MATH3205 PROJECT REPORT

An Exact Algorithm for the Heterogeneous Drone-Truck Routing Problem

Oscar Davine - 45285714, Haolong Du - 46272784

Contents

1. Introduction to the Problem

The problem that is investigated in the chosen paper is the Heterogeneous Drone-Truck Routing Problem (HDTRP). The HDTRP is a variant of the travelling salesman problem that comprises one truck and multiple drones. Each customer node must be visited by either a truck or a drone with the truck leaving from and returning to the same source / sink node, called a depot. Drones can only be launched from the truck at a node with the time spent waiting at that node being equal to the longest time a drone is spent delivering to other nodes. The paper considered that each drone could have unique speed and battery factors and the obvious objective is to minimise the truck travel time + waiting time. The authors of the paper considered that an algorithm to solve the HDTRP could have applications in logistics with the combination of truck and drones being considered as a feasible delivery method for low weight items.

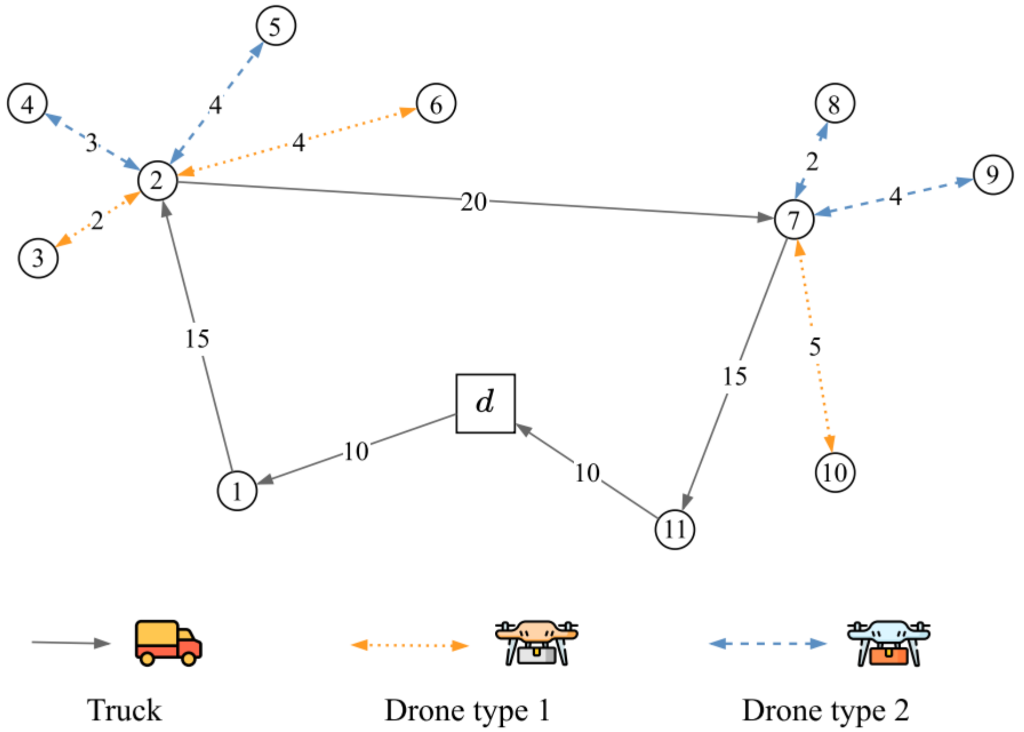

Figure 1:Graphic of solved HDTRP as given by paper

Figure 1 shows an example solution to the HDTRP for a particular instance. The truck leaves and returns to the depot depicted by “d” and the drones make deliveries from nodes 2 and 7.

2.0 Paper's approach to solving the HDTRP

The paper showcased two main algorithms to solve the problem. These were a relatively standard MIP formulation and a LBBD method starting from the original MIP formulation.

2.1 MIP

2.1.1 MIP Formulation

We implemented the MIP as described by the paper with the MIP formulation being as follows.

$x_{ij} \in \{0,1\}$: 1 if truck moves from i to j, 0 otherwise
$h_{ij}^{l} \in \{0,1\}$: 1 if drone $l \in L$ is dispatched from i to j, 0 otherwise
$V_i \in \mathbb{R}^{+}$: visiting order of the truck at i (used for sub-tour elimination constraint)
$w_i \in \mathbb{R}^{+}$: truck waiting time at i

$$(\mathrm{P}) \min \sum_{(i,j)\in A} t_{ij}^{v} x_{ij} + \sum_{i \in N_s} w_i \quad (1)$$

The objective (1) minimises the sum of truck travel times and waiting times which are generated from drone delivery times.

$$\text{s.t.} \sum_{j\in N} x_{sj} = 1, \quad (2)$$

$$\sum_{i\in N} x_{it} = 1, \quad (3)$$

$$\sum_{j\in N_t: j\neq i} x_{ij} = \sum_{j\in N_s: j\neq i} x_{ji}, \quad \forall\, i \in N, \quad (4)$$

$$v_i - v_j \le M(1 - x_{ij}) - 1, \quad \forall\, (i,j) \in A, \quad (5)$$

$$\sum_{i\in N_s: i\neq j} x_{ij} + \sum_{i\in N_s: i\neq j} \sum_{l\in L} h_{ij}^{l} = 1, \quad \forall\, j \in N, \quad (6)$$

$$M \sum_{j\in N_t: j\neq i} x_{ij} \ge \sum_{j\in N: j\neq i} \sum_{l\in L} h_{ij}^{l}, \quad \forall\, i \in N_s, \quad (7)$$

