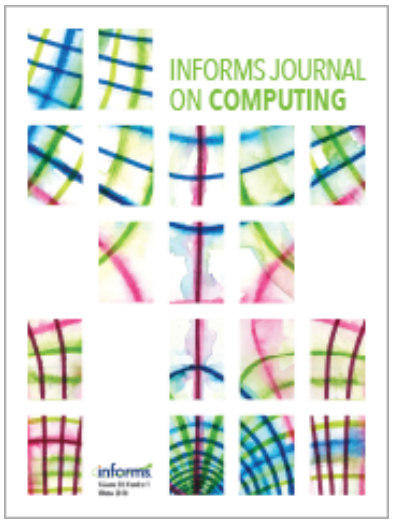


INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Branch-and-Bound Approach to the Traveling Salesman Problem with a Drone

Stefan Poikonen, Bruce Golden, Edward A. Wasil

To cite this article:
Stefan Poikonen, Bruce Golden, Edward A. Wasil (2019) A Branch-and-Bound Approach to the Traveling Salesman Problem with a Drone. INFORMS Journal on Computing 31(2):335-346. https://doi.org/10.1287/ijoc.2018.0826

http://pubsonline.informs.org/journal/ijoc/

INFORMS JOURNAL ON COMPUTING
Vol. 31, No. 2, Spring 2019, pp. 335–346
ISSN 1091-9856 (print), ISSN 1526-5528 (online)

A Branch-and-Bound Approach to the Traveling Salesman Problem with a Drone

Stefan Poikonen,[a] Bruce Golden,[b] Edward A. Wasil[c]

[a] Business School, University of Colorado Denver, Denver, Colorado 80202; [b] R.H. Smith School of Business, University of Maryland, College Park, Maryland 20742; [c] Kogod School of Business, American University, Washington, District of Columbia 20016

Contact: stefan.poikonen@ucdenver.edu, http://orcid.org/0000-0002-2924-1831 (SP); bgolden@rhsmith.umd.edu, http://orcid.org/0000-0002-5270-6094 (BG); ewasil@american.edu, http://orcid.org/0000-0002-8397-4809 (EAW)

Received: June 14, 2017
Revised: October 8, 2017; February 17, 2018; February 24, 2018; March 4, 2018
Accepted: March 5, 2018
Published Online in Articles in Advance: April 26, 2019

https://doi.org/10.1287/ijoc.2018.0826

Abstract. The Traveling Salesman Problem with a Drone (TSP-D) is a hybrid truck and drone model of delivery, in which the drone rides on the truck and launches from the truck to deliver packages. Our approach to the TSP-D uses branch and bound, whereby each node of the branch-and-bound tree corresponds with a potential order to deliver a subset of packages. An approximate lower bound at each node is given by solving a dynamic program. We provide additional variants of our heuristic approach and compare solution quality and computation times. Consideration is given to various input parameters and distance metrics.

History: Accepted by Alice Smith, Area Editor for Heuristic Search.
Supplemental Material: The online supplement is available at https://doi.org/10.1287/ijoc.2018.0826.

Keywords: 970 transportation • vehicle routing • 490 networks-graphs • traveling salesman • 630 programming • integer • algorithms • branch and bound

1. Introduction

Several technological improvements, including better battery life, improved communications systems, advances in stability, and reduction of manufacturing costs, have increased the viability of using drones to make deliveries. Drones have been used in healthcare and disaster response contexts, particularly in remote regions (Knight 2016).

Amazon, FedEx, and UPS have explored the use of drones for parcel delivery (Cary and Bose 2016). In September 2015 the Finnish postal service (Posti Group) experimented with drone delivery of packages to an island near Helsinki (Reuters 2015). Dynamic Parcel Distribution is the first company to have launched (with all regulatory approvals) a regular route service with a drone in the Provence region of France (Dynamic Parcel Distribution 2017).

In February 2017, UPS released a video of a test of a truck and drone hybrid delivery (UPS YouTube Channel 2017) for which the drone road atop the truck. The truck stopped near a customer location and launched the drone with a package to a different customer location to make a delivery. While the drone was in the air, the truck made a delivery and then rendezvoused with the drone. This hybrid model of delivery is the focus of this paper.

2. Literature Review

Murray and Chu (2015) were the first to introduce a hybrid truck and drone model under the name The Flying Sidekick. In preliminary testing of a mixed integer linear programming model, the authors indicated that routes with up to 10 packages "may require several hours to solve" to optimality. The long solution times motivated a heuristic method, such as the one below:

1. Solve a standard Traveling Salesman Problem (TSP) and use this as an initial truck route.

2. In a greedy way, select a package currently on the truck route to be delivered by a drone. This greedy selection preserves feasibility.

Ha et al. (2015) solved a mixed integer program that optimized the selection of drone operations according to various objective functions. A drone operation with triplet (i, j, k) launches the drone from the truck at package location i, delivers a package via drone to package location j, and returns the drone to the top of the truck at package location k. A modified TSP routing was then performed, based on the selection of drone operations from the mixed integer program.

Wang et al. (2017) considered theoretical bounds for the maximum speedup ratio of using a hybrid truck with drone model relative to a truck-only model. The authors described optimization problems that arose when using several trucks with one or more drones. Poikonen et al. (2017) generalized the bounds of Wang et al. (2017) to the case in which trucks and drones operated on different metrics. The authors also showed that the close-enough vehicle routing problem may serve as a lower bound to the vehicle routing problem with drones.

Agatz et al. (2018) considered a problem similar to that in Murray and Chu (2015) that they named the Traveling Salesman Problem with a Drone (TSP-D).

They constructed a family of heuristics and an integer program and found that the best-performing heuristic without applying iterative, local improvement procedures was TSP-ep, in which ep denotes exact partitioning. First, a standard traveling salesman problem was solved. The goal was then to exactly partition this solution into truck-delivered nodes and drone-delivered nodes. Without loss of generality, the authors relabeled the nodes with indices $0, 1, 2, \ldots, N$ such that if node a appeared before node b in the standard TSP solution, then $a < b$. Node 0 is the origin depot, and node N is the destination depot, which may be the same as the origin depot.

Agatz et al. (2018) continue by considering a case in which $i < k < j$ and the truck and drone are located together at node i. The drone launches from the truck to node k to deliver a package. While the drone is airborne, the truck delivers to every node $l \neq k$ such that $i < l < j$, and then both the truck and drone rendezvous at node j. Agatz et al. (2018) defined $T(i, j, k)$ as the amount of time between the drone launch at node i and the rendezvous at node j. $T(i, j, k) = \infty$ when a triplet is infeasible. They defined $T(i, j) = \min_k(T(i, j, k))$. It is beneficial to choose the package location k that minimizes the time until the rendezvous at j. Let $k = -1$ indicate that truck delivery to all nodes l such that $i < l < j$ produces a faster arrival to node j than launching a drone. TSP-ep used the following recursive formula (a special case of the Bellman-Ford Equation (Bellman 1958, Busacker and Saaty 1965):

$$V(0) = 0$$

For i = 1 to N:

$$V(i) = \min_k(V(k) + T(k, i))$$
$$Prev(i) = \operatorname{argmin}_k(V(k) + T(k, i)).$$

$V(N)$ is the best TSP-D objective value under the restriction that both the truck delivery order and the drone delivery order are subsequences of the standard TSP solution. Though optimal under this restriction, in general TSP-ep does not provide the globally optimal solution to the TSP-D. One may retrace the optimal path by iteratively backtracking from node N to $Prev(N)$, then to $Prev(Prev(N))$, then to $Prev(Prev(Prev(N)))$, etc. until reaching the depot where the truck and drone departed. The cost of exactly partitioning a TSP sequence into a TSP-D solution is $O(N^3)$.

