Hindawi
Advances in Operations Research
Volume 2020, Article ID 4053983, 10 pages
https://doi.org/10.1155/2020/4053983

Research Article

Inverting the Truck-Drone Network Problem to Find Best Case Configuration

Robert Rich

Industrial & Systems Engineering, Liberty University, Lynchburg, VA, USA

Correspondence should be addressed to Robert Rich; rkrich@liberty.edu

Received 16 October 2019; Revised 3 December 2019; Accepted 21 December 2019; Published 22 January 2020

Academic Editor: Demetrio Laganà

Many industries are looking for ways to economically use truck/rail/ship fitted with drone technologies to augment the "last mile" delivery effort. While drone technologies abound, few, if any studies look at the proper configuration of the drone based on significant features of the problem: delivery density, operating area, drone range, and speed. Here, we first present the truck-drone problem and then invert the network routing problem such that the best case drone speed and range are fitted to the truck for a given scenario based on the network delivery density. By inverting the problem, a business can quickly determine the drone configuration (proper drone range and speed) necessary to optimize the delivery system. Additionally, we provide a more usable version of the truck-drone routing problem as a mixed integer program that can be easily adopted with standardized software used to solve linear programming. Furthermore, our computational metaheuristics and experiments conducted in support of this work are available for download. The metaheuristics used herein surpass current best-in-class algorithms found in literature.

1. Introduction

The use of drones or other parallel-constrained resources in conjunction with main delivery assets offers potential performance improvements that may prove beneficial. The base problem for the truck-drone (DTSP) can be easily visualized as two shoppers working together to fetch goods from the shelves of a supermarket as efficiently as possible. As one shopper pushes the cart, the second shopper may stroll alongside or may separate and operate parallel fetching items back to the cart. While the two shoppers may be tasked independently in parallel operation, there will be times when it is more efficient to walk together. Indeed, it is obvious that there is an optimal set of routes for each shopper, but what is less intuitive is that the necessary speed and range of the parallel shopper will prevent any such optimization or may significantly delay the main shopper. Here, we examine this relationship.

For a truck-drone (say UPS) parcel-delivery system, it is easy to imagine that a very slow drone will afford little or no benefit to the truck. For at nearly every stop, the truck will wait idly while looking for the drone's return. The lumbering drone delivers its only one package on a parallel path and is practically useless—regardless of its range. Furthermore, that a very fast drone affords no real improvement in delivery time at all if it has only a short range. The drone only becomes a helpful servant if the drone's range and speed are correctly proportioned to the truck's speed. Thus, it is easy to imagine that there exists an optimal relationship between truck's speed, drone range, drone speed, and the delivery density of the network.

For this problem, we assume the truck can launch the drone with only one parcel from any delivery location and then rendezvous w/drone downstream at an adjacent delivery location while the drone delivers on a parallel path to the truck. This truck-drone routing is depicted (Figure 1).

The remaining sections comprised herein as follows. Section 2 discusses the inverted problem and other theoretical insights to solve for "best" case drone range and drone speed. Section 3 discusses the literature surrounding the truck-drone problem. Section 4 defines a more usable version for the mixed integer programming (MIP). Section 5 formulates the truck-drone problem as an evolutionary algorithm (EA) type metaheuristic algorithm used to

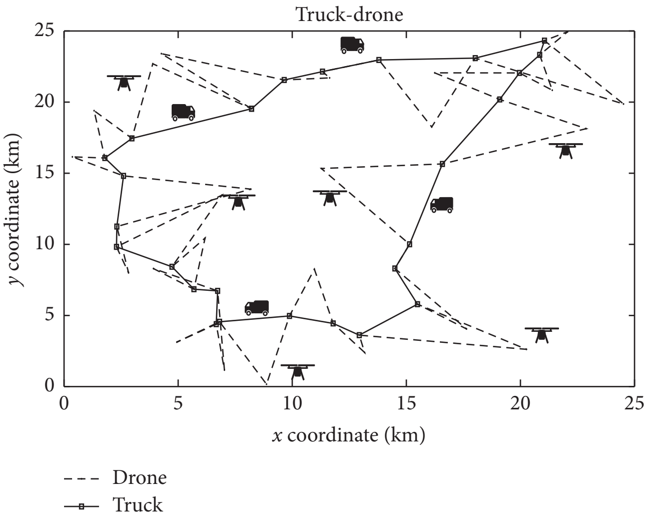


Figure 1: Truck with single-drone parcel delivery.

conduct computational experiments. Section 6 performs computation experiments for the EA against best-in-class found heuristics in literature, and Section 7 concludes the research on findings as well as the direction of future research.

2. System Model: Theoretical Insights

For a three-node problem whereby node v_0 denotes the depot and parcel deliveries are to be made to nodes i and j; then, the best delivery time for the system is the max of the drone's time or the truck's time. The problem can easily be scaled by establishing that the truck's speed is always one, and the drone's speed is a multiple (or factor of truck's speed) as in $(1 \times \alpha)$. Furthermore, since drone's speed α is as factor of truck speed, then an optimal configuration exists when drone speed equals drone's range κ. In other words, we are not saturating our drone with unnecessary resources (range or speed) to perform the delivery operation. As illustrated (Figure 2), the truck launches the drone, moves out a distance of 1 unit at rate of 1 unit/distance, and then returns to rendezvous with the drone. The total delivery time is $t = \max(2, 2\alpha)$, and an optimal configuration exists when range and speed for the drone are equal ($2 = 2\alpha$ and $\alpha = \kappa$).

However, most problems are not simple three-node problems having one truck and one drone. For more advance network problems, the delivery density ρ becomes a critical component when solving for drone range and speed (α, κ). Delivery density is defined here as the number (N) of required deliveries per area (A) of the delivery space $\rho = N/A$.