$$\sum_{i\in N_s} \sum_{j\in N: j\neq i} b_{ij}^{l} h_{ij}^{l} \le B^{l}, \quad \forall\, l \in L, \quad (8)$$

$$w_i \ge \sum_{j\in N: j\neq i} \tau_{ij}^{l} h_{ij}^{l}, \quad \forall\, i \in N_s, l \in L, \quad (9)$$

$$v_s = 0, \quad (10)$$

The constraints are given by the equations above. Constraints 2 and 3 ensure the truck departs from and arrives back at the depot. Constraint 4 is the flow balance constraint, that is the truck must leave a node if it visits that node. Constraint 5 is a subtour elimination constraint. A subtour consists of a directed graph disconnected from the truck path connecting the source and sink node. Constraint 6 ensures that each node is served by either a truck or a drone. Constraint 7 ensures that drones can only be dispatched from a node if that node is visited by a truck. Constraint 8 makes sure that drones do not exceed their maximum battery capacity. Waiting times for the truck are generated by constraint 9.

2.1.2 MIP Implementation

The implementation of the MIP was relatively easy. We were able to implement the model using Gurobi and obtain some reasonable results from the model within a short amount of time. However, we spotted some discrepancies between the objective values given in the paper and the objective values from our implementation of the model. This issue will be discussed in detail in section 2.3.

2.2 LBBD

2.2.1 LBBD Formulation

Next, the paper proposed a Logic Based Benders Decomposition (LBBD) approach to the problem. The Benders Master Problem (BMP) is formulated to find a feasible truck route, and the Benders Sub Problem (BSP) finds a drone delivery route that minimises the total waiting time (Fig 2.2.1).

$$
\begin{aligned}
\text{(BMP) } \min \; & \sum_{(i,j)\in A} t^v_{ij} x_{ij} + W && (16)\\
\text{s.t. } & \sum_{j\in N} x_{sj} = 1, && (17)\\
& \sum_{i\in N} x_{it} = 1, && (18)\\
& \sum_{j\in N_t: j\neq i} x_{ij} = \sum_{j\in N_s: j\neq i} x_{ji}, \quad \forall\, i \in N, && (19)\\
& \sum_{j\in N_t: j\neq i} x_{ij} = z_i, \quad \forall\, i \in N, && (20)\\
& x_{ij} \in \{0,1\}, \quad \forall\, (i,j) \in A, && (21)\\
& z_i \in \{0,1\}, \quad \forall\, i \in N, && (22)\\
& W \geq 0. && (23)
\end{aligned}
$$

$$
\begin{aligned}
\text{(BSP) } \min \; & \sum_{i\in N_s} w_i && (32)\\
\text{s.t. } & \sum_{i\in N_s} \sum_{j\in N: j\neq i} b^l_{ij} h^l_{ij} \leq B^l, \quad \forall\, l \in L, && (33)\\
& w_i \geq \sum_{j\in N: j\neq i} \tau^l_{ij} h^l_{ij}, \quad \forall\, i \in N_s, l \in L, && (34)\\
& \sum_{i\in N_s: i\neq j} \sum_{l\in L} h^l_{ij} \geq 1 - z^*_j, \quad \forall\, j \in N, && (35)\\
& \sum_{j\in N: j\neq i} \sum_{l\in L} h^l_{ij} \leq |N| z^*_i, \quad \forall\, i \in N, && (36)\\
& h^l_{ij} \in \{0,1\}, \quad \forall\, (i,j) \in A, l \in L, && (37)\\
& w_i \geq 0, \quad i \in N_s. && (38)
\end{aligned}
$$

Figure 2: (Left) The mathematical formulation of BMP; (Right) The mathematical formulation of BSP

Also, GCS constraints (31) are used instead of the original MTZ subtour elimination constraint (5). This is because the original constraint has a weak linear relaxation bound compared with the GCS constraints. At each node in the BnB tree, an algorithm (*Appendix 4.0:* Algorithm 1) is used to find strongly connected components in the truck route and apply GCS constraints on them. This process is performed in a Callback before optimising the BSP.

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq \sum_{(i,j)\in\delta^+(\{k\})} x_{ij}, \qquad \forall\, k \in S, S \subseteq N_s, |S| \geq 2, \tag{31}$$

If the BSP is infeasible, a feasibility cut is added to the BMP (39) which cuts out all combinations of nodes visited in BMP.

$$\sum_{i\in N_1(\mathbf{z}^*)} (1 - z_i) + \sum_{i\in N_0(\mathbf{z}^*)} z_i \geq 1. \tag{39}$$

If the BSP produces an optimal solution, an optimality cut is added which adds new nodes to the truck route.

$$W \geq \tilde{W}_{\mathbf{z}^*} - \Omega(\mathbf{z}^*) \sum_{i \in N_0(\mathbf{z}^*)} z_i, \qquad (41)$$

where

$$\Omega(\mathbf{z}^*) = \begin{cases} \max_{q=p^*+1,\ldots,\overline{p}} \left\{ \dfrac{\tilde{W}_{\mathbf{z}^*} - \underline{W}^q}{q - p^*} \right\}, & \text{if } p^* < \overline{p} \\ \tilde{W}_{\mathbf{z}^*} - \underline{W}^{p^*}, & \text{otherwise} \end{cases}$$

and $p^* = \sum_{i \in N} z_i^*$.

In order to accelerate the BnB search, three preprocessing steps were also used:

1. Primal Heuristic: Finds an initial incumbent solution and a lower bound of truck route length. (*Appendix 4.0:* Algorithm 2)
2. Truck route bounding: Using the lower bound from primal heuristic, finds an upper bound of the truck route length. (*Appendix 4.0*: Algorithm 3)
3. Determine the lower bound of the total waiting time for each possible truck route length.