Agatz et al. (2018) embedded their exact partitioning procedure in several iterative improvement procedures with the following structure.

1. Find the the optimal TSP solution, called *BestTour*.
2. Construct a neighborhood of similar tours around *BestTour*. Call it *Neighbors*.
3. For each *tour* in *Neighbors*
   3.1. Apply the exact partitioning method.
   3.2. Compute the associated TSP-D objective value of the partitioned route, called $Obj(tour)$.
   3.3. If $Obj(tour) < Obj(BestTour)$, set $NewBestTour = tour$.
4. If $BestTour \neq NewBestTour$, go to step 2.

Agatz et al. (2018) tested heuristics including TSP-ep and TSP-ep-all, in which all refers to a neighborhood of routes that can be constructed using any local swaps described in their paper. TSP-ep-all considers $O(N^2)$ neighboring TSP routes in each iteration, so it has a total computational complexity of $O(IN^5)$, where I is the number of iterations.

Coutinho et al. (2016) considered the Close-Enough Traveling Salesman Problem (CETSP), which is a generalization of the TSP whereby a city is considered visited if the tour comes within a specified radius of the city. The key feature of Coutinho et al. (2016) is their branch-and-bound solution methodology, in which each node of the branch-and-bound tree is associated with some sequence of visit locations, S. At each node of the tree, a second-order cone program (SOCP) was solved that obeys the visit order dictated by S. Though the visit order is fixed, the SOCP is free to choose the optimal representative point for each visit location (a representative point is within the specified radius of a given city). To put it another way, the branch-and-bound structure served as a mechanism to globally search potential visit sequences. The SOCP that was solved at each node optimized the CETSP solution relative to this sequence. The solution method produced exact solutions to the CETSP.

Our solution method (described in detail in Section 4) borrows certain elements from Coutinho et al. (2016) and Agatz et al. (2018). Specifically, the branch-and-bound structure in our solution method is derived from Coutinho et al. (2016) and allows us to search various visit sequences. Rather than optimizing an SOCP at each node, we optimally partition the sequence at each node into truck-delivered and drone-delivered nodes. We slightly modify the partitioning procedure from Agatz et al. (2018) such that the truck may remain stationary while the drone makes a delivery.

3. Defining the TSP-D and Notation

3.1. TSP-D: Problem Definition

We define the TSP-D as follows. There is one truck and one drone that may ride atop the truck. Let c_t and c_d be matrices of travel times. $c_t(i, j)$ is the value of row i and column j of c_t, and it is set as the time a truck takes to traverse from node i to node j. $c_d(i, j)$ is the value of row i and column j of c_d, and it is set as the time a drone takes to traverse from node i to node j. For all i, j, $c_t(i, j), c_d(i, j) \geq 0$, and the triangle inequality holds for c_t and c_d. Frequently, in our computational experiments, both the truck and the drone operate on the Euclidean metric. In these cases, we define $\alpha = c_t(i, j)/c_d(i, j), \forall i, j$, which is a measure of the relative speed of the drone to the truck. In general, c_t and c_d need not be scalar

multiples of one another, in which case α is not defined.

There are N nodes, one depot, and $N-1$ packages to be delivered, and a set of locations (P) for the packages and the depot. $P_t \subseteq P$ is the set of locations such that the package at that location must be delivered by a truck. Some packages may not be suitable for drone delivery owing to complications such as excessive weight or an obstructed landing area. Let $P_d = P \backslash P_t$ be the set of package locations eligible for drone delivery. Each package $P_1, P_2, \ldots, P_{N-1}$ must be delivered either by truck or by drone. P_0 is the origin depot location, P_N is the destination depot location, and P_0 and P_N may be the same location.

The drone has a battery life of R time units. The drone may launch from atop the truck, carry a single package to a package delivery location, and then must rendezvous with the truck within R time units. The truck may deliver packages while the drone is airborne. Launch and rendezvous points must occur at package locations or the depot. The truck and drone do not need to arrive simultaneously; they can wait for each other to arrive, as long as the rendezvous happens within R time units of the drone's launch. In addition, a drone may be launched and retrieved at the same package location. We assume that after a drone lands on the truck, its battery may be swapped for a fully charged battery instantaneously.

For simplicity, we do not consider drone service times, drone launch overhead times, drone retrieval overhead times, or battery swap times, and we assume each time is negligible. However, it is possible to modify c_d and the computation of $T(i,j,k)$ in a small way to account for all of these times. In Part A of the online supplement, we describe how this can be done.

We must construct a simple tour (i.e., a tour that cannot depart a node and revisit that same node) beginning at depot P_0 and ending at depot P_N. If $P_0 = P_N$, the tour is closed. The departure time of the truck from P_0 is $t_0 = 0$, all packages $P_1, \ldots, P_{N-1}$ have been delivered and the truck and drone have returned to P_N at time t_f. The objective is to minimize t_f.

4. Branch-and-Bound Approach

We now describe our branch-and-bound approach (BAB) to the TSP-D. The pseudocode describing BAB is given in Part B of the online supplement.

4.1. Nodal Structure and Branching Procedure

Each node in our branch-and-bound decision tree is associated with some sequence of package locations, similar to Coutinho et al. (2016). If c_t and c_d are symmetric and $P_0 = P_N$, then we assign our root node an arbitrary 3-cycle including the depot, which can be done without loss of generality. Suppose we assign the sequence $[0,1,2,N]$ to the root node. This corresponds with a route that visits P_0, P_1, P_2, and returns to P_N in that order. If c_t and c_d are symmetric and $P_0 = P_N$, then the routes corresponding to $[0,1,2,N]$ and $[0,2,1,N]$ have the same objective value. In the case that c_t and c_d are not symmetric or $P_0 \neq P_N$, we assign the root node the sequence $[0,1,N]$. Although it is possible to assign a symmetric instance $[0,1,0]$ as the root node, we choose $[0,1,2,0]$ as the root. We take advantage of known symmetry and avoid the formation of two branches with $[0,1,2,0]$ and $[0,2,1,0]$, which unnecessarily doubles the computation time. For simplicity of notation, assume $P_0 = P_N$ unless specified otherwise.

Our tree begins with only the root node. We then create children of the root node. Find the package location P_i that is farthest (in Euclidean distance) from any package location in the parent's sequence. Suppose that the farthest package location from package locations P_0, P_1, and P_2 is package location P_3. Then the children of the root node $[0,1,2,0]$ are $[0,3,1,2,0]$, $[0,1,3,2,0]$, and $[0,1,2,3,0]$. Our branching procedure inserts the farthest package into various positions of the parent node's sequence.