In order to build the case to solve for optimal speed α and range κ, we start with a "best case" or lean scenario. In such case, we would expect that approximately 50% of the deliveries will be made by the truck and 50% by the drone. Furthermore, under these ideal conditions, a set of same size triangles can be constructed inside the area of operation A to represent all the parcel deliveries (Figure 3), where vertices of the triangle denote the delivery locations. We will call this the *practical best case*.

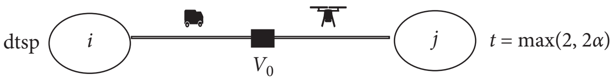


Figure 2: Best case for truck and one-drone [1].

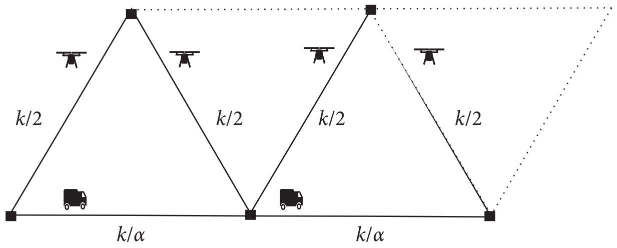


Figure 3: Practical best case parallelogram.

Using these triangles and simple parallelogram geometry, we can easily calculate the optimal drone speed and drone range necessary to rendezvous with the truck at exactly the same time at each parallel delivery operation. In such case, we would not need to spend any extra on unneeded resources while guaranteeing that the system will perform optimally.

For the *practical best case* system, we calculate the delivery density as $\rho = N/A$. Using density (ρ), we then replace N/A with $N/((N/2) - 1)(\kappa^2/2\alpha)$, where area of the parallelogram is denoted as $A = ((N/2) - 1)(\kappa^2/2\alpha)$. Since we are given the delivery density when given the problem scenario, we can then invert the problem and solve for proper drone capability (speed (α) and range (κ)) that solves the problem:

$$\rho^* \sim \frac{N}{((N/2) - 1)(\kappa^2/2\alpha)}: \alpha \geq 1, \kappa > \alpha, N > 4, \rho > 0. \quad (1)$$

Furthermore, we conducted several computational experiments to better understand the relationship between lean deliveries and randomly generated stochastic situations. For each scenario, random delivery locations were uniformly distributed in the area of operation while drone speed (α), range (κ), and operating area (A) were held constant. The number of deliveries N' was perturbated ranging from 10 to 200 deliveries within the area of operation. Here, we wished to understand the percent improvement gained Π (in delivery time) over a stand-alone truck (no drones) delivery system. We found that as the delivery density of the random scenario $\rho^* = (N'/A)$ approached or moved toward the delivery density of the lean solution $\rho^{\text{opt}} = (N'/((N/2) - 1)(\kappa^2/2\alpha))$ that performance time Π (over a stand-alone truck) improved. In such case, the delivery and routing portion of the problem for the randomly generated experiments was solved using the metaheuristics described below, while the delivery time for a stand-along truck was solved using standard tsp metaheuristics.

The results (Figure 4) showed that as the problem delivery density of the randomly generated scenario was close to the optimal delivery density (defined by N, drone speed (α), and

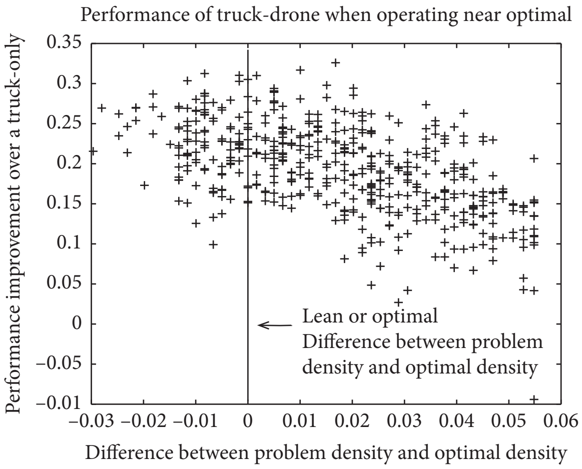


Figure 4: Percent improvement Π over truck-only time as ρ^* moves toward optimal density ρ^{opt}.

range (κ)) that the overall performance time Π improved without oversaturating the system with unneeded resources. Concretely, the abscissa (x-axis) denotes the difference between rho optimal based on alpha and kappa and the problem density rho based on N and delivery area. The ordinate (y-axis) denotes the improved performance over a truck-only delivery system (tsp-routed). The graph shows that as the two densities are closer together, the overall performance of the system improves. Given good results of using the *practical best case* methodology as a basis to evaluate the performance of a system, we constructed the following minimization problem to solve for the proper drone speed and drone range based on number of deliveries N and size delivery area of operations $A\,\text{km}^2$. As such, minimization problem here is used to select system parameters α and κ that are suited for a particular problem density ρ. It is noted that due to the stochasticity of the problem that the solution for range and speed are lower boundaries for the proper speed and range of the drone configuration:

$$\begin{aligned}
\min \quad & \left|\rho^{\text{opt}} - \rho^*\right| \\
\text{s.t.} \quad & \rho^{\text{opt}} = \frac{N}{((N/2)-1)(\kappa^2/2\alpha)}, \\
& \rho^* = \frac{N}{A}, \\
& \kappa_{\text{LB}} \le \kappa \le \kappa_{\text{UB}}, \\
& \alpha_{\text{LB}} \le \alpha \le \alpha_{\text{UB}}, \\
& N \ge 4, \rho^*, \\
& \rho > 0.
\end{aligned} \tag{2}$$

In summary, the theorems for the optimal parameters for the truck-drone are as follows.

(i) There exists a maximum *theoretical upper boundary* for time improvement factor which will rarely, if ever, be reached [1]. This is based on the comparison between truck-drone and truck-only solution:

$$\prod^{*} \le 1 - \left(\frac{1}{1+\alpha}\right). \tag{3}$$

(ii) The minimum *lower boundary* for time improvement (worst case) of truck-drone over the truck-only solution defaults to the truck-only route time or a tsp route time.