After preprocessing, an initial cut (54) is added to the BMP that restricts the truck route length (Sorry we forgot to include this in the presentation).

$$\underline{p} \leq \sum_{i \in N} \sum_{j \in N_t : j \neq i} x_{ij} \leq \overline{p}, \qquad (54)$$

In addition, a subset of the decision variables and constraints from the BSP is added into the BMP to provide a guide (24 - 29).

$$\sum_{i \in N_s : i \neq j} \sum_{l \in L} h_{ij}^l = 1 - z_j, \quad \forall\, j \in N, \qquad (24)$$

$$z_i \geq h_{ij}^l, \quad \forall\, i \in N, j \in N_t, l \in L, \qquad (25)$$

$$\sum_{i \in N_s, j \in N, j \neq i} b_{ij}^l h_{ij}^l \leq B^l, \quad \forall\, l \in L, \qquad (26)$$

$$W \geq \sum_{i \in N_s} w_i, \qquad (27)$$

$$w_i \geq \sum_{j \in N : j \neq i} \tau_{ij}^l h_{ij}^l, \quad \forall\, i \in N_s, l \in L, \qquad (28)$$

$$h_{ij}^l \geq 0, \quad \forall\, (i,j) \in A, l \in L. \qquad (29)$$

2.2.2 LBBD Implementation

The implementation of the LBBD algorithm proved to be difficult due to its complexity and our inexperience in implementing primal heuristics. The enormous size of the algorithm also makes debugging extremely difficult. However, after roughly one week of patient coding and debugging, the LBBD algorithm was successfully implemented, and the test results are consistent with the result from the MIP model, which shows the LBBD implementation is functionally the same as the MIP implementation.

2.3 Difficulties

After the model was implemented, it was found that the optimal objective value returned by our implementation of the algorithm does not match the objective value given in the paper.

Table 1: Difference between objective values

Test Instance	Our implementation	Results from the paper	Difference
P-n16-k8	191.54	177.20	14.34
P-n19-k2	230.53	215.93	14.60
P-n20-k2	243.77	229.39	14.38
P-n21-k2	255.87	239.36	16.51
P-n22-k2	261.54	244.66	16.88

It is clear that the difference between the objective values is too high to be caused by rounding errors. After a prolonged period of debugging, we determined that our implementation has no issues, thus there is something wrong with the parameters given in the paper.

We first tried to make small changes to the parameters (i.e., changing the drone ratio or the battery capacity). We managed to replicate the result of test “A-n37-k5” by setting the battery capacity parameter from [100, 100] to [100, 130] as shown below (*Figure 3*). However, the objective values for other tests are still different.

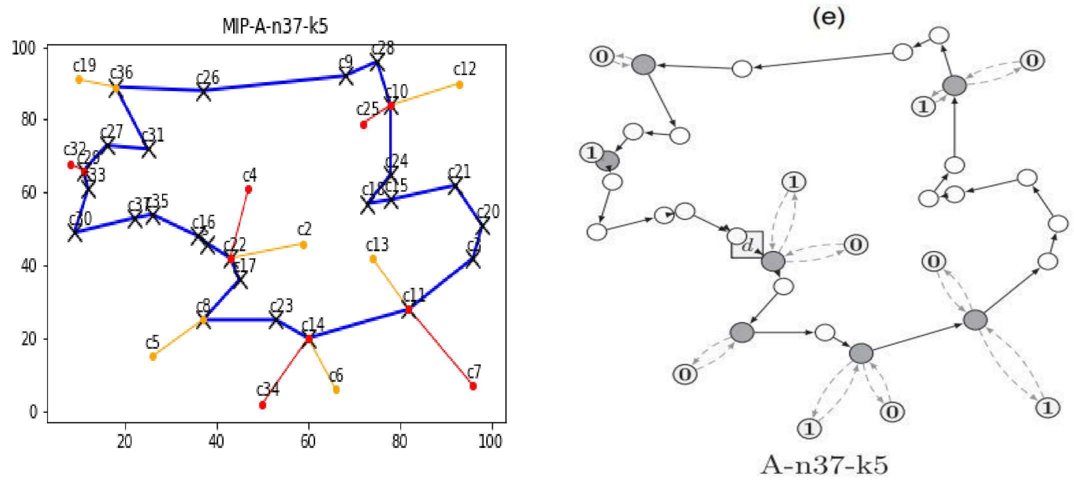

Figure 3: (Left) The result from our MIP implementation; (Right) The sample solution in the paper

By pure luck, we came across a copy of a PowerPoint about this paper, and we noticed that the testing parameters used in the PowerPoint are different.

$$B^l \leftarrow \text{(the maximum duration time)}, \quad \forall\, l \in L, \quad (55)$$

$$\hat{b}_{ij} \leftarrow 2\hat{t}_{ij} + \hat{s}, \quad \forall\, i \in N_s, j \in N, i \neq j. \quad (56)$$

Then, we set

$$b^l_{ij} \leftarrow \alpha_l \hat{b}_{ij}, \quad \forall\, i \in N_s, j \in N, l \in L, i \neq j, \quad (57)$$

$$\tau^l_{ij} \leftarrow 2\hat{t}_{ij}/\alpha_l + \hat{s}, \quad \forall\, i \in N_s, j \in N, l \in L, i \neq j, \quad (58)$$

where α_l is the speed parameter for the drone $l \in L$.

$$t^l_{ij} \leftarrow \alpha^l t^v_{ij}, \qquad \forall (i,j) \in A, \qquad (50)$$

$$s^l_i \leftarrow \beta^l s^v_i, \qquad \forall i \in N, \qquad (51)$$

$$b^l_{ij} \leftarrow 2t^l_{ij} + s^l_j, \qquad \forall (i,j) \in A. \qquad (52)$$

Figure 4: (Left) The parameter stated in the paper; (Right) The parameter on the PowerPoint

We then attempted to rerun some of the tests using the new parameters from the PowerPoint. The resulting objective values are the same as the objective shown on the PowerPoint, which leads us to believe that the testing parameters given in the paper may be flawed or we misunderstood the instructions given in the paper.