We represent a sequence by $S = [0, s_1, \ldots, s_n, 0]$, where n is the number of package locations visited in the sequence, and s_i is the package location in position i of the visit sequence.

4.2. Lower Bound Evaluation for a Node

Suppose *TSPSeq* is the optimal TSP sequence. Let *aep*(*TSPSeq*) be the objective value of TSP-ep from Agatz et al. (2018) when we apply the exact partitioning function (*aep*) of Agatz et al. (2018) to the input sequence denoted by *TSPSeq*. Let *aep*(S) denote the objective value produced by applying the exact partitioning procedure to an input sequence of package locations, S.

In the *aep* function, drone operations (i,j,k) are considered only if $i<k<j$ (although, elsewhere in Agatz et al. 2018 this restriction is relaxed). Therefore, launching and retrieving a drone at the same customer node with the truck remaining stationary is impossible in any solution produced by *aep*, even though this may be characteristic of the optimal solution. Let *ep* denote an exact partitioning function that incorporates the possibility of the truck remaining stationary throughout the drone's flight into *aep*. The technical details of *ep* are given in Part C of the online supplement.

Suppose some node has an associated sequence $S = [0, s_1, s_2, \ldots, s_n, 0]$. At this node, we seek to find a partition of S into an ordered set of packages delivered by the truck (S_t) and an ordered set of packages delivered by the drone (S_d). However, rather than requiring S_t and S_d to be subsequences of a specified TSP solution, we require that S_t and S_d be subsequences of S. Thus, if a node has associated sequence S, it has an associated objective value *ep*(S). The result is the optimal TSP-D objective value for delivering

packages $s_1, s_2, \ldots, s_n$ subject to the constraint that S_t and S_d are subsequences of S. For a node with sequence S, we say that is has an assumed lower bound of $ALB(S) = ep(S)$.

Suppose some parent node has a sequence S. Suppose its child has a sequence S^+, which is necessarily a supersequence of S. Our algorithm works under the assumption that the insertion of additional package stop locations onto a TSP-D route generally increases the objective value, that is, we assume that $ep(S^+) \geq ep(S) = ALB(S)$. Thus, for a parent node with sequence $S = [0, s_1, s_2, \ldots, s_n, 0]$, we assume that the objective value of any child node is at least $ALB(S)$. Transitively, any direct descendant node is assumed to have a larger objective value than its ancestor.

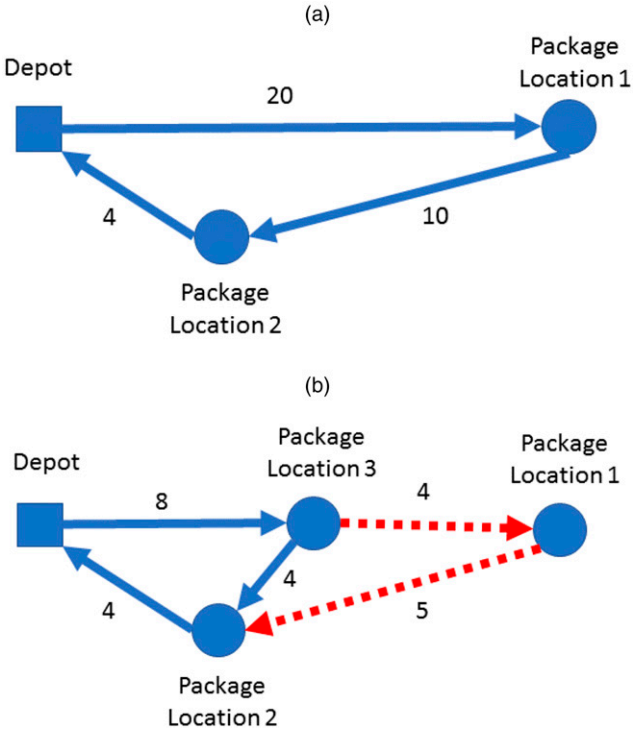
The insertion of additional package locations onto the TSP-D route can actually decrease the objective value (unlike TSP routes, where package insertions can never decrease the objective value), that is, occasionally $ep(S^+) < ep(S)$. This is directly related to the finite range of the drone. Including additional stops in the route introduces new locations where a drone could potentially launch or land. Thus, a package that would have been delivered by a truck (due to the lack of any launch or landing locations suitable for the range of the drone) could potentially be delivered by the drone, after a new stop location is added to the route. In Figure 1, we provide an example in which inserting an additional package location onto the route decreases the objective value. In this example $R = 10$, blue edges represent truck movement, red edges represent drone flight, and the number next to each edge is the time required to traverse the edge. Numbers next to blue edges are driving times in minutes; numbers next to red-dashed edges are flying times in minutes. In Figure 1(a), the truck drives to package location 1, because the drone only has 10 minutes of battery life. The drone does not have enough battery life to fly to package location 1, make a delivery, and return to the truck. The completion time is 20 + 10 + 4 = 34 minutes. In Figure 1(b), the insertion of package location 3 gives the drone a feasible launching point to deliver to package location 1. The drone launches from package location 3, makes a delivery to package location 1, and lands on the truck at package location 2 with a completion time of 8 + 9 + 4 = 21 minutes.

Although the insertion of a new package location onto a TSP-D route occasionally decreases the objective value, in testing we found that it is more likely that inserting additional package locations increases the objective values.

4.3. Exploration, Upper Bounds, and Terminating the Algorithm

If a parent node with sequence $[0, s_1, s_2, \ldots, s_n, 0]$ has been evaluated and its children have not yet been evaluated, then the lower bound of the parent node is $ALB([0, s_1, s_2, \ldots, s_n, 0])$. If all children of a parent have been evaluated, then the lower bound of the parent node is equal to the smallest lower bound of the children.

Figure 1. (Color online) Example in Which the Insertion of an Additional Package Location Decreases the Objective Value



Among all nodes whose children have not yet been evaluated (leaf nodes), we iteratively choose to evaluate the children of the node with the smallest lower bound. If a sequence contains all N package locations, then it is marked as a feasible solution to the overall problem. The assumed lower bound for that node is also an upper bound.

Let LB be the smallest lower bound among nodes that still have unexplored children, and let UB denote the best objective value found for any complete feasible solution (this solution contains all package locations). If no complete feasible solution has yet been found, then $UB = \infty$. We terminate the branch-and-bound algorithm once $LB/UB \geq TER$, where $TER \geq 1$ is our tree exploration ratio. If our assumed lower bound was always a valid lower bound, then setting $TER = 1$ would yield the globally optimal solution. Because our assumed lower bound is sometimes too high, we may compensate for this by setting $TER > 1$.

For example, suppose we set $TER = 1.15$ and find a complete feasible solution with objective value of 100. If we did not find a new complete feasible solution with objective value less than 100, then our algorithm would terminate when all nodes with lower bounds less than 115 had been explored. Thus, our solution is globally

optimal, if we did not overestimate the lower bound of any node by more than $0.15(UB)$.

An alternate, but logically equivalent, interpretation is that the assumed lower bound of a node with the associated sequence $[0, s_1, s_2, \ldots, s_n, 0]$ is given by $ALB([0, s_1, s_2, \ldots, s_n, 0])/TER$ and we terminate the algorithm when $LB \geq UB$.