(iii) The improvement (Π) over truck-only moves toward optimal as the uniformly distributed vertices within a delivery area move toward the optimal density:

$$\rho^{\text{opt}} \sim \frac{N}{((N/2)-1)(\kappa^2/2\alpha)}. \tag{4}$$

Therefore, for a delivery scenario, if we are to be within the vicinity of the "best set" of operating parameters to optimize Π^*, then system parameters are found by minimizing the difference between delivery density ρ^* and optimal density ρ^{opt}. The resulting drone parameters and relationship between the two done parameters α and κ become the lower boundaries for an optimal configuration for the drone.

3. Literature

Today, a vast body of literature exists on tsp and the closely related vehicle routing problem (vrp). Several approaches to both problems can be found in surveys, reports, and papers [2–4]. As a rule, the vrp problem extents the tsp by adding additional constraints. These constraints comprised such things as time windows, priorities, range, loiter times, permissible-to-vehicle type route segments, load configurations, and traffic patterns [5]. While many variants of the tsp/vrp problems exist in literature to include multivehicle mtsp, customer pickup and delivery problem, multiple synchronization constraints vrp [6], multiple depot vehicle scheduling problem [7], and many-to-many milk run routing problem [8], there exists only a handful of studies concerning the dtsp, and no studies exist that address the proper configuration (capabilities) of a main tool (truck) and one or more assisting tools when solving for the optimal network routing.

Work that addresses the main tool and a constrained assisting tool is the work of Agatz et al. [1] and Murray and Chu [9]. Both of these authors propose the truck-drone problem as a type of main tool with assisting tools. Their works delineate the main differences between existing problems and their specific version of the tsp problem by distinguishing the drone's ability to travel with (on/in) the truck as well as to operate with the truck in several parallel operations described as launch-deliver-rendezvous tasks. Before this demarcation, no problem dealt squarely with the unique paradigm. Concretely, the dtsp describes the drone as a parallel resource to the truck required to periodically rendezvous with the truck due to payload and distance restrictions. Murray and Chu [9] formally define the *flying sidekick traveling salesman problem (FSTSP)* as an *NP-hard* problem. Their study suggests a mixed integer programming (MIP) approach as well as metaheuristic approach. They also consider a second similar hub type problem that addresses the case where the customers are close enough to the depot to be serviced directly from the depot by the drone as the *parallel drone scheduling TSP (PDSTSP)*. Both Murray and Agatz discuss the mixed integer formulation for the optimal min-time route. They allude to the fact that a drone is constrained in range, capacity, and speed as it relates to the truck. However, they do adequately address this interrelationship. Furthermore, Agatz only considers a slightly altered version of the FSTSP and does not address the PDSTSP. Agatz, like Murray, considers the truck-drone in tandem as a team whereby the truck launches the drone, traverses to a separate delivery location from the drone, and then rendezvous with the drone again. However, the main difference between the approaches is that Agatz et al. [1] require that the drone and truck traverse along the road network system; a constraint not enforced by [9]. They do this to facilitate construction of heuristic approaches with approximations that guarantee a bound on the maximum achievable gain of the delivery system over a "truck-only solution." Murray and Chu [9] formally define the *flying sidekick traveling salesman problem (FSTSP)* as an NP-hard problem. Their study suggests a mixed integer programming (MIP) as well as the metaheuristic approach. They also consider a second similar hub-type problem that addresses the case where the customers are close enough to the depot to be serviced directly from the depot by the drone as the *parallel drone scheduling TSP (PDSTSP)*.

More recently Agatz et al. [10] presented an exact solution approach for the truck-drone problem denoted as the traveling salesman problem with drone (TSP-D) based on dynamic programming. They modeled the problem first as an integer program and then developed multiple route-first, cluster-second heuristics based on local search and dynamic programming. Their insights suggested that larger problem instances could be solved by dynamic programming better than other mathematical programming approaches found in literature. They show worst case approximation ratios for their heuristics and compare the performance to optimal solutions for smaller instances. They applied their heuristics to several artificial instances with differing characteristics and sizes to show substantial improvements over the truck-only solution.

Perhaps, the problem herein most similar is the covering traveling salesmen problem described by Current and Schilling [11]. The covering problem finds the shortest tour for the "tsp" network by reducing the problem to a subset of the total nodes. In such case, if the adjacent nodes can be reached or "covered" within range then they can be grouped as a single node or stop, thus reducing the final optimal tour as well as the number of permutations of the problem significantly. For the covering problem, they propose a metaheuristic based solution to solve larger problem sets. Several generalizations and extensions to the covering problem can be found in literature [12]. For the main tool with assisting tools, it is not enough to bypass the unvisited or 'covered' nodes for three reasons: (1) first, the main and assisting tool requires that the truck and drone rendezvous at the end of each operation, and thus their end of operation timing is a factor of the total time. (2) Secondly, the covered nodes 'within-range' are subject to the number of assisting tools fitted to the main tool. (3) And, thirdly, the range, speed, and number of assisting tools are interdependent to the node coordinates and thus determine the total possible 'coverage' area which may be different for each operation.

Furthermore, the speed and range of the assisting tool is significant to the overall performance of the system. Herein, we give a closed form approach to determine the proper speed and range necessary for assisting tools to obtain an optimal performance of the system without oversaturating the system. This is not currently found in any literature. Many other authors create variations of the main/assisting tool problem, but do not address these fundamental relationships.

