A hybrid genetic search algorithm for the parallel drone scheduling vehicle routing problem

Glenn Feys Veerle Fack

Abstract

In this paper we take a look at drone delivery, and how this can potentially change the way last-mile delivery is done today. We therefore use the parallel drone scheduling vehicle routing problem, which was first introduced by (Murray & Chu, 2015). We discuss our solution of this problem that uses a hybrid genetic search approach that was inspired by the work of (Vidal et al., 2012) and updated it to be able to work with drones and handle our new constraints. We then discuss the use-case of drones in same-day delivery and extend our algorithm to be able to work online in this kind of environment. Finally, we test our solution against the current state-of-the-art, where we were able to make big improvements on large instances. We also critique the current test-/evaluation methods and produce our own. We also provide an open-source implementation of the algorithm on Github.

1. Introduction

In 2013 Amazon came with the announcement of their research on prime air (Amazon), a revolutionary concept of drone delivery. In the last couple of years drones and other unmanned vehicles have seen a significant increase in applications and research. They are currently frequently used in the film-, surveillance-, and maintenance industry to name a few. But drone delivery could well be the next big revolution in the transportation sector and change the way we receive parcels. It can potentially lower costs, reduce delivery times and make the deliver process more sustainable for the environment. Research regarding the technical side of drone delivery is already well-established, but research regarding algorithms for drone delivery has only really started since the paper of Murray and Chu (Murray & Chu, 2015) on the flying sidekick traveling salesman problem (FSTSP). This was the first research done about adding unmanned aerial vehicles to the TSP/VRP. In their paper they have formally defined two variants of the problem. The first is called Flying sidekick TSP (FSTSP), where a truck collaborates with a drone in tandem, and the drone can take-off and land of the truck. This way both the drone and truck deliver packages, but they work together in a tandem. The second variant is called the parallel drone scheduling TSP (PDSTSP), here the truck and drone start from the same depot, but the drone delivers packages independently from the truck. The drone just goes back and forth between customers and the depot to deliver the packages. The ideas have later been generalized with m trucks and n drones into a vehicle routing problem with drones. The problem we will be focusing on in this paper is the VRP variant of the PDSTSP, the parallel drone scheduling vehicle routing problem (PDSVRP). We used a hybrid genetic search (HGS) algorithm inspired by (Vidal et al., 2012) and extended it to be able to handle drones and called it hybrid genetic search with drones (HGSD). Since drones will be used to reduce delivery times and go to a system that can deliver the same day of ordering, we also discuss how we can extend the HGSD to a dynamic, online model where orders can be added, and routes can be driven at any time in the calculations. The focus of the paper is on large scale instances that reflect real-life scenarios. The HGSD algorithm has also proven to be competitive against the current state-of-the-art beating a lot of best-known solutions, especially with a big leap for large instances. We also released an open-source implementation on Github of HGSD and produced a new method to test and evaluate PDSVRP solutions.

2. Parallel drone scheduling vehicle routing problem (PDSVRP)

The vehicle routing problem (VRP) was first introduced by (Dantzig & Ramser, 1959) and was a generalization of the traveling salesman problem (TSP) with n trucks instead of one. The capacitated VRP is defined by a fleet of trucks, a depot where trucks start from and a set of locations the trucks need to visit. Each location must be visited by exactly one truck. Each location has a certain demanded capacity and vehicles also have a maximum capacity. The goal is to construct routes that minimize the total distance traveled by the fleet and does not violate the capacity constraint of any truck. Since the TSP is NP-hard and the VRP is a generalization on this problem, the VRP is also NP-hard. The parallel drone scheduling vehicle routing problem (PDSVRP) is an

