphic and tabular forms. The KPIs chosen are used to analyse how effective the algorithms are when attempting to improve on the best solution. The main KPIs consider the feasibility of solutions and their ability to search the entire solution space. The KPIs are measured over the 30 optimisation runs of the TSPDi study, as described in Sect. 6.

7.1 Duplicates Solutions

Firstly, the average % of duplicate solutions per optimisation run is presented in Table 3. Duplicate solutions are solutions with the same objective function value, which is the total delivery time. Solutions containing different routes that lead to the same total delivery time are considered duplicates. In more technical terms, when different genotypes have the same phenotype, they are considered to be duplicates. The nature of the optimisation processes of the different algorithms will always lead to multiple duplicate solutions, as metaheuristics can come across the same solution more than once. The theory is that due to multiple solutions in the decision space mapping to a single solution in the objective space, it is possible that continuous priority-based metaheuristics are inefficient for solving combinatorial optimisation problems.

Duplicate solutions are not clear indicators of poor algorithm performance, although research on the genetic algorithm, which is also a population-based metaheuristic, has shown that duplicates degrade quality early in an optimisation run [47]. The quality loss is attributed to diversity loss through duplicate solutions, which can be a serious weakness in population-based heuristics. Superior performance is associated with moving through a greater area of the search space and is attained through minimal duplicates, which are an indication that fewer solutions are repeatedly analysed and more of the search space is explored.

Table 3. Average % of duplicates per run

Dataset	Nodes	SaNSDE-1	SaNSDE-2	GCPSO-1	GCPSO-2	CMA-ES
Uniform-51-n10	10	**82.76%**	83.03%	93.07%	92.78%	94.64%
Uniform-52-n10	10	**80.83%**	81.00%	91.92%	90.80%	95.37%
Uniform-53-n10	10	79.99%	**78.98%**	91.14%	90.58%	94.28%
Uniform-61-n20	20	67.64%	**67.31%**	76.12%	76.29%	88.03%
Uniform-62-n20	20	63.09%	**62.73%**	74.17%	74.71%	86.77%
Uniform-63-n20	20	63.30%	**62.93%**	75.56%	75.41%	86.64%
Uniform-71-n50	50	82.78%	82.16%	87.22%	86.00%	**77.96%**
Uniform-72-n50	50	85.00%	84.56%	87.68%	85.34%	**80.51%**
Uniform-73-n50	50	83.90%	83.90%	86.73%	85.91%	**77.52%**
Uniform-91-n100	100	81.24%	**80.11%**	84.67%	82.13%	84.69%
Uniform-92-n100	100	81.35%	**80.16%**	84.50%	81.91%	84.85%
Uniform-1-n250	250	67.69%	**66.06%**	77.56%	85.14%	69.36%
Uniform-2-n250	250	67.92%	**66.57%**	78.13%	83.19%	69.62%

Overall, the SaNSDE algorithm is the best performing algorithm when considering the average % duplicate solutions per optimisation run. The CMA-ES is performing slightly worse than the GCPSO algorithm among smaller datasets. The CMA-ES, however, generates the fewest duplicates for the 50-node datasets and performs better than the GCPSO as the dataset sizes increase beyond 50. The GCPSO, specifically the GCPSO-2, is the overall worst-performing algorithm combination, generating more than 80% duplicate solutions for all but the 20-node datasets. The overall good performance of the CMA-ES on the 50-node datasets is attributed to its much better performance during early iterations.

7.2 Inherent Duplicates

The next KPI, inherent duplicates, are solutions with exactly the same decision variables. In order for two solutions to be inherently duplicate, identical genotypes, that will automatically have the same phenotypes, must be present. There will thus always be a 'less than or equal' number of inherent duplicates than duplicates, as inherent duplicates also have the same objective function value and are normal duplicates of one another. Inherent duplicates are much scarcer than duplicates, as the algorithms alter the decision variables when attempting to move through the solution space. Early convergence around a perceived optimal can lead to inherent duplicates. The inherent duplicates are presented in Table 4.