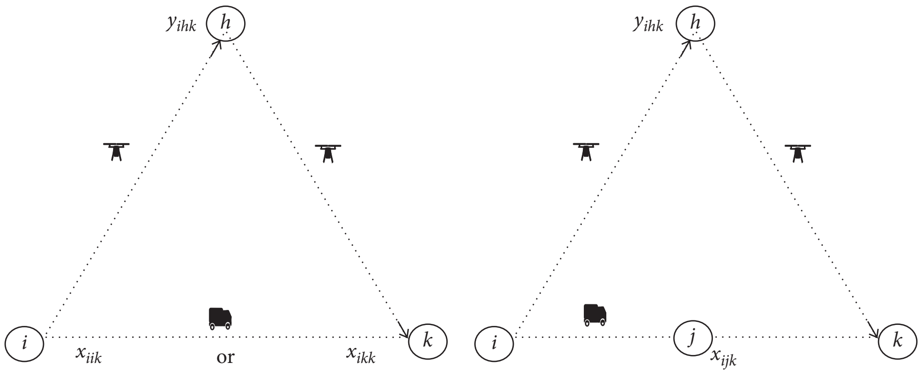
4. Truck-Drone Mixed Integer Programming

The parallel resource truck-drone problem is recognized when the associated second resource is constrained to remain at some proximity to the main vehicle, and when separated, it will eventually rendezvous with the main vehicle at a downstream location. It is permitted to temporarily separate from the main resource, but must soon return due to range or operational constraints.

In order to establish a generalized mathematical formulation and improve the tractability and software modeling (Lingo/Lindo ®) of this problem, we assume the following: (1) number, location, and distances between customers (nodes) are known and deterministic, (2) each node must be visited no more than once by a vehicle or a group of vehicles, (3) vehicles must traverse along arcs (edges) between nodes, (4) vehicles separate and rendezvous at node locations, not along arc space between nodes, (5) any vehicle may separate from the group to deliver to a node before it must rendezvous with the main vehicle (or group) at a downstream node; once rendezvous has occurred, each vehicle is again permitted to separate to another delivery node and then rendezvous again at a downstream node, and (6) each delivery node has demand of one unit; thus, one sortie of drone or truck is capable of delivering a parcel.

Given we have one drone (y) operating in adjacent space of the main delivery truck (x), then if drone (y) is used for delivery, and it shall be launched from the truck at node i, traverses to a delivery node (customer) at j, and rendezvous with the delivery truck at third node k, whereby the binary tuple variable $(y_{ijk} = 1)$ flags that the route segment is used by drone. After launching drone, the truck (x) has two options: (1) traverses directly from launch node to the rendezvous node $(x_{ijk} = 1)$ as depicted (Figure 5) and (2) traverses to yet another delivery node j and then proceeds onto the drone rendezvous node $(x_{ijk} = 1)$.

The truck-drone dtsp problem's material elements can be described mathematically as a network graph $G = (V, E)$, where V denotes the customers-delivery-stops and E denotes the edges between stops. Each vertex $V = \{1, \ldots, |V|\}$ and N denotes the cardinality of $|V|$ or total number stops. An edge E is described by two vertices $\{i, j\}$, whereas two joined edges are described as three vertices $\{i, j, k\}$, where vertex j is the vertex between i and k. Furthermore, binary variables (x_{ijk}, y_{ijk}) denote whether an edge triplicate is used by a truck x_{ijk} or a drone y_{ijk}; and x_{ijk} or $y_{ijk} = 1$ if used, else x_{ijk} or $y_{ijk} = 0$. The binary variable $x_{ijk} = 1$ if truck traverses



(a)

(b)

Figure 5: Truck and one-drone problem depiction: (a) truck option 1 and (b) truck option 2.

$E(i,j)$ and then further traverses to $E(j,k)$. If a drone travels on a truck in a dormant configuration, the drone is assigned the truck's i, j, k tuple index variables $y_{ijk} = 1$; else, a drone may serve in an active delivery role and therefore be assigned binary variable $y_{ihk} = 1$ denoting the drone was launched from vertex i, delivered to h, and subsequently recovered at vertex k. An important characteristic of the problem here is that the first and last vertex index of the truck and the drone must always be the same. The distance matrix D and the construction of triplet distances as tuple d_{ijk} denotes the cost or distance of traversing the $E(i,j)$ and $E(j,k)$. The subtour elimination variable u_i ensures that sequence of j follows i in the event that $E(i,j)$ is part of the solution; as such, it restricts subtour formations.

The minimax objective function minimizes the max $(Z = \sum_{i\in C}\sum_{j\in C} z_{ij})$ time of the truck or drone as each visit three nodes in an operation. As such, z_{ij} is used to evaluate each vehicle's time, where i, j are the first and last nodes of the three visited nods for truck (i,k,j) or drone's (i,h,j). Since z_{ij} is greater than or equal to the truck or drone time, we force z_{ij} to evaluate each vehicle's time using $z_{ij} \geq \sum_{k\in C} x_{ikj} d_{ikj}$ which denotes truck time while $z_{ij} \geq \sum_{h\in C} y_{ihj} d_{ihj}(1/\alpha)$ denotes drone time divided by speed factor α which is a factor of truck's speed:

$$\text{Minimize} \quad Z = \min \sum_{i\in V}\sum_{j\in V} z_{ij} \tag{5}$$

$$\text{subject to} \quad \begin{cases} z_{ij} \geq \sum_{k\in V} x_{ikj} d_{ikj} \\ z_{ij} \geq \sum_{h\in V} y_{ihj} \dfrac{d_{ihj}}{\alpha} \end{cases} \quad \forall i, j \in V, i \neq j, \tag{6}$$

$$\begin{cases} x_{ikj} == 0 \\ y_{ikj} == 0 \end{cases} \quad i = k, k = j, \quad \forall i, j, k \in V, \tag{7}$$

$$\sum_{k\in V} x_{ikj} = \sum_{h\in V} y_{ihj} = 1, \quad \forall i, j \in V, i \neq j, \tag{8}$$