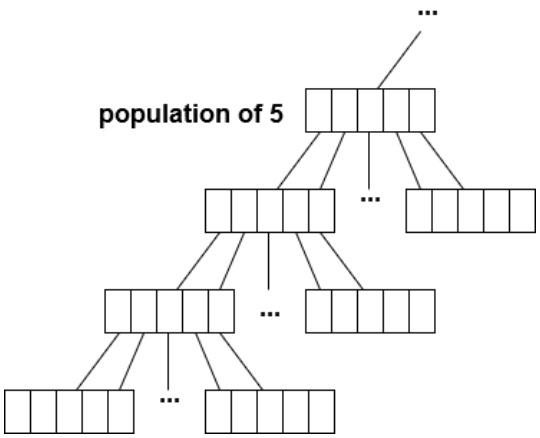

Figure 1. Dynamic chain hirarchy used in population management

extension of the VRP where we include drones in the fleet, therefore it is a heterogenous fleet VRP (HFVRP), but a special case because drones only have a capacity of only one parcel. In PDSVRP the drones start from the depot, deliver the package, and return to the depot for another package. Since a drone always must go back-and-forth to the depot the total distance will always be higher with a drone compared to a car. Therefore, we minimize the makespan in PDSVRP or the completion time from the first vehicle that leaves the depot to the last vehicle that finishes in the depot. Because drones only deliver one package, the order of the deliveries for drones does not matter, and each delivery has a fixed duration, we can transform the planning of the drones into an identical parallel machine scheduling problem (IPS). Now we have two sub-problems the IPS for drones and the VRP for trucks. The IPS has a great greedy solution method, longest-processing-time-first (LPT) (Graham, 1969), that gives close to optimal solutions in linear time. This way we can treat the problem as a relaxation of the VRP where not every location needs to be visited, this is known as the team orienteering problem (TOP). This way we can schedule in the left-over free locations for drones and efficiently get the routes with LPT. So, we can now optimize the makespan of the entire system at once in an efficient manner. A solution consists of a list of routes for trucks and a set of free locations that are scheduled in for drones.

3. Hybrid genetic search with drones (HGSD) for PDSVRP

To solve PDSVRP we use a Hybrid genetic search inspired by (Vidal et al., 2012) this is a genetic algorithm to solve (multi-depot/periodic) capacitated VRP (cVRP). The genetic algorithm uses a combination of mutations and crossovers in an evolutionary manner over a population of solutions. We use advanced population management and diversification management to guarantee a broad search over the solution-space. For the population management we use a chained hierarchy where leaves are random restarts that get combined into a new population. We have a main population with our best solutions where we keep adding solutions from a population of random restarts into each level of the hierarchy, forming a chain as illustrated in figure 1. Each level has a max number of iterations to start the new level. This dynamic population management is also made to easily allow parallelization as we can easily calculate the random restarts on a different thread. Diversification management is used to keep our population diverse; a diversification score is calculated between each pair of solutions in our population, based on the number of predecessors and successors each location has in common between the solutions. The distance is the percentage of different successors and predecessors. We get a diversity contribution of P, ΔP by taking the average of the n lowest diversity scores of all pairs of solutions including P. We then use a biased fitness score BF of a solution, we calculate it as follows for solution P:

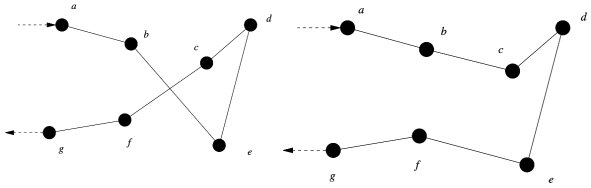

Figure 2. Illustration of 2-OPT

$$BF(P) = (1 - f) * fitness(P) + f * fitness(P) * \Delta P$$

The f is a constant percentage that gives weight to the importance of the diversity in our population.

We used numerous mutations with each an own heuristic to efficiently search over the useful parts of the solution-space. These mutations are efficient, effective, and scalable, so we can do a ton of mutations and see what gives an improvement. What follows is an overview of the mutations used:

- **Add:** Inserts at a random insertion position p of a route, n free locations that are closer than the average distance/location of all routes.
- **Remove:** Removes a sequence of n locations from a random route with a bad distance/location ratio and makes them free locations.
- **Swap-free:** Exchanges a random free location fl with a location l in a route, where l and fl are closer than the average distance/location of all routes.
- **Swap-pos:** Changes the position of a random sequence

of n nodes in a random route to the best positions in the route for that sequence.