Consequently, in the next section, the results we are comparing with came from the PowerPoint instead of the paper.

2.4: Result Comparison

The models were tested using the Augerat and Solomon test sets specified by the paper, with the parameters stated in the PowerPoint (Appendix 1.0). The following four charts (2.4.1 - 2.4.4, *Figure 5*) show the comparison of completion time between MIP and LBBD approaches on different test sets. Note that the y-axis for the first two plots is in log scale.

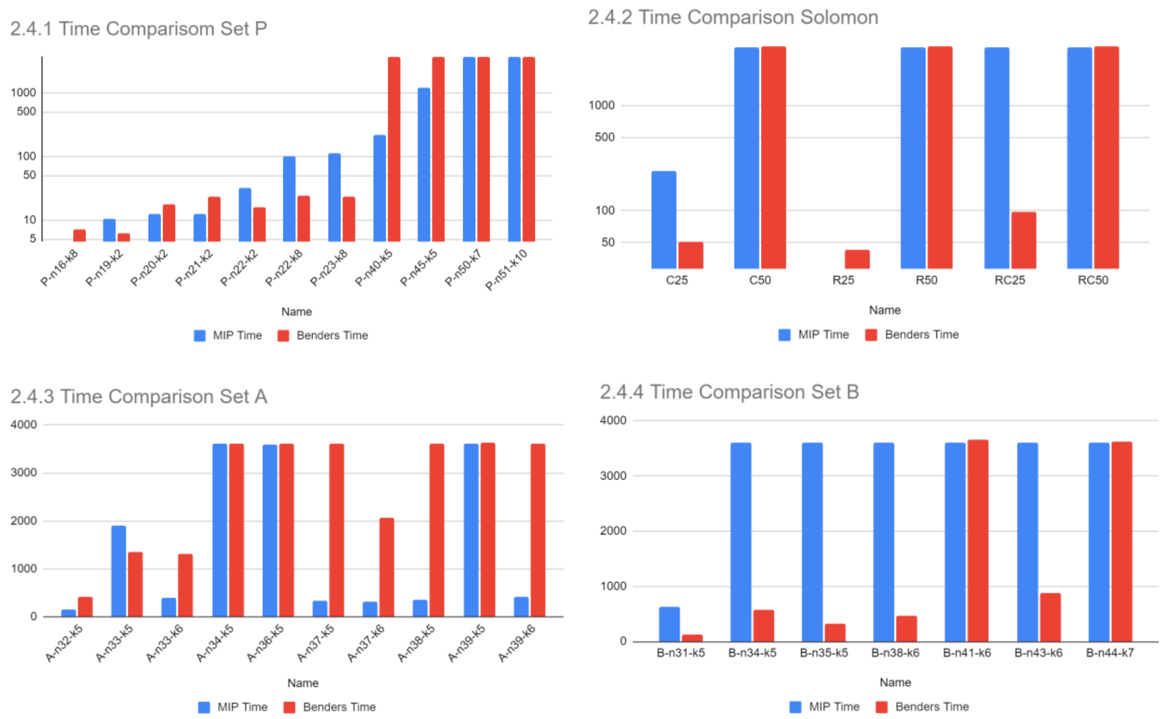

Figure 5: Time comparison between MIP and LBBD

Generally, both MIP and LBBD approaches are able to solve instances (both random and clustered) with less than 30 customer nodes.

It is clear that the LBBD approach is much faster than the MIP when the nodes in the test instance are clustered. For example, the LBBD approach has managed to complete most of the tests in set B, whereas the MIP failed almost all of them. This is likely due to the fact that the incumbent solution generated by the primal heuristics used in LBBD is better. The primal heuristics (*Appendix 4.0*: Algorithm 2) finds an incumbent solution by iterating through all possible truck route lengths (from 0 to |N|), returning the first BSP feasible solution. Which means the incumbent solution has the least number of nodes visited by the truck. If the nodes form several clusters, then the incumbent solution would be to visit each cluster once by truck and use drones to deliver to the rest of the nodes in that cluster, which means the initial gap is much smaller.

On the other hand, the LBBD approach seems to struggle with randomly generated test instances, especially in Set A, in which the MIP implementation can solve many instances within a fraction of the time that it took for the LBBD approach to solve the same instances.

By comparing these results with the results given in the PowerPoint we were able to conclude that some claims in the paper & PowerPoint have been validated, that the LBBD approach excels in instances with clusters of customer nodes. However, the LBBD approach's gap values are not always smaller than the MIP's gap value.

3.0 Improvements attempted

3.1 MIP with Lazy GCS Constraints

3.1.1 MIPLazy Formulation and Implementation

Before successfully implementing the LBBD we decided to test the MIP with GCS constraints added in a Callback instead of the original subtour elimination constraints. This is the same subtour elimination method used in the Benders formulation, but the paper did not test instances using their MIP with lazy GCS constraints.

$$\sum_{(i,j)\in\delta^+(S)} x_{ij} \geq \sum_{(i,j)\in\delta^+(\{k\})} x_{ij}, \qquad \forall\, k \in S, S \subseteq N_s, |S| \geq 2, \tag{31}$$

Each time the MIP is solved to optimality, the solution is checked for subtours which, if they exist, are removed by GCS constraints. These constraints effectively ensure that the number of arcs leaving the subtour is greater than or equal to the number of arcs leaving any node in the subtour which is described mathematically in (31).