After we explore a feasible solution, we set its lower bound to INF. We define INF as any value greater than $N \times maxEdge$, where $maxEdge = \max_{i,j} c_t(i,j)$. The value $N \times maxEdge$ serves as an upper bound to the objective value of any feasible sequence partitioned by ep. If $TER = \infty$, BAB terminates when the root node's lower bound is equal to INF, which occurs only when all feasible solutions have been explored.

4.4. Example of the Branch-and-Bound Approach

In Figure 2, we begin with evaluating the root node that produces an objective value of 80. We then evaluate all of its children, and the child with sequence [0,1,2,3,0] has the lowest objective value of 85, so we then evaluate its children. Among all leaf nodes of the tree, the one with sequence [0,3,1,2,0] has the lowest objective value of 96, and we would explore its children next.

In Figure 3, we display an example with four package locations in addition to the depot. The full exploration of the branch-and-bound tree when $TER = 1.00$ is shown. If $TER = 1.15$, then we evaluate the children of the red node with objective value of 112, because it has an objective value less than $1.15 \times 100 = 115$.

4.5. Reduction to $O(Cn^2)$

When computing $T(i,j,k)$ for each $i < k < j$, we have an $O(n^3)$ computation. Because this computation occurs for each node visited in the branch-and-bound tree, this computation becomes very costly. Therefore, before starting the branch-and-bound approach, we compute a constant C associated with c_t, the truck network metric. C is the largest integer such that a truck may visit C distinct nodes on the street network within R time units.

Now, we need only compute $T(i,j,k)$ for each $i < k < j \leq i + C$. We need not compute any potential drone deliveries for $j \geq i + C + 1$. Suppose $j \geq i + C + 1$. There are at least $C + 2$ nodes between i and j, inclusive of the endpoints. If some node k is serviced by the drone, then at least $C + 1$ nodes must be visited by the truck within time R before the drone battery loses its charge. However, C is the maximum number of nodes visited by the truck in time R. Computing $T(i,j,k)$ is unnecessary whenever $j \geq i + C + 1$ owing to infeasibility associated with the battery life of the drone.

Therefore, we have reduced the computational complexity of each nodal evaluation from $O(n^3)$ to $O(Cn^2)$ at the cost of computing C once before starting

Figure 2. (Color online) Initial Exploration of a Branch-and-Bound Tree with Associated Sequences and Objective Values in Parentheses

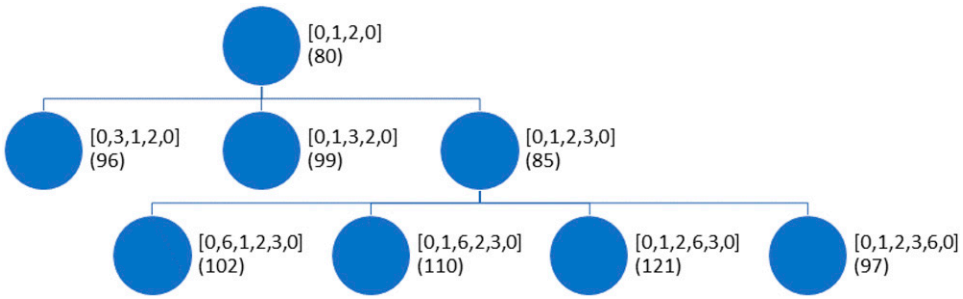
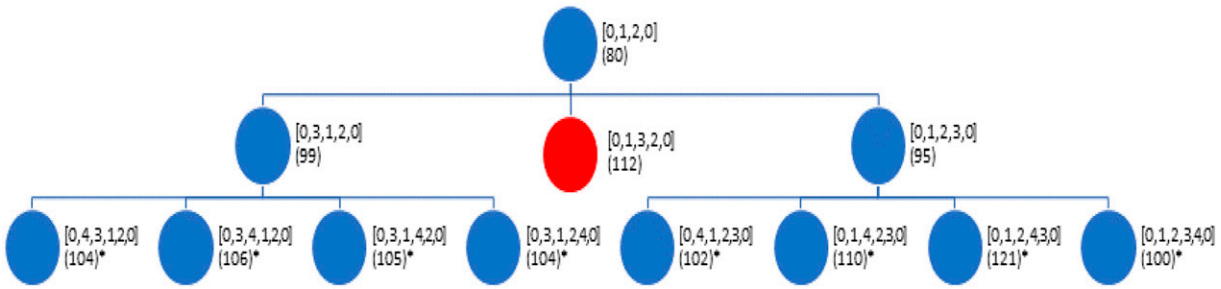


Figure 3. (Color online) Full Exploration Tree for BAB when $TER = 1.00$



Notes. Associated sequences are in brackets, objective values are in parentheses, and an asterisk indicates a feasible solution that visits all package locations. As shown, $LB = 112$ and $UB = 100$, so the heuristic terminates, because $LB/UB = 1.12 > 1.00$. The node with objective value 112 has unevaluated children.

the BAB procedure. C may be computed exactly by solving the integer program given in Part D of the online supplement. In areas of low density and little clustering of delivery locations, we expect C to be small. Alternatively, we could compute an upper bound on C, denoted C^+, by summing the smallest elements of c_t until the sum exceeds R. The number of distinct elements that may be summed before exceeding R is C^+. Then we only compute $T(i,j,k)$ for each $i<k<j\le i+C^+$.

5. Additional Heuristics for the TSP-D

5.1. Boosted Lower Bound Heuristic

In practice, the computation time required to perform the branch-and-bound algorithm is heavily dependent on the ability to prune large portions of the decision tree. BAB was built on the assumption that as more package locations are inserted into a sequence, the objective values associated with that sequence strictly increase. Thus, among two sequences with the same associated objective value, we prefer the longer sequence, because it has fewer packages that need to be inserted to form a feasible solution.

Consider the sequence $[0,s_1,s_2,\ldots,s_n,0]$ that does not visit $N-1-n$ package locations that are required for a full global solution. We define our heuristic lower bound (HLB) as

$$HLB([0,s_1,s_2,\ldots,s_n,0]) = ALB([0,s_1,s_2,\ldots,s_n,0]) + f(N-1-n),$$

where f is an increasing function and $f(0)=0$.

Our boosted lower bound heuristic uses the original branch-and-bound structure and branching process. However, we replace the objective value of each node and, thus, the lower bound on all descendant nodes ALB with HLB. When we find a feasible solution at some node, $N-1=n$, $f(N-1-n)=f(0)=0$. Thus, for any feasible sequence S with all package locations, $HLB=ALB$.

5.1.1. Linear Boost Heuristic. Let $f(N-1-n)=\gamma(N-1-n)$, where $\gamma>0$ is a specified constant. Our linear boost heuristic assumes that, for any sequence, the insertion of additional packages into that sequence will cost at least an additional γ time units per package on average.