- **2-OPT:** Heuristic by (Clarke & Wright, 1964) that removes crossings from a single route. (illustrated in figure 2)
- **cross-over mutation:** Exchanges a random sequence of a random route with a sequence from another route where the start and stop positions of the random sequences are close to each other.
- **Cross:** Extension of 2-OPT that finds different routes that cross and removes the crossing as an inter-route variant of 2-OPT.
- **Cross-over:** Exchanges a random sequence of a random route of the parent solution, into the route of the child solution on the start location of this sequence.

As a fitness function we use the biased fitness function that minimizes the makespan. The problem with only minimizing the makespan is that other advancements in routes different from the longest route are not reflected in the fitness score. Therefore, we made a fitness function that also reflects improvements in the deliveries/distance of the routes. Our fitness function becomes the following:

$$fitness(P) = \sum_{r \in routes} \frac{|r|}{Dist_r + f * makespan}$$

Here $routes$ is the set of all routes of the trucks and drones, $|r|$ is the number of locations in the route and $Dist_r$ the distance of the route, f is a weight to give more importance to the makespan. We plug this into the biased fitness function, and this gives the score we use to rank our population and choose the survivors. For the solution with the highest fitness, we ignore ΔP so it will always be on top of the population.

4. Online HGSD for solving the dynamic PDSVRP

When we want to use drones to facilitate same-day delivery, we will have to make an online variant of HGSD that can manage new incoming orders and remove locations when they are being handled. We use the term dynamic in the sense that the problem can change during calculations. When a route gets driven, it is fixed and will no longer change, therefore a route can be removed from calculations when it is started. A big added complexity is that now a vehicle can drive multiple routes. As a fitness function we no longer use the makespan, as the total time of all operations can no longer be calculated. Since a vehicle can do multiple routes, we can also wait to serve locations until the next route. Therefore, we optimize the deliveries/hour, with the objective to deliver as many parcels as possible in the given time. A problem however is that if we would just optimize deliveries/hour, small very efficient routes would almost always be chosen. Therefore, we add a starting penalty p_{start} so we look to the efficiency over at least p_{start} hours. Since we work with time, we also need truck and drone speed and load and deliver penalties (p_{load}, $p_{deliver}$).

$$time_r = dist_r/v_t + |r| * p_{deliver} + p_{load} + p_{start}$$

$$fitness_{truck} = \sum_{r \in routes} \frac{|r|}{time_r}$$

$$fitness(P) = fitness_{drone} + fitness_{truck}$$

$fitness_{drone}$ is calculated by adding the n drones with shortest delivery duration. The sum of the durations needs to be as long as possible but shorter than the current makespan, then the fitness of drones is given by $fitness_{drone} = \frac{n}{makespan}$. The other locations are for the next shift and not included in the current shift's fitness. Drones deliver the free locations from close to far. Because they come to the depot every time, the route can always change. They always take the closest free location of the current best solution. For a truck we always give the longest route that is not driven yet from the chosen solution.

When making the algorithm online we need to be able to adapt the problem dynamically. We can add locations by adding them in as a free location for all solutions in the current population. The problem this gives is that the fitness might change. Recalculating each fitness is too expensive. Adding a free location can only increase the deliveries/hour for a route, so we can class the location as not scheduled and the fitness does not change. For our best solution we however do recalculate our drone scheduling to get the exact best fitness. When we need to pick a route, we first choose the current best solution and freeze it, then we distribute the routes from this solution amongst the next trucks that need new routes. When we choose a solution, we start new calculations with all free locations from the chosen solution. When we remove a location because a drone needs a new customer, we take the closest free location of the current best solution. The fitness of a solution can however change if we remove a location. As we already mentioned recalculating to too expensive, but what we can do is make an over-approximation. The fitness of drones is calculated by $\frac{|r|}{2*makespan+p_{start}}$ where $|r|$ is the number of locations the drone does in the makespan. So, if we remove a location of a drone the fitness can go down with a maximum of $\frac{1}{2*makespan+p_{start}}$ and for a route it will also approximately go down with this value. So instead of recalculating, every fitness we can just remove the location from every solution and lower the fitness of every solution by $\frac{1}{2*makespan+p_{start}}$.