GCS constraints have much stronger linear relaxation bounds than the original MIP subtour constraints. However, the reason they are added as lazy constraints is because, without using a Callback, exponentially many constraints would be required to be added before solving.

3.1.2 MIPLazy Result Comparison

The MIP with GCS constraints (MIPLazy) is tested using the same test sets as in section 2.4, and bar charts were made to compare the completion time (3.1.2.1 - 3.1.2.4, *Figure 6*).

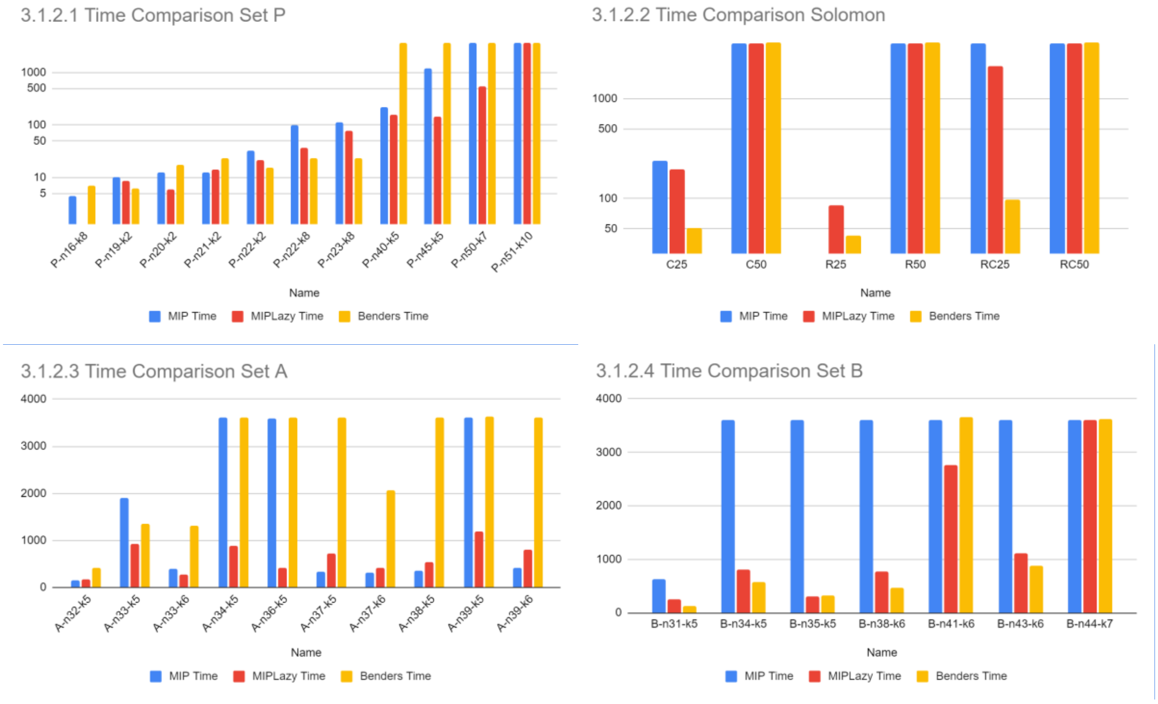

Figure 6: Time comparison between MIP with GCS cuts (MIPLazy) and the papers approaches

We found that the MIPLazy implementation was the most efficient of the three algorithms for most test instances. It was outperformed by the LBBD in certain instances, particularly instances which contained clusters of nodes (Set B and Solomon C and RC). There were also some instances in which the original MIP outperformed the MIP with lazy GCS constraints, however these instances were usually relatively close in solve time compared to many scenarios in which the lazy MIP outperformed the original MIP.

Moreover, implementing the MIP with lazy constraints in Gurobi was much easier than implementing LBBD and was rarely outperformed on test instances, so we believe that this is an improvement.

3.2 Other Improvements Attempted

Another improvement that we attempted was adding disaggregated cuts for the BSP.

$$\sum_{i\in N_s:i\neq j}\sum_{l\in L} h_{ij}^l \geq 1 - z_j^*, \qquad \forall\, j \in N, \qquad (35)$$

$$\sum_{j\in N:j\neq i}\sum_{l\in L} h_{ij}^l \leq |N| z_i^*, \qquad \forall\, i \in N, \qquad (36)$$

This was found to be very difficult as there are multiple constraints in the BSP that sum over the drones (31 and 32). As a result we were not able to implement disaggregated cuts.

3.3 Potential Future Improvements

A potential improvement to our MIPLazy implementation is adding preprocessing to provide an incumbent solution. Some instances solved by the MIPLazy implementation take a long time to find an initial incumbent solution (e.g., Solomon C50) so this could potentially help in niche cases. Preprocessing could follow a similar method to what the paper uses for their LBBD implementation.

4.0 Reflections on the Project

4.1 Holdups in Implementing MIP and LBBD

When we first encountered the objective value discrepancy, we did not think that the parameters in the paper may be wrong. Instead, we spent weeks on finding a bug that does not exist. Which resulted in us not having enough time to test some possible improvements to the MIP with lazy GCS constraints.

Implementing the LBBD approach proved to be a great challenge since we are quite inexperienced in writing complex Gurobi Callback functions. Days of time were spent on reading Gurobi documentations and examples.