Clearly, neither of the GCPSO combinations generate any inherent duplicates, whereas the CMA-ES has a few inherent duplicates in the small datasets. Interestingly, the SaNSDE algorithm is the only algorithm generating a substantial number of inherent duplicates, considering it generates the least total number of duplicates.

Table 4. Inherent duplicates

Dataset	Nodes	SaNSDE-1	SaNSDE-2	GCPSO-1	GCPSO-2	CMA-ES
Uniform-51-n10	10	1.42%	1.31%	**0%**	**0%**	0.01%
Uniform-52-n10	10	1.23%	1.16%	**0%**	**0%**	0.05%
Uniform-53-n10	10	1.24%	1.24%	**0%**	**0%**	**0%**
Uniform-61-n20	20	3.62%	3.34%	**0%**	**0%**	**0%**
Uniform-62-n20	20	3.51%	3.36%	**0%**	**0%**	**0%**
Uniform-63-n20	20	3.01%	3.28%	**0%**	**0%**	**0%**
Uniform-71-n50	50	6.42%	6.47%	**0%**	**0%**	**0%**
Uniform-72-n50	50	6.65%	6.66%	**0%**	**0%**	**0%**
Uniform-73-n50	50	7.08%	7.17%	**0%**	**0%**	**0%**
Uniform-91-n100	100	6.09%	6.06%	**0%**	**0%**	**0%**
Uniform-92-n100	100	7.27%	7.34%	**0%**	**0%**	**0%**
Uniform-1-n250	250	6.95%	5.88%	**0%**	**0%**	**0%**
Uniform-2-n250	250	6.35%	5%	**0%**	**0%**	**0%**

7.3 Algorithm Efficiency

To obtain the number of true duplicates, the inherent duplicates have to be deducted from the total number of duplicates. The true duplicates are then used to calculate the efficiency of the algorithms. The overall efficiency is simply determined as the percentage of unique solutions generated on average per run, as in Table 5.

The SaNSDE combinations are almost twice as efficient as the GCPSO and CMA-ES combinations on all of the datasets. The repair mechanisms do not significantly change the SaNSDE efficiency, with only a small difference in efficiency observed in favour of the GCPSO-1 for large datasets. The average efficiency over the combined SaNSDE combinations is 28.93%. The SaNSDE is 12.79% more

Table 5. Algorithm efficiency

Dataset	Nodes	SaNSDE-1	SaNSDE-2	GCPSO-1	GCPSO-2	CMA-ES
Uniform-51-n10	10	**18.66%**	18.28%	6.93%	7.22%	5.37%
Uniform-52-n10	10	**20.4%**	20.16%	8.08%	9.2%	4.68%
Uniform-53-n10	10	21.25%	**22.26%**	8.86%	9.42%	5.72%
Uniform-61-n20	20	35.98%	**36.03%**	23.88%	23.71%	11.97%
Uniform-62-n20	20	40.42%	**40.63%**	25.83%	25.29%	13.23%
Uniform-63-n20	20	39.71%	**40.35%**	24.44%	24.59%	13.36%
Uniform-71-n50	50	23.64%	**24.31%**	12.78%	14%	22.04%
Uniform-72-n50	50	21.65%	**22.1%**	12.32%	14.66%	19.49%
Uniform-73-n50	50	23.18%	**23.27%**	**13.27%**	14.09%	22.48%
Uniform-91-n100	100	24.85%	**25.95%**	15.33%	17.87%	15.31%
Uniform-92-n100	100	25.92%	**27.18%**	15.5%	18.09%	15.15%
Uniform-1-n250	250	39.26%	**39.82%**	22.44%	14.86%	30.64%
Uniform-2-n250	250	**38.43%**	**38.43%**	21.87%	16.81%	30.38%