In a next mutation the fitness will then again be exact. we try to over approximate a little so, that way new mutations can have higher scores which also adds to the diversity.

Another problem is how do we stop? If we always create new routes for m trucks the locations always get split in m, if however, the end of the day is near, and some trucks are driving their last route we want trucks that start a new route to take as many parcels as they still can and not divide the parcels between m. When we start new calculations, we already know the approximated time the trucks will arrive for a new route. If this time is after the time limit, we know they will not need an extra route, so we can start the calculations with less routes. This way we can fully distribute the last locations amongst the last routes, that way we can make a clean stop.

5. PDSVRP test method & results

In literature there are multiple testing methods that are used in PDSVRP, there is not yet consensus about how we evaluate PDSVRP solutions. Firstly, there are multiple variants with different objective functions, the most common ones are the makespan as used by (Murray & Chu, 2015) another common fitness function is minimizing the logistics cost. But even in the case of minimizing the makespan there is no consensus on how we calculate this. To evaluate PDSVRP, firstly, we need to make a difference between trucks and drones, and secondly, we need to determine drone uneligible locations. In many papers the difference between drones and trucks is expressed by letting drones use Euclidean distance and trucks Manhattan distance. Furthermore, drones also have a different speed, sometimes by a factor and sometimes by different speed in km/h but this can also be reduced to a factor. Also currently choosing uneligible routes is done randomly or done by a method that chooses the first 30% based on index and the last 70% by farthest distance from depot. Randomly choosing uneligible locations makes the solutions not objectively reproducible, and choosing farthest locations adds a heuristic in the testing instance which is not ideal. The biggest problem with these testing methods in literature are that Manhattan distance has multiple routes with the same length between two points, therefore when we want to visit a far location, we can visit multiple locations between these locations without increasing the distance by only going in the direction of the far location (illustrated in figure 3). Therefore, a lot of the best makespans are just the Manhattan distance to the farthest uneligible location. This gives ugly solutions; therefore we propose a new objective testing method that uses following steps:

- Choose a cVRP instance (preferably from CVRPLIB)
- Choose drone uneligible locations: For K% drone eligible locations take the $\lfloor n * (K-1) \rfloor$ locations with the highest capacity and lowest index.
- Take the depot of the cVRP problem
- Drones and Trucks both use Euclidean distance.
- Minimize the distance of the longest route (makespan) with a distance-reducing speed factor for drones ($\frac{distance}{speed_{drone}}$).
- All locations are within drone range
- There is no capacity constraint on trucks

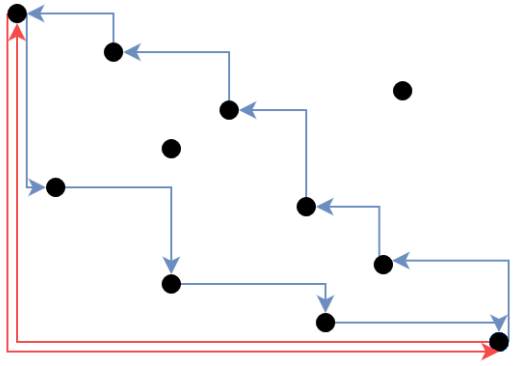

Figure 3. The big problem with Manhattan, blue route has the same Manhattan distance as the red route

Our calculations were done on the UGent HPC (2 x 18-core Intel Xeon Gold 6140 (Skylake @ 2.3 GHz), 88 Gb RAM (4Gb used)) with 1 hour of caluclation time for the instances in tables 1, 3, 4 and 8 hour for the instances in table 2. We tested our solutions against the current best solutions from (Raj et al., 2021) and (Saleu et al., 2022). We simulated the tests with the fitness function and constraints used in the referenced paper. Our results compared to Raj et al. are in table 4 and our results against Salue et al. are in 1. We were able to make improvements for a lot of instances. Especially for bigger instances our improvements are verry high. For instances with 250+ locations there were no solutions yet because most algorithms were not able to get decent results on these instances. We provide our solutions of larger instances 250-1000 nodes in table 3 an example is gives in figure 4 and of very large-scale instances 4.000-11.000 in table 2.