$$\sum_{j\in V}\sum_{\substack{k\in V\\ i\neq k}} x_{ijk} + \sum_{j\in V}\sum_{\substack{k\in V\\ k\neq j}} x_{kij} + \sum_{\substack{j\in V\\ j\neq i}}\sum_{k\in V} x_{jki} + \sum_{j\in V}\sum_{k\in V} y_{kij} \geq 1, \quad \forall i \in V, \tag{9}$$

$$\begin{cases} \sum_{j\in V}\sum_{\substack{k\in V\\ i\neq k}} x_{ijk} \leq 1 \\ \sum_{j\in V}\sum_{\substack{k\in V\\ j\neq k}} x_{jik} \leq 1 \\ \sum_{j\in V}\sum_{\substack{k\in V\\ j\neq i}} x_{jki} \leq 1 \end{cases} \quad \forall i \in V, \tag{10}$$

$$\begin{cases} \sum_{j\in V}\sum_{\substack{k\in V\\ j\neq i}} y_{jik} \leq 1, \end{cases} \quad \forall i \in V, \tag{11}$$

$$\sum_{k\in V}\sum_{\substack{i\in V\\ i\neq j}} x_{ikj} = \sum_{k\in V}\sum_{\substack{j\in V\\ i\neq j}} x_{jki}, \quad \forall i \in V, \tag{12}$$

$$u_j \geq u_i + \sum_{k\in V} x_{ikj} - (N-2)\left(1 - \sum_{k\in C} x_{ikj}\right) + (N-3)\sum_{k\in V} x_{jki}, \quad \forall i, j \in V, \tag{13}$$

$$y_{ijk} d_{ijk} < \kappa + \left(M x_{ijk}\right), \quad \forall i, j, k \in V, \tag{14}$$

$$\forall d_{ijk} \geq 0, \quad i, j, k \forall V, \tag{15}$$

$$\forall x_{ijk}, y_{ijk} \in \begin{cases} 0 \\ 1 \end{cases} \quad \forall i, j, k \in V, u \in \{1, 2, \ldots, N\}. \tag{16}$$

Equation (5) minimizes the sum of the max time for any route triplet segment starting at i and ending at j. Equation (6) mandates that the sum of the max time is the max of either the truck time or the drone time for any given operation described as a triplet where an operation could be a truck-only or truck-drone. Equation (7) mandates that no triplet can contain $i = j = k$. Each operation must be a move where node i is different from node j or node k. Ensures that vehicle loitering. Equation (8) enforces the concept of an operation whereby a truck and a drone must operate with same launch and recovery node; however, the delivery node can be different. Equation (9) forces every node to be visited by a truck or a drone or a combination thereof. Equation (10) constrains a truck to visit a city once and only once. Equation (11) constrains a drone to only visit a city to deliver at most once. Equation (12) constrains each tuple segment entered by the truck shall exit the same by truck. Equation (13) is a subtour elimination constraint to force the sequencing of the route such that no subtour is possible. Equation (14) forces all drone deliveries within the drone's delivery operation range κ (launch, delivery, and rendezvous) unless riding in a truck in which case Mx_{ijk} disables the constraint. Equation (15) constrains all Euclidean distances to greater than zero. Equation (16) assigns the variables x and y as binary; and the utility sequencing variable u is an integer between one and the total number of nodes (or customer deliveries).

5. Truck-Drone Evolutionary Algorithm Approach

The tournament-based evolutionary algorithm (EA) here adopts a *cluster-during-routing* approach to solve the truck-drone problem. More accurately, it assigns both *truck* and *drone* labels during the routing process. This is much different from all other algorithms found in literature. The *best-in-class* found in literature perform an entire routing operation first, and then the algorithm labels *truck* or *drone*. Conversely, the algorithm herein denoted as EA creates a population matrix of randomly permuted routes whereby each node in a tour is evaluated as a potential drone-delivery node unless that node is out of range. Since a population of many randomly generated tours is evaluated simultaneously, any node within drone range is autoassigned and labeled *drone*. The EA performs the following process steps:

(a) Randomly permutes a population P of m tours where each tour denoted as a genome sequence $(1, 2, \ldots, n)$ for n delivery nodes in the tour.

(b) Determines the fitness for each population member (tours) based on total tour delivery time. All fitness times are saved for seed tournament.

(c) The total population is divided into groups of five tours each to conduct a set of seed tournaments.

(d) For each of the groups, the best member within the seed group (of the five) is chosen as the single gene to mutate for the remaining four members of the seed group.

  (1) Gene mutation (tour mutation) first copies the fittest member of the group of five within the seed tournament to replace the four less fit members.

  (2) Each of the four less fit members (now identical to the fittest) are then slightly mutated to improve fitness.

  (3) Mutations comprised (a) randomly selecting and swapping two nodes within the tour, (b) reverse ordering of the tour between two nodes, (c) sliding a tour segment down between nodes to left or right, and (d) replacing the last node in the tour with any other node.

(e) Repeat step (b) until convergence. Stopping condition is based on a predetermined iteration budget, tolerance, or saturation found in improvements.

(f) Return fittest member of the entire population.

A seed tournament genetic algorithm's strength lies in the ability to retain multiple paths toward optimization during the process which is critical for any network routing problem. Furthermore, because there are multiple members in a seed tournament, the algorithm allows for various proven mutation methodologies to be performed on the members of the seed tournament. In this case, the swap, flip, and slide mutations have proven to be robust, fast, and extremely accurate for many routing problems including the tsp, multiple-truck tsp, as well as vehicle routing problems (vrp).

The performance of the algorithm is based on the underlying theoretical principles:

(1) By initializing a relatively large population (i.e., $5n$) of randomly permuted tours, multiple paths (seed tournaments) increase the probability of an optimal convergence.