Consider the example in Figure 2. The next step in BAB evaluates the children of the node with sequence [0,3,1,2,0] and objective value 96. Suppose that $N=11$. The linear boost heuristic (denoted by BAB+L for the BAB method plus an additional linear term) with $\gamma=2$ has a lower bound of $96+2(7)=110$ for the sequence [0,3,1,2,0], because there are seven package locations that have not yet been inserted into the sequence. The node with sequence [0,1,2,3,6,0] has a lower bound of $97+2(6)=109$, because there are six remaining package locations to be inserted into the sequence. BAB+L would evaluate the children of the node with sequence [0,1,2,3,6,0] next.

5.1.2. Quadratic Boost Heuristic. Our branching procedure is similar to farthest insertion. Those package locations farthest away are inserted before those package locations that are nearest to the existing subtour. The second variant of our heuristic assumes that the marginal cost per package insertion is larger for short sequences. As the subtour grows and iteratively inserts the package that is farthest away, the marginal insertion cost is assumed to decrease.

Rather than using $f(N-n)=(N-n)\gamma$, which is linear in the number of package locations not yet inserted (i.e., constant marginal insertion cost of γ), we use $f(N-n)=(N-n)^2\gamma$, which is consistent with decreasing marginal insertion costs as n increases. We denote this method by BAB+Q.

5.2. Divide-and-Conquer Heuristic

For BAB, BAB+L, and BAB+Q, computation times increase superlinearly (see Section 6). For large problem sizes, we consider a different heuristic method that we call the divide-and-conquer heuristic (denoted by DCH).

Let $CT(N)$ be the average computation time for BAB for instances of size N. Because $CT(N)$ increases superlinearly, $mCT(N/m)<CT(N)$. Thus, we expect the computation time of solving m problems of size N/m to be less than the computation time of solving a single larger problem of size N.

DCH begins by solving the standard TSP on the truck metric. We relabel nodes according to their order of appearance in the standard TSP solution. The first node visited in the standard TSP solution is relabeled node 1; the second node visited in the standard TSP solution is relabeled node 2; generally, the ith node visited in the standard TSP solution is relabeled node i. Node 0 and node N may be identical and serve as the origin and destination depots.

Next, we split the relabeled nodes from the TSP solution into m groups. The first group has nodes $0,1,2,\ldots,\lfloor N/m\rfloor$. The second group has nodes $\lfloor N/m\rfloor, \lfloor N/m\rfloor+1,\ldots,\lfloor 2N/m\rfloor$. Generally, group i has nodes $\lfloor (i-1)N/m\rfloor, \lfloor (i-1)N/m\rfloor+1,\ldots,\lfloor iN/m\rfloor$. Thus, the node set is divided into m groups whereby each group has a size of $\lfloor N/m\rfloor$ or $\lfloor N/m\rfloor+1$.

For each of the m groups, we solve a subproblem. In particular, we solve a TSP-D on the set of nodes in each group with a condition. For group i, we set the root node sequence to $[\lfloor (i-1)N/m\rfloor, \lfloor iN/m\rfloor]$. Node $\lfloor (i-1)N/m\rfloor$ acts as the origin depot for this subproblem; node $\lfloor iN/m\rfloor$ acts as the destination depot for this subproblem. Then each subproblem is solved using

BAB. Any node j such that $\lfloor (i-1)N/m \rfloor < j < \lfloor iN/m \rfloor$ is inserted between the origin and depot nodes on subproblem i. The full problem solution is the union of the solutions of all subproblems in order.

In Figure 4, we give an example with $N = 30$ nodes and $m = 3$. A standard TSP route has already been specified, and nodes have been relabeled accordingly. Subproblem 1 requires that the truck and drone start at node 0 and service nodes 1 through 9 in some order. After servicing nodes 1 through 9, the truck and drone must rendezvous at node 10. Subproblem 2 requires that the truck and drone start at node 10, service nodes 11 through 19 in some order, then rendezvous at node 20. Subproblem 3 requires that the truck and drone start at node 20, service nodes 21 through 29 in some order, then rendezvous at node 30. Combining the solution to all subproblems produces a solution to the full problem with $N = 30$.

The intuition behind DCH is that the truck route in a good TSP-D solution may have a similar broad shape to the optimal TSP solution. By solving each subproblem, we optimize the local structure. In Figure 4, we have the flexibility to rearrange the order of nodes 1 through 9, 11 through 19, and 21 through 29. By solving m subproblems, we reduce computation times significantly.

6. Computational Results

All instances were created by randomly generating N locations on a 50 by 50 grid where the coordinates were distributed uniformly in each of the two dimensions. One of the N locations was randomly designated as the depot. All computations were performed on a computer with an i7-6700 processor operating at 3.4 GHz, 16 GB of RAM, and no parallelization. All computation times are reported in seconds. For DCH and the TSP-ep method (Agatz et al. 2018), an optimal standard (truck-only) TSP solution was used as input. Computation times for DCH and the TSP-ep method do not include the time required to compute the solution to the standard TSP. Instance data may be found online at http://stefan-poikonen.net/tspd_instance_data.zip.

In Tables 1 and 2, we see the apparent convergence of objective values when we increase *TER*. In Table 3, we compare BAB with TSP-ep and TSP-ep-all on the benchmark instances of Agatz et al. (2018). In Table 4, the objective values and computation times of five solution methods are reported. In Tables 5 and 6, we vary γ in the BAB+L and BAB+Q heuristics. In Table 7, we display a trade-off of computation time versus solution quality for DCH by changing the number of subproblems. In Table 8, we study the effect of changing R and α. In Table 9, we consider the choice of alternative truck and drone metrics.

6.1. BAB Results for Different Tree Exploration Ratio Values

In Table 1, we generated 100 random instances with $N = 10$ and solved each instance with BAB using various values of *TER*. By setting $TER = \infty$, we enumerate the entire branch-and-bound tree and obtain the optimal solution to each instance. In the column Obj, the average objective value over the 100 instances for a specified value of *TER* is given. The column Gap (Opt) is (Obj-Opt)/Opt, where Opt is the average objective value of the 100 optimal solutions. The column Time gives the computation time (in seconds) required on