Table 6. Infeasible Solutions

Dataset	Nodes	SaNSDE-1	SaNSDE-2	GCPSO-1	GCPSO-2	CMA-ES
Uniform-51-n10	10	98.42%	98.00%	99.76%	94.26%	**68.05%**
Uniform-52-n10	10	98.11%	97.23%	98.29%	93.09%	**53.05%**
Uniform-53-n10	10	97.78%	97.14%	96.03%	88.84%	**57.96%**
Uniform-61-n20	20	99.95%	99.86%	99.99%	98.92%	**81.03%**
Uniform-62-n20	20	99.96%	99.85%	99.99%	98.16%	**90.48%**
Uniform-63-n20	20	99.94%	99.84%	99.99%	98.21%	**72.77%**
Uniform-71-n50	50	99.99%+	99.99%+	100%	**99.99%**	100%
Uniform-72-n50	50	100%	**99.99%**	100%	99.99%+	100%
Uniform-73-n50	50	100%	99.99%	99.99%	**98.25%**	100%
Uniform-91-n100	100	100%	**99.99%+**	100%	100%	100%
Uniform-92-n100	100	100%	99.99%+	100%	**99.09%**	100%
Uniform-1-n250	250	100%	100%	100%	**99.78%**	100%
Uniform-2-n250	250	100%	99.99%	100%	**99.95%**	100%

efficient than the CMA-ES, which has an overall efficiency of 16.14%. The average GCPSO has an efficiency of 16.21%, which is 12.7% less than the SaNSDE.

7.4 Infeasible Solutions

Next, the average percentage of infeasible solutions per optimisation run is measured. A solution is considered infeasible when the decision variable values of an individual map to a route that contains consecutive drone nodes in its solution, which is invalid. The nature of the population-based metaheuristics considered for the TSPDi ultimately leads to large numbers of infeasible solutions, usually initialised already by the random decision variable values in the initial population. Only considering feasible solutions reduces the search space, and therefore, although worrying, infeasible solutions are part of the search process (Table 6).

The algorithms struggle with consistently generating feasible solutions on the benchmark datasets. Only the CMA-ES performs decently on small datasets. The high number of individuals requiring a modification to their genotypes will indefinitely lead to worse fitness values, as changing a drone node to a truck node will not only increase the truck distance and delivery time, but the modification does not consider the effect on the objective function value.

When navigating through the solution space, the metaheuristics struggle with steering the solutions in a way that would lead to mostly feasible routes. For the smaller datasets, the CMA-ES is able to create child populations more optimally by avoiding generating as many infeasible solutions as the other algorithms. Although the SaNSDE and GCPSO generate an overwhelming number of infeasible solutions for all the datasets, the second repair mechanism performs slightly better. The CMA-ES is considered the best-performing algorithm, while the other algorithms generate almost only infeasible solutions.

Ultimately, the continuous population-based algorithms are not very efficient in moving through the search space, especially when considering the large

numbers of infeasible solutions being generated. A need for developing better mechanisms to move through the search space is required, which will greatly improve algorithm performance.

8 Conclusion

This paper proposes a framework and implements SaNSDE, GCPSO, and CMA-ES-based algorithms for the TSPDi on benchmark datasets ranging from 10 to 250 nodes. Two versions of the SaNSDE and GCPSO are implemented to test different repair mechanisms. The algorithms are then compared to an ACS algorithm of Moremi [1]. The SaNSDE-1 algorithm is the best-performing algorithm when 20 or fewer nodes are present in the system, while the CMA-ES performs best for larger datasets. Different KPIs are measured to understand the performance of the metaheuristics on the TSPDi, specifically the duplicate solutions and infeasible solutions being generated.

Future research can apply alternative algorithms within the framework, specifically the genetic algorithm, simulated annealing, or tabu search. There is also an opportunity to apply problem-specific heuristics and operators to the developed metaheuristics. Lastly, an opportunity arises to expand the problem to a VRPD, where multiple trucks and drones are used to serve customers.

Acknowledgements. This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Numbers: 129340).