(2) By saving the fittest gene in a seed, and then slightly perturbing (mutating) the best gene (tour) found in the seed group of five tours ensures the solution never gets worse while promoting improvements at each iteration.

(3) By autoassigning the drone to any "within range" node, the use of the drone is maximized throughout the routing process while simultaneously reducing the truck's overall tour length. The risk of assigning the wrong node to a drone is mitigated by first initializing the population with random permutation and then maintaining multiple paths toward optimization.

(4) Multiple path random search is much faster than having to calculate the greediness or the exactness of each neighborhood within reach as in other algorithms. Therefore, the algorithm relies on computational speed and iterations without the burden of unnecessary calculations.

For the algorithm, we randomly permute a population of tours denoted as the initial population matrix P comprising m tours each having n nodes in the tour (n length of tour). In such case, $P[1,:]$ denotes the first tour (1) in the population matrix and all the nodes (:) for that first tour. Whereas variable R^1 denotes the first tour in population $P[1,:]$, then R_i^1 also denotes the i^{th} node within the first tour or route. The function D denotes a distance function that properly calculates the distance between segments of the route. The distance function nomenclature $D(R_i^p, R_{i+2}^p)$ denotes the distance between nodes (i) and node ($i+2$) found in route $R^p \in P[p,:]$. Furthermore, the neighborhood function $N(R_i^p)$ denotes the next three nodes or route segments for R_i^p as in $[R_i, R_{i+1}, R_{i+2}]$ which is used to evaluate whether or not a segment is reachable by the drone given the constraint drone range. In such case, the algorithm's distance function can handle the distance between two nodes, or a neighborhood of three nodes as in $D(N(R_i^p))$. The returned distance from D function is then divided by the speed of the truck or the drone to arrive at time.

6. Metaheuristic Comparisons

As stated, our evolutionary algorithm (EA) (Algorithm 1) metaheuristic uses a *cluster-during-routing* approach, whereby each drone is autoassigned to any next delivery node in the tour if it is within range. This forces the drone to be highly utilized. Next, we initialize the population with multiple randomly permuted routes to improve potential optimization paths thus helping to preclude a local optima. At each iteration, each tour in the population is slightly improved using mutations which comprised random node swaps, random tour segment flip, and random segment slide right or left.

Concretely, our *cluster-during-routing* EA is robust, fast, and is able to solve large problems optimally better than any other found in literature. To illustrate, we compared its performance to mathematical procedures (MIP), artificial intelligence constraint-based programming procedures (CP), and the 'best-in-class' heuristics conducted by Agatz et al. [13]. To prove optimality, we compared the EA to closed form mixed integer programming approaches for smaller size problems less than 15 nodes and then to IBM constraint-based programming (CP) for larger problem sizes up to 80 delivery nodes.

The EA was compared to the following methods:

(1) Closed form mixed integer programming (MIP) for smaller problems (less than 12 nodes) to prove optimality.

(2) Constraint-based programming (CP) for larger problems (less than 80 nodes) to prove optimality.

(3) MST-gp-all (route mst first, cluster with greedy algorithm, all iterative improvements)

(4) MST-ep-all (route mst first, cluster with exact partitioning dynamic programming with all iterative improvements)

(5) TSP-gp-all (route tsp first, cluster using greedy algorithm with all iterative improvements)

(6) TSP-ep-all (route tsp first, cluster using exact partitioning with all iterative improvements)

6.1. Comparison Study 1 (Smaller Problem Sizes: Against Best-in-Class). For the sake of simplicity, we generate node coordinates within an (x, y) Cartesian coordinate system and assume truck or drone distance Euclidean distance based on the uniform distribution. In the original comparison problem [13], the metaheuristic comparison experiments drew from three different distributions: uniformly (random) distributed nodes, 1-center Gaussian-distributed nodes, and 2-center Gaussian-nodes. In our study, we analyzed the results obtained from the different distributions and found that the underlying distribution had no statistical relevance for our EA metaheuristic; therefore, for simplicity, we adopted the uniform distribution from which to draw our experiments. Concretely, all nodes for problem comparisons herein were drawn from the uniform distribution with from $\{0, 1, 2, \ldots, 100\}$.

The table (Table 1) below provides results for 10 randomly generated instances with 10 nodes of each instance type. For each instance the optimality delta gap is defined as

$$\Delta = \frac{\text{objective value heuristic} - \text{optimal objective value}}{\text{optimal objective value}}. \tag{17}$$

Results proved the EA outperformed other methods found in literature. The EA herein proved optimal for 10/10 instances for similar problem sizes found in current literature. As such, we increased the problem size, constructed a constraint based model (CP) model in order to compare to optimal, and conducted additional experiments.

6.2. Computational Study 2 (Larger Problem Sizes: MIP, CP, and EA). The study was run on a 64-bit version of Xumbutu® 15.04 on virtualBox™ 4.3.12 hypervisor with windows 7™ as the host OS. The EA was coded in the MATLAB®. The files we created for experimental purpose were made available at Mathworks® file exchange (dtsp_ga_basic) for evaluation/comparison. The hardware configuration consisted of an Intel® Core i7-4770 CPU with 16 GB of RAM. And, the MIP and CP were all coded in IBM OPL 12.8.0 on a personal computer with an Intel Core i5-3537 @ 2.5 Ghz processor and 8 GB RAM.