Figure 4. (Color online) Divide and Conquer Heuristic with $N = 30$ and $m = 3$

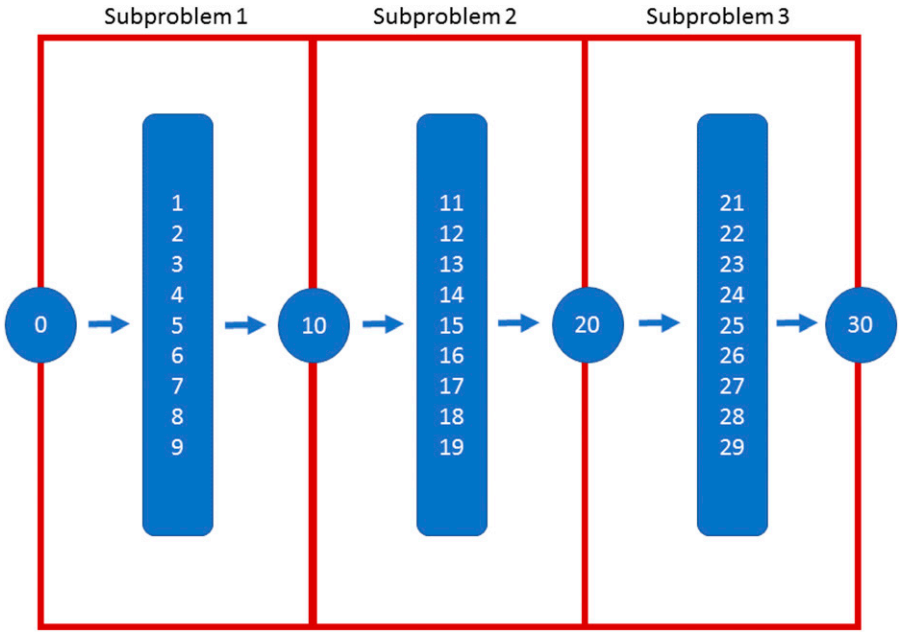


Table 1. Computational Results for BAB with $N = 10$

TER	Obj	Gap (Opt)	Optimal	TSP-ep better	Time (s)
1.000	153.243	0.032	36	11	0.031
1.025	152.400	0.027	44	9	0.041
1.050	151.254	0.019	55	6	0.064
1.075	151.048	0.017	58	5	0.104
1.100	150.614	0.014	64	5	0.164
1.125	150.067	0.011	75	3	0.239
1.150	149.757	0.009	81	3	0.331
1.175	149.335	0.006	87	2	0.449
1.200	148.914	0.003	90	1	0.594
1.225	148.823	0.002	93	1	0.794
1.250	148.673	0.001	94	0	1.013
1.275	148.661	0.001	95	0	1.281
1.300	148.518	0.000	97	0	1.568
1.325	148.517	0.000	98	0	1.989
1.350	148.517	0.000	98	0	2.443
∞	148.462	0.000	100	0	77.835
TSP-ep	159.21	0.103			0.003
TSP	186.24	0.210			0.001

Note. Obj, objective value; Gap (Opt), gap from optimal solution value.

Table 2. Computational Results for BAB with $N = 15$

TER	Obj	Gap (Best)	Best solution	TSP-ep better	Time (s)
1.000	164.446	0.055	8	1	0.300
1.025	163.820	0.051	9	1	0.468
1.050	161.708	0.038	11	0	1.096
1.075	160.486	0.030	15	0	2.904
1.100	159.987	0.027	19	0	6.057
1.125	159.319	0.023	23	0	12.294
1.150	158.375	0.017	31	0	23.024
1.175	157.258	0.009	37	0	36.851
1.200	156.284	0.003	44	0	62.069
1.225	156.115	0.002	47	0	107.165
1.250	155.804	0.000	50	0	175.542
TSP-ep	183.821	0.180			0.003
TSP	214.309	0.376			0.001

Note. Gap (Best), gap from best solution value.

average for a specified value of *TER*. In the column Optimal, we show the number of instances where the value of *TER* produced the optimal solution. The number of instances where TSP-ep produced a better objective value than BAB for a specified value of *TER* is shown in the column TSP-ep Better. In the bottom two rows, we show the average objective value, average gap from the optimal TSP-D solution, and average computation times for TSP-ep and the standard TSP. As the value of *TER* increases, the objective value of BAB converges. When *TER* reaches a value of 1.250, the objective value of BAB is less than or equal to the objective value of TSP-ep in all 100 instances. In 8 of the 100 instances, TSP-ep produced the optimal solution.

In Table 2, we randomly generated 50 instances with $N = 15$. Computation times were intractably large for $TER = \infty$ (i.e., not a single instance was solved after several hours of testing). In the column Best Solution, we show the number of instances in which the value of *TER* produced the lowest objective value among all solution methods that were tested. Because we do not know the optimal solution, the column Gap (Best) shows the average value of (Obj-Best)/Best, where Best is the lowest objective produced by any method tested. BAB had an objective value less than or equal to the objective value of TSP-ep in all instances with $TER > 1.025$. The objective values seem to be converging as *TER* increases.

In Figure 5, we show an example in which TSP-ep fails to find the optimal solution and BAB finds the optimal solution. In this example, $\alpha = 2$ and distances between package locations are 10, except along the diagonals where the distance is $10\sqrt{2}$. The TSP-ep solution begins by launching a drone to package location 1, while the truck drives to package location 2. By the time the truck arrives at package location 2, the drone has already delivered a package at location 1 and is ready to land on the truck. The drone is launched again to package location 3, while the truck returns to package location 0. The drone will rendezvous with the truck at package location 0. The BAB solution initially launches a drone to package location 2 and sends the truck to package location 1. The truck waits for $(10 + 10\sqrt{2})/2 - 10$ time units at package location 1 for the drone to arrive. The drone is then launched to package location 3 to make a delivery and will eventually rendezvous with the truck at package location 0. The solution produced by BAB could not occur with TSP-ep, because the statement $0 < 2 < 1$ is not valid. In TSP-ep, only drone operations for triplets (i, j, k) where $i < k < j$ are considered. Thus, a drone operation beginning at package location 0 and ending at package location 1 could not launch a drone to package location 2.

In Table 3, we report the results for BAB, TSP-ep, and TSP-ep-all on the instances with $N = 10$ and $\alpha = 2$ that were solved in Agatz et al. (2018). There were 10 random instances of three types: uniform, 1-center, and 2-center. Uniform instances distributed package locations uniformly over a square grid. In 1-center instances, the distance of a package location from the center was distributed normally with standard deviation 50 and the angle relative to the grid was distributed uniformly over $[0, 2\pi]$. The 2-center instances were generated in the same way as 1-center instances, except that package locations were shifted horizontally by 200 with probability 0.5. In the columns labeled Opt, we report the number of instances (out of 10) that were solved optimally. In the columns labeled Gap, we report how much the objective value exceeded the optimal solution on average. We found that BAB performed best on 1-center instances and worst on

Table 3. Computational Results on Instances with $N = 10$ and $\alpha = 2$ from Agatz et al. (2018)

	Uniform instances		One-center instances		Two-center instances	
TER	Opt	Gap	Opt	Gap	Opt	Gap
1.000	2	0.055	3	0.009	5	0.013
1.025	3	0.037	5	0.007	6	0.009
1.050	5	0.024	6	0.002	7	0.006
1.075	6	0.021	9	0.001	9	0.002
1.100	6	0.016	9	0.001	9	0.002
1.125	7	0.013	10	0.000	9	0.002
1.150	8	0.010	10	0.000	9	0.002
1.175	9	0.009	10	0.000	9	0.002
1.200	9	0.003	10	0.000	9	0.002
1.225	9	0.003	10	0.000	10	0.000
1.250	9	0.003	10	0.000	10	0.000
1.275	10	0.000	10	0.000	10	0.000
1.300	10	0.000	10	0.000	10	0.000
∞	10	0.000	10	0.000	10	0.000
TSP-ep	0	0.160	0	0.152	0	0.127
TSP-ep-all	6	0.004	5	0.011	5	0.013

uniform instances. One possible reason for relatively bad performance on uniform instances may be related to the fact that these are the least-clustered instances. For a specific delivery location, the set of potential launch points for the drone may be especially limited in these instances. The insertion of additional stop locations into a sequence may more frequently decrease the objective value, relative to 1-center or 2-center instances. We note that, in Section 4.2, we described how an objective value can decrease by adding stop locations.