References

1. Tsietsi John, M., et al.: An ant colony optimisation approach to scheduling truck and drone delivery systems. Ph.D. thesis, Stellenbosch University (2022)
2. Allen, J., et al.: Understanding the impact of e-commerce on last-mile light goods vehicle activity in urban areas: the case of London. Transp. Res. Part D: Transp. Environ. **61**, 325–338 (2018)
3. De Souza, R., Goh, M., Lau, H.-C., Ng, W.-S., Tan, P.-S.: Collaborative urban logistics-synchronizing the last mile a Singapore research perspective. Procedia. Soc. Behav. Sci. **125**, 422–431 (2014)
4. Lim, S.F.W., Jin, X., Srai, J.S.: Consumer-driven e-commerce: a literature review, design framework, and research agenda on last-mile logistics models. Int. J. Phys. Distrib. Logist. Manag. **48**(3), 308–332 (2018)
5. Ranieri, L., Digiesi, S., Silvestri, B., Roccotelli, M.: A review of last mile logistics innovations in an externalities cost reduction vision. Sustainability **10**, 782 (2018)
6. Wallace, G.: Amazon says drone deliveries are the future (2013). http://money.cnn.com/2013/12/01/technology/amazon-dronedelivery/index.html
7. Glaser, A.: 7-eleven beats Google and Amazon to the first regular commercial drone delivery service in the US (2016)
8. Perez, S., Kolodny, L.: UPS tests show delivery drones still need work (2017). https://techcrunch.com/2017/02/21/ups-tests-show-delivery-drones-still-need-work

9. Franco, M.: DHL uses completely autonomous system to deliver consumer goods by drone. New Atlas, 10 May 2016
10. Murphy, M.: The future is here: drones are delivering Domino's pizzas to customers (2016). https://qz.com/838254/dominos-is-delivering-pizza-with-autonomousdrones-to-customers-in-new-zealand
11. Levin, A.: Alphabet and Chipotle are bringing Burrito delivery drones to campus (2016). https://www.bloomberg.com/news/articles/2016-09-08/burrito-by-dronecoming-to-campus-in-test-of-alphabet-s-delivery
12. Wang, D.: The economics of drone delivery. IEEE Spectr. (2016)
13. Ackerman, E., Strickland, E.: Medical delivery drones take flight in East Africa. IEEE Spectr. **55**, 34–35 (2018)
14. Vanian, J.: This drone startup just achieved a milestone in doorstep delivery, 2016 (2016). http://fortune.com/2016/03/25/flirtey-drone-legal-delivery-urban
15. Agatz, N., Bouman, P., Schmidt, M.: Optimization approaches for the traveling salesman problem with drone. Transp. Sci. **52**, 965–981 (2018)
16. Bouman Paul, A., Schmidt, M.: Instances for the TSP with drone (2015). https://doi.org/10.5281/zenodo.22245
17. Solomon, M.M., Desrosiers, J.: Survey paper-time window constrained routing and scheduling problems. Transp. Sci. **22**, 1–13 (1988)
18. Laporte, G.: The vehicle routing problem: an overview of exact and approximate algorithms. Eur. J. Oper. Res. **59**, 345–358 (1992)
19. Toth, P., Vigo, D.: The Vehicle Routing Problem. SIAM (2002)
20. Kumar, S.N., Panneerselvam, R.: A survey on the vehicle routing problem and its variants (2012)
21. Lin, C., Choy, K.L., Ho, G.T., Chung, S.H., Lam, H.: Survey of green vehicle routing problem: past and future trends. Expert Syst. Appl. **41**, 1118–1138 (2014)
22. Otto, A., Agatz, N., Campbell, J., Golden, B., Pesch, E.: Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: a survey. Networks **72**, 411–458 (2018)
23. Khoufi, I., Laouiti, A., Adjih, C.: A survey of recent extended variants of the traveling salesman and vehicle routing problems for unmanned aerial vehicles. Drones **3**, 66 (2019)
24. Chung, S.H., Sah, B., Lee, J.: Optimization for drone and drone-truck combined operations: a review of the state of the art and future directions. Comput. Oper. Res. **123**, 105004 (2020)
25. Poikonen, S., Campbell, J.F.: Future directions in drone routing research. Networks **77**, 116–126 (2021)
26. Moshref-Javadi, M., Winkenbach, M.: Applications and Research avenues for drone-based models in logistics: a classification and review. Expert Syst. Appl. **177**, 114854 (2021)
27. Liang, Y.-J., Luo, Z.-X.: A survey of truck-drone routing problem: literature review and research prospects. J. Oper. Res. Soc. China **10**, 343–377 (2022). https://doi.org/10.1007/s40305-021-00383-4
28. Ha, Q.M., Deville, Y., Pham, Q.D., Há, M.H.: On the min-cost traveling salesman problem with drone. Transp. Res. Part C Emerg. Technol. **86**, 597–621 (2018)
29. Murray, C.C., Chu, A.G.: The flying sidekick traveling salesman problem: optimization of drone-assisted parcel delivery. Transp. Res. Part C Emerg. Technol. **54**, 86–109 (2015)
30. Boysen, N., Briskorn, D., Fedtke, S., Schwerdfeger, S.: Drone delivery from trucks: drone scheduling for given truck routes. Networks **72**, 506–527 (2018)