4.2 Extendibility and Scale

After completing benchmarking, we have been able to draw a number of conclusions regarding what type of instances the algorithm can feasibly complete. Out of all instances that were attempted, the largest we managed to solve to optimality using the MIP with GCS implementation contained 50 customer nodes (P-n50-k7). However, the smallest that wasn't solved to optimality within 3600 seconds contained only 44 nodes (B-n44-k7). This suggests that the algorithm is suitable for instances of up to about 40 - 50 nodes using similar hardware to ours. If willing to run the algorithm for longer or use results that have not yet been solved to optimality, then the algorithm would likely be able to be used for slightly larger instances. It might be wise to use a heuristic or other type of algorithm if trying to solve instances containing hundreds of nodes.

It is worth noting that changing the drone battery factor can have a significant impact on the time of completion of an instance so it may be possible to solve larger instances by if interested in using different battery parameters.

There are many ways to easily extend the algorithm to solve slightly more complicated problems. The paper suggests multiple such additions. These include allowing for drone battery replacement during the journey, adding weight limits so some nodes would not be able to be serviced by a drone, and adding wind conditions which could impact drone battery usage and delivery times. All these additions can easily be added by constraints to the MIP, MIPLazy or LBBD formulation.

5.0 Conclusion

We were able to implement all techniques and methods described by the paper and use these techniques to solve the same instances as solved by the paper. We ran into initial trouble, having difficulty replicating the same results as the paper, but found a presentation on the same paper that gave an alternate formulation of drone parameters. Using these parameters, we were able to achieve the same objective results as the paper authors with both our MIP implementation and LBBD implementation. We considered that a potential improvement to the paper's method was to use the subtour elimination constraints from their LBBD method with the initial MIP. This MIP with GCS constraints implementation was considered an improvement as it is both much simpler than the LBBD method and more efficient at solving most test instances. There is further work that could possibly improve on this method such as adding preprocessing steps. The implementation could also be extended by adding additional constraints to solve slightly more difficult, and maybe more practical problems.

Appendices

1.0 Parameters from PowerPoint

Table 1 : Computational results for smaller problems with **$|N|$ < 40**. Drone parameters: L={0,1}, (B^0, α^0, β^0)=(100,0.4,0.4) and (B^1, α^1, β^1)=(50,0.2,0.2). *: time limit (3600 seconds) reached

Problem	$\|N\|$	Cplex				Benders				Time ratio	Drone ratio(%)
		Time	BnB	GAP(%)	Obj.	Time	BnB	GAP(%)	Obj.		
C	25	301.2	447 678	0.0	173.71	28.3	2225	0.0	173.71	10.63	84.0
R	25	52.8	74 325	0.0	360.81	17.5	1811	0.0	360.81	3.02	56.0
RC	25	3600.0*	1 745 751	16.2	277.67	37.1	6153	0.0	277.67	**>97.08**	80.0
A-n32-k5	31	430.6	428 257	0.0	575.71	92.3	14 046	0.0	575.71	4.67	51.6
A-n33-k5	32	3600.0*	1 572 781	2.8	549.29	1039.5	83 382	0.0	548.74	**>3.47**	50.0
A-n33-k6	32	3202.8	1 736 877	0.0	552.37	2891.2	64 337	0.0	552.37	1.11	50.0
A-n34-k5	33	3600.0*	2 106 201	5.7	580.98	3600.0*	63 508	1.6	580.57	-	51.5
A-n36-k5	35	3600.0*	1 400 170	1.6	602.55	3600.0*	39 569	1.2	602.55	-	45.7
A-n37-k5	36	873.9	628 443	0.0	633.74	3600.0*	174 774	1.9	633.74	<0.24	44.4
A-n37-k6	36	1149.7	880 493	0.0	635.22	3600.0*	53 406	0.0	635.22	<0.32	47.2
A-n38-k5	37	3119.6	1 762 775	0.0	615.36	3600.0*	394 592	0.4	615.36	<0.87	48.6
A-n39-k5	38	3600.0*	1 589 056	1.4	696.34	3600.0*	37 820	3.7	701.84	-	39.5
A-n39-k6	38	1586.9	823 580	0.0	686.81	3600.0*	65 320	2.3	686.81	<0.44	39.5
B-n31-k5	30	3527.8	1 491 121	0.0	366.19	188.0	11 497	0.0	366.19	18.76	80.0
B-n34-k5	33	3600.0*	2 320 989	16.9	418.52	309.0	18 489	0.0	416.11	**>11.66**	75.8
B-n35-k5	34	3600.0*	1 926 631	19.4	479.80	290.2	36 736	0.0	479.74	**>12.41**	73.5
B-n38-k6	37	3600.0*	2 056 881	16.9	465.87	1040.5	93 853	0.0	465.87	**>3.46**	73.0
B-n39-k5	38	3600.0*	2 334 460	21.0	458.87	3600.0*	52 557	15.5	459.49	-	71.1
P-n16-k8	15	2.5	5875	0.0	150.92	4.1	271	0.0	150.92	0.61	80.0
P-n19-k2	18	12.9	22 383	0.0	187.08	7.8	771	0.0	187.08	1.65	77.8
P-n20-k2	19	16.2	25 097	0.0	200.37	9.7	729	0.0	200.37	1.67	73.7
P-n21-k2	20	44.5	85 038	0.0	208.71	11.8	949	0.0	208.71	3.76	75.0
P-n22-k2	21	69.8	88 403	0.0	213.99	18.2	1809	0.0	213.99	3.83	76.2
P-n22-k8	21	402.5	662 730	0.0	310.94	13.6	1218	0.0	310.94	29.61	61.9
P-n23-k8	22	375.9	470 368	0.0	219.77	22.5	1128	0.0	219.77	16.74	77.3
P-n40-k5	39	1403.8	543 432	0.0	517.96	3600.0*	27 474	3.7	526.58	<0.39	43.6