6.2. Solution Quality and Computation Time Results for Five TSP-D Solution Methods

In Table 4, we give the results for five methods and the optimal TSP solution with $R = 20$, $\alpha = 2$, and $N = 10, 20, \ldots, 90, 100, 200$. The truck follows the taxicab metric while the drone follows the Euclidean distance metric and travels at a speed twice as fast as the truck. Each method used the same set of 25 instances for each value of N. *TER* is set at 1.05 for BAB, BAB+L, BAB+Q, and all subproblems in DCH. Each row gives the average results for 25 randomly generated instances for a value of N. Obj gives the average objective value, and Time (s) gives the average solution time in seconds.

In BAB+L and BAB+Q, we set the parameter $\gamma = 5.0$ and $\gamma = 5.0/N$, respectively. In DCH, the number of subproblems is defined by $m = N/10$, so that the subproblem size remains constant at 10 regardless of N. BAB, BAB+L, and BAB+Q have computation times that grow quickly. In DCH, by keeping subproblem size constant at 10, computation time grows linearly with N. TSP-ep is an $O(N^3)$ method. BAB produced the best objective values, but it was the slowest method. DCH had objective values that, on average, were smaller than TSP-ep for every value of N. TSP-ep was the fastest method for every instance.

6.3. Linear and Quadratic Boost Heuristics Trade-off

BAB+L and BAB+Q use the input parameter γ. In Tables 5 and 6, we show a trade-off between objective value and computation time. We generated 25 random instances with size $N = 20$ and constant parameter values $R = 20$, $\alpha = 2$, and $TER = 1.00$. The objective values and computation times were averaged over all 25 instances. In Tables 5 and 6, larger values of γ had smaller computation times and produced worse objective values, on average. We point out that, when $\gamma = 0$ and $N\gamma = 0$, we have the same results as BAB. The column Gap in Tables 5 and 6 is computed by (Obj-BAB)/BAB, where BAB represents the objective value found by setting $\gamma = 0$.

Table 4. Computation Time and Objective Value Averages for Five Methods and the Objective Value for the Optimal TSP Solution

	BAB		BAB+L		BAB+Q		DCH		TSP-ep		TSP
N	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)	Obj	Time (s)	Obj
10	149.532	0.066	154.287	0.022	152.340	0.036	149.532	0.068	159.760	0.002	176.662
20	171.642	58.726	185.495	2.145	180.675	11.969	182.230	0.180	197.785	0.003	237.681
30	—	—	209.112	8.095	—	—	200.945	0.308	215.904	0.005	277.929
40	—	—	239.345	28.906	—	—	226.153	0.908	250.627	0.007	319.502
50	—	—	—	—	—	—	241.360	1.818	276.284	0.011	352.407
60	—	—	—	—	—	—	267.539	2.973	301.867	0.018	382.892
70	—	—	—	—	—	—	283.304	3.080	316.564	0.026	407.699
80	—	—	—	—	—	—	299.092	3.685	339.768	0.036	438.725
90	—	—	—	—	—	—	322.370	5.269	362.058	0.050	464.001
100	—	—	—	—	—	—	337.906	5.797	377.677	0.066	486.096
200	—	—	—	—	—	—	465.627	14.000	523.734	0.486	666.792

Note. A dash (—) indicates that the 25 instances could not be solved within five hours.

Table 5. Trade-off Between Solution Quality and Computation Time for BAB+L for $N = 20$

γ	Obj	Gap	Time (s)
0	173.56	0.000	9.493
2	177.80	0.024	1.590
4	181.60	0.046	0.571
6	184.81	0.065	0.399
8	186.74	0.076	0.271

Table 6. Trade-off Between Solution Quality and Computation Time for BAB+Q for $N = 20$

$N\gamma$	Obj	Gap	Time (s)
0	173.56	0.000	9.493
2	176.67	0.017	4.770
4	178.19	0.027	3.023
6	178.74	0.030	1.426
8	179.55	0.035	0.986

6.4. DCH Trade-off

When $m = 1$, DCH is equivalent to BAB and when $m = N$, DCH produces the truck-only TSP solution. In Table 7, an intermediate number of subproblems is considered where $N = 48$. We set $R = 20$, $\alpha = 2$, and $TER = 1.00$, and average the results from 25 instances. In Table 7, m is the number of subproblems, N/m is the average size of each subproblem, Obj is the average objective, and Time is the average computation time in seconds. There is a clear trade-off: solving many small subproblems is computationally faster, but objective values are worse. Large values of m create a more constrained problem that is anchored at $m + 1$ points of the initial TSP solution. Anchor points are package locations that occur as either the first node or last node visited in one of the subproblems. In Figure 4, package locations 0, 10, 20, and 30 are anchor points. Furthermore, all nodes of group i must be visited before any nodes of group $i + 1$. In Figure 4, this means package locations 1 through 9 are serviced before package locations 11 through 19; package locations 11 through 19 are serviced before package locations 21 through 29. Small values of m provide more solution flexibility but suffer from slower computation times.

Table 7. Trade-off Between Solution Quality and Computation Time for DCH for $N = 48$

N/m	m	Obj	Time (s)
4	12	280.85	0.011
6	8	261.94	0.039
8	6	250.49	0.132
12	4	245.74	2.031
16	3	240.15	45.366
24	2	237.28	512.213

Table 8. Drone Battery and Speed vs. TSP-D Objective Value for $N = 48$

R	α	Obj
10	0.5	347.99
10	1.0	333.57
10	2.0	286.22
10	3.0	256.84
20	0.5	345.33
20	1.0	295.69
20	2.0	240.90
20	3.0	224.51
30	0.5	337.82
30	1.0	279.44
30	2.0	232.33
30	3.0	219.67
TSP		348.06

6.5. Effect of Drone Battery Duration and Speed on the TSP-D Solutions

We consider the effects of drone battery life and drone speed on the solution to the TSP-D. In Table 8, 25 instances were generated with $N = 48$. Each instance was solved by DCH with $R = 10, 20, 30$, $\alpha = 0.5, 1.0, 2.0, 3.0$, $TER = 1.00$, and $m = N/10$. The average TSP objective value over the 25 instances is 348.06.