A total of 10 test problem instances having nodes (between 10 and 100) were randomly generated and then averaged. As in other studies, the drone is assumed to be twice the speed of the truck ($\alpha = 2$). In this case, all customers are distributed across an 8 mile square region. The maximum flight endurance (drone range) is set as 14 miles as drone range. Table 2 summarizes the computational results of test problem instances. Column 1 shows different job sizes (number of nodes n). Columns 2–4, 5–7, and 8–10 record the elapsed computational processing time (*elapse ti.*), objective function (f), denoting total time, and gap of MIP, CP, and EA, respectively, from optimal. The gap is calculated against the best solution if optimality guaranteed. However, if optimality not guaranteed for larger problems, gap is calculated against best of MIP, CP, and EA delivery time. In such case, IBM CP and LINDO/LINGO MIP optimization software

```
DATA: a population $P$ matrix of initially randomly generated tours where tour $R^1_: = P[1, :]$ Size of the population is $m$ and the length
of any tour is $n$.
The distance function Distance (launch, deliver, rendezvous) determines the total distance between the nodes in the
RESULT: an ~ optimal tour for the nodes $[1 : n]$.
FOR iter 1 to Budget LOOP
    FOR $p$ in $m$ LOOP (for each population of tours)
        $R^p \longleftarrow P[p, :]$; (a tour within the population of tours)
        $R^p \longleftarrow P[p, 1 : n, 1, 2, 3]$; (add wrap back around to depot to tour)
        $\text{time}^p \longleftarrow 0$; (initialize total tour time for this tour)
        WHILE $i \leq n$ DO (go through each of the nodes in tour)
            case ⟵ 1; (initialize default case: truck carries drone and delivers)
            launch ⟵ $R^p$ $(i)$; (truck launch node)
            deliver ⟵ $R^p$ $(i + 1)$; (drone delivery node)
            rendezvous ⟵ $R^p$ $(i + 1)$; (truck and drone rendezvous node)
            launch fathom ⟵ $R^p$ $(i + 1)$; (check next operation truck launch node)
            deliver fathom ⟵ $R^p$ $(i + 2)$; (check next operation drone delivery node)
            rendezvous fathom ⟵ $R^p$ $(i + 3)$; (check next operation truck/drone rendezvous node)
            drone dist ⟵ Distance (launch, deliver, rendezvous)
            truck dist ⟵ Distance (launch, , rendezvous)
            IF drone dist <range AND $i + 1 \leq n$ THEN
                case ⟵ 2;
                fathom 1 ⟵ max[(drone dist)/(drone speed), (truck dist)/(truck speed)]
                drone dist 2 ⟵ Distance (fathom launch, fathom deliver, fathom rendezvous)
                truck dist 2 ⟵ Distance (fathom launch, , fathom rendezvous)
                fathom 2 ⟵ max[(drone dist 2)/(drone speed), (truck dist 2)/(truck speed)]
                IF drone dist 2 < range AND fathom 2 < fathom 1 AND $i + 2 \leq n$ THEN
                    case ⟵ 1; (save drone for next operation, set to truck deliver for this iteration)
                    drone dist ⟵ 0; (no drone distance)
                END IF
            ELSE
                    drone dist ⟵ 0; (out of drone range. . .)
            END IF
        SWITCH CASE
            CASE == 1 (truck delivers)
                truck dist := Distance (launch, , deliver); (find truck distance to next node)
                $k \longleftarrow k + 1$;
            CASE == 2 (truck and drone deliver)
                truck dist := Distance (launch, , rendezvous); (find truck distance to rendezvous)
                drone dist := Distance (launch, deliver, rendezvous); (find drone distance for operation)
                $k \longleftarrow k + 2$; (two nodes satisfied)
        END CASE
        $\text{time}^p = \text{time}^p$ + max[(drone dist)/(drone speed), (truck dist)/(truck speed)] (capture and record the total time for population
member $p$)
        END WHILE LOOP
    END FOR LOOP
$P$ ⟵ randomly shuffle rows in population matrix $P$ for a tournament (do not change tours)
FOR $p = 5 : 5 : m$ LOOP (select groups comprised of 5 tours each from the population $P$ of size $m$)
    Best time ⟵ Get Best Time ($P[p, :]$) for the group of the 5 tours
    Best Id ⟵ Find route id of the group of 5 having the best time (Best Time)
    $P'[p_{1,2,3,4,5}, :]$ ⟵ Replace all tours in population group $P(p, :)$ with fittest tour of the 5 tours.
    $P'$ $[p_1, :]$ ⟵ keep (do not mutate) the fittest tour of the group of 5 (keep for next iteration)
    $P'[p_{2,3,4,5}, :]$ ⟵ Mutate the other 4 less fit tours as follows:
        (1) tour 1: randomly select 2 points in the route $P'[p_2, :]$ to swap
        (2) tour 2: randomly select 2 points $P'[p_3, :]$to reverse all nodes in between
        (3) tour 3: randomly select 2 points in route $P'[p_4, :]$to slide to left and replace last with first
        (4) tour 4: replace the first and last nodes $P'[p_5, :]$with two randomly selected nodes
END FOR LOOP
$P$ ⟵ update $P$ with all mutations $P'$
END FOR LOOP
RETURN best $\text{time}^p$ and best route found in population
```

Algorithm 1: Evolutionary algorithm.

Table 1: Comparison against best-in-class heuristics. Comparison against "best-in-class" heuristic solutions to the optimal solution (uniform $n = 10$ nodes, $\alpha\ = 2$, averaged over 10 instances).

	Nodes: Cartesian coordinates—uniform distribution (10 nodes)		
	Δ from optimal		
	Avg.	Max	#opt
MTSP-gp-all [13]	2.0	6.4	1/10
MTSP-ep-all [13]	0.7	2.7	4/10
TSP-gp-all [13]	1.6	3.1	1/10
TSP-ep-all [13]	0.4	2.3	6/10
EA	0.00	0.00	10/10

Table 2: Elapsed time and objective function value according to different job sizes.