6. Conclusion

Drone delivery is a remarkably interesting and hot topic in research right now and could potentially become a huge revolution in the transport industry. With projects from major companies like Amazon with prime air (Amazon) and Google with Wing (Wing), drone delivery could be a reality sooner than you think. The research for algorithms

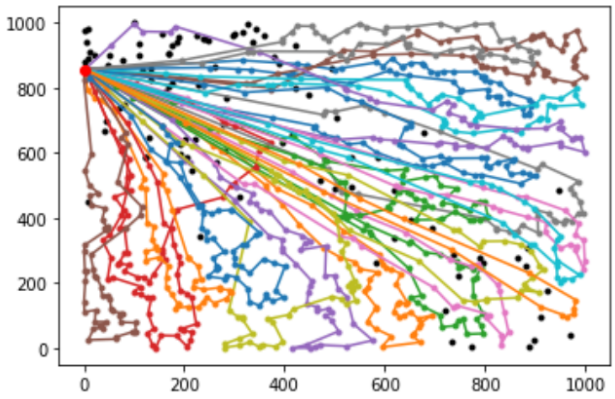

Figure 4. Solution for X-n1001-k43 80% drone eligible with black dots done by drones

to optimize this novel problem are still in an early stage with a lot of new papers in upcoming journals. Currently the solutions are still of varying quality and there are not yet standards or common practices among the researchers. In this paper we researched the problem and adapted one of the best performing cVRP algorithms hybrid genetic search form (Vidal et al., 2012) to be able to efficiently work with drones. The result beat the current state-of-the-art in PDSVRP solutions and even beat some solutions in PDSTSP. We also worked out a way to extend the algorithm to work online for the dynamic problem enabling the algorithm to also work in a same-day delivery environment. We also gave critique on the current way of testing PDSTSP and PDSVRP solutions and produced a new objective testing method for PDSTSP/PDSVRP. PDSVRP is new trendy part of VRP research where there is still a lot of opportunity to do research and potentially work on the next revolution within parcel delivery, to end in the words of Amazon "It would be easy to say, the sky is the limit, but that's not exactly true anymore, is it?".

Table 1. Results PDSmTSP compared against (Saleu et al., 2022)

Instance	n	#D	#V	BKS	HGSD	gap
CMT1	50	3	2	166	166	-0.0 %
CMT2	75	5	5	130.23	128.774	-0.95 %
CMT3	100	4	4	184	184	-0.0 %
CMT4	150	6	6	160.38	154	-4.14 %
CMT5	199	9	8	138	134	-2.99 %
E-n51-k5	50	3	2	168	166	-1.2 %
E-n76-k8	75	4	4	154	152	-1.32 %
E-n101-k8	100	4	4	184	184	-0.0 %
M-n151-k12	150	6	6	154	152	-1.32 %
M-n200-k16	199	8	8	144	138	-4.35 %
P-n51-k10	50	5	5	111.07	110	-0.97 %
P-n55-k7	54	4	3	126	126	-0.0 %
P-n60-k10	59	5	5	114	114	-0.0 %
P-n65-k10	64	5	5	126	124	-1.61 %
P-n70-k10	69	5	5	128	128.253	+0.2 %
P-n76-k5	75	3	2	200	200	-0.0 %
P-n101-k4	100	2	2	342	342	-0.0 %
X-n110-k13	109	7	6	1864	1812	-2.87 %
X-n115-k10	114	5	5	2258	2150	-5.02 %
X-n139-k10	138	5	5	2492	2376	-4.88 %

References

Amazon. Amazon prime air. https://www.amazon.com/Amazon-Prime-Air. Online; accessed 28 January 2022.

Clarke, G. and Wright, J. W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations research*, 12(4):568–581, 1964.

Dantzig, G. B. and Ramser, J. H. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

Graham, R. L. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429, 1969.

Mbiadou Saleu, R. G., Deroussi, L., Feillet, D., Grangeon, N., and Quilliot, A. An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. *Networks*, 72(4):459–474, 2018.