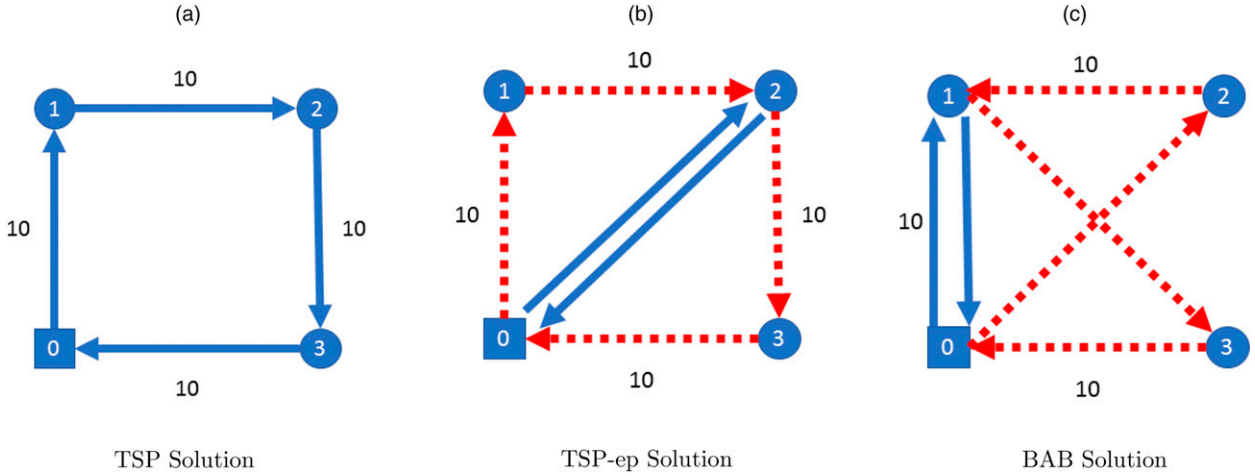
Larger values of drone range and faster speeds produced smaller objective values for the TSP-D. By adding a very low performance drone, the improvement in objective value is typically very small. For example when $R = 10$ and $\alpha = 0.5$, the performance improvement was only 0.02% compared with the TSP solution. In contrast, a high-performance drone ($R = 30$ and $\alpha = 3$) produced TSP-D solutions with objective values 36.89% lower than the TSP solution.

$R\alpha$ is the range of the drone in units of distance. If we compare two sets of parameters with equal values of $R\alpha$, such as $R = 30$ and $\alpha = 1.0$ versus $R = 10$ and $\alpha = 3.0$, the set of parameters with a larger value of α produced a smaller objective value in each case. This indicates that for two drones with equal range (in distance units), the drone with larger speed is usually more valuable than the drone that is capable of hovering for a long period of time to preserve feasibility of certain operations.

Table 9. Comparison of TSP and TSP-D Results for Three Different Metrics

	TSP-D Taxi/Euc		TSP-D Euc/Euc		TSP Taxi
N	Obj	Improve	Obj	Improve	Obj
12	165.02	−0.216	132.60	−0.370	210.38
24	198.76	−0.304	161.36	−0.435	285.47
36	232.93	−0.310	190.87	−0.434	337.51
48	263.17	−0.312	218.64	−0.428	382.30
60	290.71	−0.316	241.23	−0.432	425.30

Figure 5. (Color online) An Example in Which BAB Outperfoms TSP-ep



Notes. In (a), the TSP solution has an objective value of 40. In (b), the TSP-ep solution has an objective value of $20\sqrt{2} \approx 28.28$. In (c), the BAB solution has an objective value of $10 + 10\sqrt{2} \approx 24.14$.

6.6. The Effect of Distance Metrics on the TSP-D Solutions

In Table 9, we consider the effect of different distance metrics on objective values. For each size N, 25 instances were generated, and the average objective values over the 25 instances are reported.

In the column TSP-D Taxi/Euc, we give the TSP-D objective value with c_t defined by the taxicab distance and c_d defined by the Euclidean distance divided by two. In the column TSP-D Euc/Euc, we give the TSP-D objective value with c_t defined by the Euclidean distance and c_d defined by the Euclidean distance divided by two (i.e., $\alpha = 2$). TSP Taxi gives the optimal objective value of the standard TSP using the taxicab distance. DCH was used for TSP-D Taxi/Euc and TSP-D Euc/Euc with $m = N/12$ and $R = 20$. Improve gives the average reduction in objective value in relative terms compared with TSP Taxi. We see that, for all instance sizes except $N = 12$, TSP-D Taxi/Euc has an average objective value that is more than 30% less than TSP Taxi. If the truck is free to move in Euclidean space, the average completion time reduction exceeds 40% except when $N = 12$.

7. Conclusions and Future Work

In this paper, we presented four heuristics for the TSP-D based on the branch-and-bound algorithm. For smaller instances, we showed that increasing the value of *TER* with BAB leads to the convergence of objective values. This suggests that BAB may generate solutions that are very close to the optimal solution when *TER* is sufficiently large. For larger instances, DCH produced objective values that compared favorably to TSP-ep. Although TSP-ep produced the smallest computation time in all instances, DCH had an average computation time of less than 15 seconds for the largest instances ($N = 200$). Because DCH can be solved in a reasonable amount of time on problems of practical size, DCH might be useful to drone delivery services. Additional computational experiments analyzed the effect of input parameters. We showed that when the truck was constrained to the taxicab metric, a single drone with battery life of 20 minutes and double the speed of the truck produced very significant savings, often in excess of 30%.

In future work, we hope to consider variants of the TSP-D, including allowing more than one drone per truck and allowing drones to launch or land along an edge in addition to package stop locations. We want to model the overhead time required for each drone launch or landing and want to add an extra cost factor to the objective for each drone launch. We also want to consider embedding the TSP-D in a vehicle routing problem with multiple trucks. Because TSP-D produced objective values nearer to optimal on 1-center instances than on uniformly distributed instances, we want to consider the impact of customer distribution on the TSP-D and related solution methods.

References

Agatz N, Bouman P, Schmidt M (2018) Optimization approaches for the traveling salesman problem with drone. *Transportation Sci.* 52(4):965–981.

Bellman R (1958) On a routing problem. *Quart. Appl. Math.* 16(1):87–90.

Busacker R, Saaty T (1965) *Finite Graphs and Networks: An Introduction with Applications* (McGraw-Hill, New York).

Cary N, Bose N (2016) UPS, FedEx and Amazon gather flight data to prove drone safety. Accessed May 17, 2017, https://venturebeat.com/2016/09/24/ups-fedex-and-amazon-gather-flight-data-to-prove-drone-safety/.

Coutinho WP, et al. (2016) A branch-and-bound algorithm for the close-enough traveling salesman problem. *INFORMS J. Comput.* 28(4):752–765.

Dynamic Parcel Distribution (2017) DPD group drone delivers parcels using regular commercial line. Accessed May 17, 2017, https://www.dpd.com/home/news/latest_news/dpdgroup_drone_delivers_parcels_using_regular_commercial_line.

Ha QM, Deville Y, Pham QD, Hà MH (2015) Heuristic methods for the traveling salesman problem with drone. Technical report, ICTEAM, Université catholique de Louvain, Louvain-la-Neuve, Belgium.

Knight R (2016) Drones deliver healthcare. Accessed May 17, 2017, http://insideunmannedsystems.com/drones-deliver-healthcare/.

Murray C, Chu A (2015) The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Res. Part C: Emerging Tech.* 54(May):86–109.

Poikonen S, Wang X, Golden B (2017) The vehicle routing problem with drones: Extended models and connections. *Networks* 70(1):34–43.

Reuters (2015) Finnish post office tests drone for parcel delivery. *Reuters* (September 14), http://www.reuters.com/article/us-finland-postaldrone-idUSKCN0RE15E20150914.

UPS YouTube Channel (2017) UPS tests residential delivery via drone. Accessed May 15, 2017, https://www.youtube.com/watch?v=%5C%5Cxx9%5C_6OyjJrQ.

Wang X, Poikonen S, Golden B (2017) The vehicle routing problem with drones: Several worst-case results. *Optim. Lett.* 11(4):679–697.