Figure 7: Computational results and parameters as taken from PowerPoint

2.0 Link to PowerPoint Presentation Associated with Paper

Link to the PowerPoint: kiie.org/wp/2020a/pdf/MT1.pdf

3.0 Benchmarking Results

Table 2: MIP and MIPLazy Benchmarking

Name	MIP Time	MIP objVal	Gap	Node	MIPLazy Time	MIPLazy objVal	# GCS Cuts	Gap	Node
P-n16-k8	4.5723	151.02	0	2564	1.3148	151.02	126	0	2897
P-n19-k2	10.3158	187.08	0	9030	8.8344	187.08	225	0	5757
P-n20-k2	12.641	200.38	0	13018	5.9607	200.38	151	0	7176
P-n21-k2	12.6302	208.74	0	18210	14.4097	208.74	263	0	14047
P-n22-k2	32.2181	214.18	0	43882	21.5496	214.18	390	0	26501
P-n22-k8	99.6664	311.77	0	303350	36.6037	311.77	1251	0	80109
P-n23-k8	112.0777	219.97	0	183630	78.0669	219.97	1176	0	88347
P-n40-k5	219.9142	517.96	0	128414	155.7941	517.96	2612	0	39125
P-n45-k5	1181.091	601.27	0	321330	146.3502	601.27	1506	0	67692
P-n50-k7	3601.625	629.53	0.5	693921	545.5455	628.65	4135	0	242902
P-n51-k10	3601.738	675.24	0.72	877470	3601.08	675.22	11252	0.49	652917
C25	241.0857	173.69	0	195964	196.7246	173.69	959	0	11617
C50	3605.15	449.93	15.33	893443	3624.411	476.86	5686	10.52	6431
R25	28.0817	360.82	0	30265	85.078	360.82	569	0	4417
R50	3602.769	715.99	1.07	1508479	3602.657	717.11	11411	1.56	6502
RC25	3602.514	277.65	12.89	2568708	2146.421	277.65	2124	0	131476
RC50	3606.031	600.1	22.95	883909	3614.242	856.56	3513	37.68	5361
A-n32-k5	153.3306	575.73	0	144595	166.0239	575.73	1076	0	3176
A-n33-k5	1906.815	548.72	0	2812611	920.9443	548.72	950	0	18778
A-n33-k6	405.0379	552.39	0	481711	271.2028	552.39	1513	0	4907
A-n34-k5	3601.834	580.08	2.17	2702373	875.0614	580.08	2214	0	17848
A-n36-k5	3601.243	602.22	0.21	6362441	405.3279	602.22	1137	0	5258
A-n37-k5	341.583	633.76	0	226991	712.0417	633.76	460	0	10585
A-n37-k6	309.5276	635.54	0	234297	412.143	635.54	806	0	5236
A-n38-k5	348.6701	615.37	0	248101	545.3621	615.37	1138	0	5519

A-n39-k5	3606.074	700.54	1.78	3258779	1196.428	696.35	1813	0	9296
A-n39-k6	414.5999	686.82	0	238023	793.8129	686.82	2491	0	7571
B-n31-k5	628.0017	366.2	0	302070	256.5711	366.2	1860	0	6228
B-n34-k5	3606.762	416.1	17.74	2074329	801.5968	416.1	1469	0	19596
B-n35-k5	3607.39	480.32	19.06	1940123	301.0749	479.82	1829	0	3991
B-n38-k6	3604.946	465.91	14.47	1719743	778.9093	465.91	2665	0	7332
B-n41-k6	3604.162	528.99	26.59	1176799	2761.205	526.28	3103	0	19459
B-n43-k6	3602.2771	486.31	2.54	1157021	1111.6623	486.33	1470	0	5457
B-n44-k7	3604.9304	476.14	20.65	1156875	3607.6443	483.74	3115	7.5	12686

Table 3: LBBD Benchmarking

Name	Benders Time	Benders objVal	# GCS Cuts	# Feas Cut	# Opti Cut	Gap	Node
P-n16-k8	7.0619	151.02	113	0	22	0	513
P-n19-k2	6.2596	187.08	127	0	7	0	483
P-n20-k2	17.8738	200.38	384	0	45	0	1175
P-n21-k2	23.1542	208.74	203	0	28	0	791
P-n22-k2	15.654	214.18	164	0	22	0	700
P-n22-k8	23.9532	313.72	524	0	29	0	1682
P-n23-k8	23.0757	219.97	336	0	27	0	971
P-n40-k5	3647.907	525.53	45103	0	489	2.57	15961
P-n45-k5	3603.656	614.76	48059	0	218	3.92	9235
P-n50-k7	3667.093	633.7	31863	1	153	2.55	6731
P-n51-k10	3601.428	685.97	11649	0	166	3.57	7056
C25	51.0463	173.69	916	0	22	0	2430
C50	3668.417	452.33	6331	0	32	3.96	6742
R25	43.0448	360.82	274	0	41	0	1453
R50	3632.32	723.2	22881	2	163	2.9	6567
RC25	96.8248	277.65	491	0	40	0	3124
RC50	3677.941	606.89	5057	0	46	13.77	6257
A-n32-k5	424.676	575.73	689	1	417	0	7373