31. Li, H., Chen, J., Wang, F., Zhao, Y.: Truck and drone routing problem with synchronization on arcs. Nav. Res. Logist. (NRL) **69**, 884–901 (2022)
32. Salama, M.R., Srinivas, S.: Collaborative truck multi-drone routing and scheduling problem: package delivery with flexible launch and recovery sites. Transp. Res. Part E Logist. Transp. Rev. **164**, 102788 (2022)
33. Thomas, T., Srinivas, S., Rajendran, C.: Collaborative truck multi-drone delivery system considering drone scheduling and en-route operations. Available at SSRN 4080821
34. Ferrandez, S.M., Harbison, T., Weber, T., Sturges, R., Rich, R.: Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. J. Ind. Eng. Manag. (JIEM) **9**, 374–388 (2016)
35. Chang, Y.S., Lee, H.J.: Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. Expert Syst. Appl. **104**, 307–317 (2018)
36. Raj, R., Murray, C.: The multiple flying sidekicks traveling salesman problem with variable drone speeds. Transp. Res. Part C Emerg. Technol. **120**, 102813 (2020)
37. Yoon, J.J.: The traveling salesman problem with multiple drones: an optimization model for last-mile delivery. Ph.D. thesis, Massachusetts Institute of Technology (2018)
38. Tu, P.A., Dat, N.T., Dung, P.Q.: Traveling salesman problem with multiple drones. In: Proceedings of the Ninth International Symposium on Information and Communication Technology, pp. 46–53 (2018)
39. Price, K., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-31306-0
40. Yang, Z., Tang, K., Yao, X.: Self-adaptive differential evolution with neighborhood search. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), pp. 1110–1116 (2008)
41. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: 2005 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1785–1791 (2005)
42. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN 1995 - International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
43. Sahab, M.G., Toropov, V.V., Gandomi, A.H.: A review on traditional and modern structural optimization: problems and techniques. In: Metaheuristic Applications in Structures and Infrastructures, pp. 25–47 (2013)
44. Jedrzejewski-Szmek, Z., Abrahao, K.P., Jedrzejewska-Szmek, J., Lovinger, D.M., Blackwell, K.T.: Parameter optimization using covariance matrix adaptation - evolutionary strategy (CMA-ES), an approach to investigate differences in channel properties between neuron subtypes. Front. Neuroinform. **12**, 47 (2018)
45. Grobler, J., Engelbrecht, A.P., Kok, S., Yadavalli, S.: Metaheuristics for the multi-objective FJSP with sequence-dependent set-up times, auxiliary resources and machine down time. Ann. Oper. Res. **180**, 165–196 (2010). https://doi.org/10.1007/s10479-008-0501-4
46. Knoetze, F.E.: Solving a last mile truck and drone delivery schedule to optimality. Bachelor's thesis, University of Stellenbosch (2021)
47. Ronald, S.: Duplicate genotypes in a genetic algorithm. In: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), pp. 793–798 (1998)