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

Raj, R., Lee, D., Lee, S., Walteros, J., and Murray, C. A branch-and-price approach for the parallel drone scheduling vehicle routing problem. *Available at SSRN 3879710*, 2021.

Saleu, R. G. M., Deroussi, L., Feillet, D., Grangeon, N., and Quilliot, A. The parallel drone scheduling problem with multiple drones and vehicles. *European Journal of Operational Research*, 300(2):571–589, 2022.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., and Rei, W. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, 60(3):611–624, 2012.

Wing, G. How it works. https://wing.com/how-it-works/. Online; accessed 29 November 2021.

Table 2. large-scale PDSVRPD

Instance	n	#D	#V	makespan
Leuven2	4000	35	35	3319.08
Antwerp2	7000	60	60	4466.41
Ghent2	11000	95	95	3386.71

Table 3. Results PDSVRP

Instance	#D	#V	HGSD 80%	HGSD 100%	gap %	HGSD Manh
X-n139-k10	5	5	1844.74	1795.57	-2.66 %	2164
X-n275-k28	14	14	1050.08	1040.16	-0.94 %	1528
X-n439-k37	19	18	1309.90	1290.09	-1.53 %	1856
X-n513-k21	11	10	1757.50	1710.78	-2.73 %	2208
X-n627-k43	22	21	2801.00	2563.95	-9.24 %	3960
X-n716-k35	18	17	1950.77	1786.24	-9.21 %	2718
X-n783-k48	24	24	2487.93	2274.74	-9.37 %	3592
X-n837-k142	71	71	2287.9*	1954.04	-17.08 %	3282
X-n916-k207	104	103	2680.47*	2026.65	-32.26 %	3920
X-n957-k87	44	43	1640.85	1569.79	-4.53 %	2436
X-n979-k58	29	29	2697.49	2600.55	-4.04%	3992
X-n1001-k43	22	21	2542.55	2405.77	-5.36 %	3574

Table 4. PDSTSP & PDSVRP results for TSPLIB instances from Salue et al. (Mbiadou Saleu et al., 2018)

		1			2			3		
Instance	#D	BKS	HGSD	gap %	BKS	HGSD	gap %	BKS	HGSD	gap %
att48	2	28686	**25695.5**	-10.42	17032.0	17196	0.96	14062	**13340**	-5.13
	4	28610	**20276**	-29.12	16500.0	**13811.6**	-16.29	13394	**12990**	-3.02
	6	28610	**20182**	-29.45	16500.0	**13246**	-19.72	13394	**12624**	-5.75
berlin52	2	5290.65	5330	0.74	3328.2	3348.78	0.62	2995.0	**2945**	-1.67
	4	5190.00	5190	0.0	2995.0	**2945**	-1.67	2625.0	2685	2.29
	6	5190.00	**5180**	-0.19	2995.0	**2945**	-1.67	2625.0	2675	1.90
eil101	2	456.00	473	3.73	305.4	**302**	-1.11	235.0	**229**	-2.55
	4	346.68	395	13.94	253.7	**252**	-0.67	200.1	204	1.95
	6	314.0	395	25.79	215.3	224	4.04	181.0	186	2.76
gr120	2	1188.51	1212	1.97	764.0	**754**	-1.31	597.0	**584**	-2.18
	4	946.04	1000	5.70	646.0	**644**	-0.31	528.0	**522**	-1.14
	6	851.4	927	8.88	581.2	**580**	-0.21	527.2	**476**	-9.71
pr152	2	70244	73630	4.82	48967.0	**41566.6**	-15.11	48135	**33590.1**	-30.22
	4	59772	64773.1	8.37	52353.0	**38672.6**	-26.13	43024	**32507.9**	-24.44
	6	56262	62202	10.56	50854.4	**35509**	-30.18	41324	**30807**	-25.45
gr229	2	1673.72	1742.09	4.08	1403.7	**1008.54**	-28.15	1145.4	**698.94**	-38.98
	4	1518.62	**1484.02**	-2.28	1346.2	**916.94**	-31.89	1132.3	**647.9**	-42.78
	6	1467.76	**1362.46**	-7.17	1327.6	**867.48**	-34.66	1130.0	**631.54**	-44.11