A-n33-k5	1352.259	548.72	3666	0	1363	0	28023
A-n33-k6	1307.31	552.39	8678	0	1193	0	24938
A-n34-k5	3610.22	580.08	1016	0	4061	0.52	71674
A-n36-k5	3611.374	602.22	2250	3	2919	0.48	70755
A-n37-k5	3621.411	634.74	6333	3	2544	2.18	40346
A-n37-k6	2062.761	635.54	26269	1	629	0.01	22131
A-n38-k5	3604.393	615.37	1607	0	3022	1.12	41079
A-n39-k5	3629.671	702.39	20888	8	2071	3.29	24210
A-n39-k6	3619.693	686.89	20001	0	1452	2.23	26370
B-n31-k5	127.8213	366.2	548	0	33	0	3162
B-n34-k5	578.1163	416.1	3742	1	312	0	8403
B-n35-k5	326.4534	479.8	3738	0	37	0	4536
B-n38-k6	471.5052	465.91	5220	0	10	0	3687
B-n41-k6	3649.403	526.91	5461	0	61	4.22	22033
B-n43-k6	887.5328	486.31	1096	0	80	0	6116
B-n44-k7	3618.9481	476.14	7104	2	67	1.93	15434

4.0 Benders preprocessing algorithms from paper

Algorithm 1 (Separation of GCS for Subtour Elimination)

1: $\mathbf{x}^* \leftarrow$ solution of the (BMP) at the current BnB node
2: $\epsilon \leftarrow 0.8$
3: Construct graph $G(N_{st}, A^*)$, where $A^* := \{(i,j) \in A | x^*_{ij} > 0 \text{ or } x^*_{ji} > 0\}$
4: $\mathcal{S} \leftarrow \{S \subseteq N_s | S$ is a strongly connected component on G$\}$ $\triangleright$ Depth-first search on $G(N_{st}, A^*)$
5: $C \leftarrow \emptyset$
6: **for** $S \in \mathcal{S}$ **do**
7: **for** $k \in S$ **do**
8: $v \leftarrow \sum_{(i,j)\in\delta^+(\{k\})} x^*_{ij} - \sum_{(i,j)\in\delta^+(S)} x^*_{ij}$
9: **if** $v \geq \epsilon$ **then**
10: $C \leftarrow C \cup \{(v, S, k)\}$
11: **end if**
12: **end for**
13: **end for**
14: **return** C

$$
\begin{aligned}
(\text{LW}-p)\ \min\ & \sum_{i\in N_s} w_i && (42)\\
\text{s.t.}\ & \sum_{i\in N_s} \sum_{j\in N: j\neq i} b^l_{ij} h^l_{ij} \leq B^l, \quad \forall\, l \in L, && (43)\\
& w_i \geq \sum_{j\in N: j\neq i} \tau^l_{ij} h^l_{ij}, \quad \forall\, i \in N_s, l \in L, && (44)\\
& y_{ij} = \sum_{l\in L} h^l_{ij}, \quad \forall\,(i,j) \in A, && (45)\\
& z_i \geq y_{ij} \quad \forall\, i \in N_s, j \in N, i \neq j, && (46)\\
& \sum_{i\in N} z_i = p, && (47)\\
& \sum_{i\in N_s: i\neq j} y_{ij} + z_j \geq 1, \quad \forall\, j \in N, && (48)\\
& z_s = 1, && (49)\\
& z_i \in \{0,1\}, \quad \forall\, i \in N_s, && (50)\\
& y_{ij} \in \{0,1\}, \quad \forall\,(i,j) \in A, && (51)\\
& h^l_{ij} \in \{0,1\}, \quad \forall\,(i,j) \in A, l \in L. && (52)
\end{aligned}
$$

LW-p implementation which finds minimum total waiting times for given p (number of nodes).

Algorithm 2 (Primal Heuristic)

```
P ← {1, ..., |N| − 1}
for p ∈ P do
    Solve (LW-p)
    if (LW-p) is feasible then
        W^p ← objective value of (LW-p)
        z^p ← solution of (LW-p)
        Solve TSP with z^p. Let T^p be the objective value.
        Ẑ ← T^p + W^p          ▷ Primal heuristic solution
        p̲ ← p                  ▷ Minimum truck route length
        Break
    end if
end for
return Ẑ, p̲
```

The paper proposes that the above algorithm is used to provide a good incumbent solution, accelerating the branch-and-bound search of the LBBD implementation. The value Z represents the incumbent solution and p represents the lower bound on truck route length.

$$
\begin{aligned}
&(\mathrm{P}{-}p)\ \min\ (1) \\
&\text{s.t. } (2)-(12), \\
&\qquad \sum_{i \in N} \sum_{j \in N_t : j \neq i} x_{ij} = p.
\end{aligned} \tag{53}
$$

Problem P-p: The original MIP problem with truck length restriction.

Algorithm 3 (Truck Route Length Bounding)

Require: $\hat{Z}, \underline{p}$ from Algorithm 2

$p \leftarrow \underline{p} + \text{round}((|N| - \underline{p})/3)$
loop
  Solve the root relaxation of (P-p)
  $\underline{Z}^p \leftarrow$ objective value of the root relaxation of (P-p)
  if $\underline{Z}^p \leq \hat{Z}$ **then**
    Break
  else
    $p \leftarrow \underline{p} + \text{round}((p - \underline{p} + 1)/3)$
  end if
end loop
$\hat{p} \leftarrow p$
for $p \in \{\hat{p} + 1, \hat{p} + 2, \ldots, |N| - 1\}$ **do**
  Solve the root relaxation of (P-p)
  $\underline{Z}^p \leftarrow$ objective value of the root relaxation of (P-p)
  if $\underline{Z}^p > \hat{Z}$ **then**
    $\overline{p} \leftarrow p - 1$ ▷ Maximum truck route length
    Break
  end if
end for
return $\overline{p}$

The above algorithm finds the upper bound on truck route length.