1	2	3	4	5	6	7	8	9	10
Nodes		MIP			CP			EA	
	Elapse time	f	Gap	Elapsed time	f	Gap	Elapsed time	f	Gap
10	1 s	***228.0**	0.0%	1 s	***228.0**	0.0%	1 s	***228.0**	0.0%
20	54 s	285.5	0.0%	20 s	**285.5**	0.0%	5 s	285.5	0.0%
30	1834 s	400.0	8.4%	10 s	**369.0**	0.0%	15 s	374.0	1.4%
40	1692 s	621.0	44.6%	55 s	**429.5**	0.0%	30 s	434.0	1.0%
50		—		118 s	**466.0**	0.0%	45 s	478.5	2.7%
60		—		422 s	**415.5**	0.0%	60 s	419.5	1.0%
70		—		1398 s	**511.5**	0.0%	90 s	504.0	1.7%
80		—		1745 s	**523.0**	0.0%	120 s	546.5	4.5%
90		—			—		120s	**638.0**	0.0%
100		—			—		120s	**655.0**	0.0%

Best solution in bold. *Optimal values, no valid bound found within the time limit of 1800 s.

cannot guarantee optimality for jobs (number of nodes) greater than 20+ or when computer computational time is above 1800 seconds on current computer systems. In such case, CP reports the best solution it finds and gives a probability of being optimal (i.e., up to 80 job instances). Conversely, EA consistently delivered an efficient solution with the shortest computational time. Larger sized problems require more computer processing capability for comparison studies to optimal.

All the test instances and MIP and CP logs are located at the following link: https://drive.google.com/open?id=19ZZ9ukEwSSfjCqNID1P1kPsGMOyMefXC.

7. Conclusions

The current literature lacks any usable information in terms of the highest yield regions of the design space (speed, range, and number of drones) for the truck-drone configurations, and any company considering the use of a delivery-drone has no way to guide the selection of drone parameters (speed factor α and range κ) without some visibility into the high yield regions of the operating space. The *lean geometry approach* shown herein inverts the problem space to solve for practical best case drone speed and drone range given a randomly generated scenario. As such, it is quite useful for practical design decisions regarding the proper/most efficient drone speed and range to achieve the maximum desired yield over the truck-only solution. As such, it serves as a way to screen out the inadvertent selection of low performing designs.

The study also gives a simplified version of the MIP not found in any other study as well as a useful and easily implemented metaheuristic necessary to solve for the optimal route and optimal time for the truck-drone problem. We used a simple single chromosome evolutionary algorithm (EA) metaheuristic to test each case. The EA was modeled as function within the MATLAB® development environment language, and the files were made available at Mathworks® file exchange (dtsp_ga_basic) for evaluation. The EA was tested against current best-in-class heuristics found in literature and significantly surpassed them in terms of accuracy as well as computational time performance.

In conclusion, this research answers the questions of expected efficiencies in time would be expected given a truck-drone configuration as well as finding on "what is the proper configuration in terms of drone range and drone speed for a truck-drone situation" given a typical delivery scenario. It gives business a foundation to evaluate a variety of configurations against their typical daily last-mile parcel-delivery scenarios. The work also opens several additional questions for future research. The most obvious questions involve the design space time and/or efficiencies of 1-truck which comprised many drones.

Data Availability

The metaheuristic codes and algorithms are available at MATLAB.com and also from the corresponding author upon request.

Conflicts of Interest

The author declares that there are no conflicts of interest.

References

[1] N. Agatz, P. Bouman, and M. Schmidt, "Optimization approaches for the travelling salesman problem with drone," in *Social Sciences Research Network*, Social Science Electronic Publishing Inc. by Elsevier, Amsterdam, Netherlands, 2015.

[2] B. Golden, S. Raghavan, and E. Wasil, "The vehicle routing problem: latest advances and new challenges," in *Operations Research/Computer Sciences Interfaces*, vol. 43, Springer, Berlin, Germany, 2008.

[3] P. Toth and D. Vigo, *The Vehicle Routing Problem: SIAM Monographs on Discrete Mathematics and Applications*, SIAM Monographs on Discrete Mathematics and Applications, Philadelphia, PA, USA, 2002.

[4] S. N. Kumar and R. Panneerselvam, "A survey on vehicle routing problem and its variants," *Intelligent Information Management*, vol. 4, no. 3, 2012.

[5] C. E. Miller, E. W. Tucker, and R. A. Zemlin, "Integer programming formulations and travelling salesman problems," *Journal of the ACM*, vol. 7, pp. 326–329, 1960.

[6] M. Drexl, "Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints," *Transportation Science*, vol. 46, no. 3, pp. 297–316, 2012.

[7] A.-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman, "A comparison of five heuristics for the multiple depot vehicle scheduling problem," *Journal of Scheduling*, vol. 12, no. 1, pp. 17–30, 2009.

[8] Y. Lin, Z. Bian, S. Sun, and T. Xu, "A two stage simulated annealing algorithm for the many-to-many milk run routing problem with pipeline inventory cost," *Mathematical Problems in Engineering*, vol. 2015, Article ID 428925, 22 pages, 2015.

[9] C. C. Murray and A. G. Chu, "The flying sidekick traveling salesman problem: optimization of drone-assisted parcel delivery," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 86–109, 2015.

[10] P. Bouman, N. Agatz, and M. Schmidt, "Dynamic programming approaches for the traveling salesman problem with drone," 2017, https://ssrn.com/abstract=3035323.

[11] J. R. Current and D. A. Schilling, "The covering salesman problem," *Transportation Science*, vol. 23, no. 3, pp. 208–213, 1989.

[12] D. J. Gulczynski, J. W. Heath, and C. C. Price, "The close enough traveling salesman problem: a discussion of several heuristics," in *Operations Research/Computer Science Interfaces Series*, F. B. Alt, M. C. Fu, and B. L. Golden, Eds., vol. 36, pp. 271–283, Springer, Boston, MA, USA, 2006.

[13] N. Agatz, P. Bouman, and M. Schmidt, "Optimization approaches for the traveling salesman problem with drone," *Transportation Science*, vol. 52, no. 4, pp. 965–981, 2018.