2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Optimal Collaborative Transportation for Under-Capacitated Vehicle Routing Problems using Aerial Drone Swarms

Akash Kopparam Sreedhara*, Deepesh Padala*, Shashank Mahesh*, Kai Cui, Mengguang Li, Heinz Koeppl

***Abstract*— Swarms of aerial drones have recently been considered for last-mile deliveries in urban logistics or automated construction. At the same time, collaborative transportation of payloads by multiple drones is another important area of recent research. However, efficient coordination algorithms for collaborative transportation of many payloads by many drones remain to be considered. In this work, we formulate the collaborative transportation of payloads by a swarm of drones as a novel, under-capacitated generalization of vehicle routing problems (VRP), which may also be of separate interest. In contrast to standard VRP and capacitated VRP, we must additionally consider waiting times for payloads lifted cooperatively by multiple drones, and the corresponding coordination. Algorithmically, we provide a solution encoding that avoids deadlocks and formulate an appropriate alternating minimization scheme to solve the problem. On the hardware side, we integrate our algorithms with collision avoidance and drone controllers. The approach and the impact of the system integration are successfully verified empirically, both on a swarm of real nano-quadcopters and for large swarms in simulation. Overall, we provide a framework for collaborative transportation with aerial drone swarms, that uses only as many drones as necessary for the transportation of any single payload.**

I. Introduction

In recent work, drones have been theorized to provide various civilian services. Possible use cases include search and rescue [1], performing inspections in hard-to-reach places [2] or last mile deliveries [3], see also [4] for benefits of using drones to achieve the above. Drone deliveries have become increasingly sought-after in recent works relating to city logistics [4]. This interest has been further accentuated by Google's Project Wing and Amazon's use of quadcopters for their deliveries. A considerable portion of road traffic is caused by package delivery partners, which can be alleviated by using drones to deliver these packages. However, due to the limitations of current battery technology, the thrust-to-weight ratio of drones is relatively low [5]. This limitation requires larger drones, which are very expensive. Instead, this work explores using multiple under-capacitated drones to deliver heavier payloads cooperatively. Using a swarm of (possibly homogeneous) drones working together to achieve a common goal makes transportation faster, easier to scale, and more efficient [6]. However, effectively coordinating many drones remains a challenge that will be addressed in this work. Our goal is to provide an integrated solution for collaborative transportation that optimizes transportation plans, using collision-free path planning for coordination.

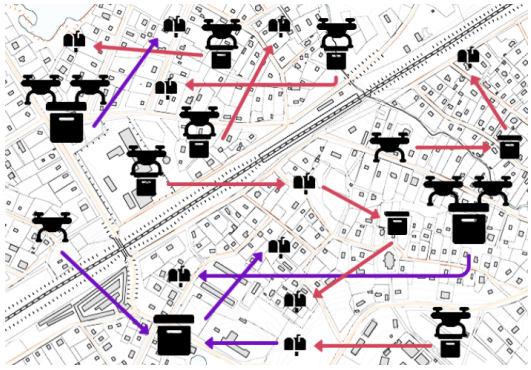

Fig. 1. **Drone swarms in a logistics scenario.** Swarms of drones are sequentially allocated to collaborative transportation missions for many packages. Some packages can be transported by single drones (red), while others require multiple drones to transport collaboratively (purple).

Fig. 1 visualizes our setting in a city logistics scenario. Each drone has a certain capacity and velocity, and each payload has a variable mass. The delivery of all payloads is formulated as a list of missions to be completed. A single drone can complete some missions, while others require cooperation between drones due to a heavier payload. From an economic application point of view, the problem statement requires (i) the minimization of the total time it takes for all drones to perform missions and arrive back at their depot, and (ii) the minimization of total distance travelled, as a proxy for energy or transportation costs. Reducing the time between missions pushes the agents to collaborate, avoiding cases where few agents complete all missions. This results in a complex combinatorial optimization problem, where an improper assignment of drones to missions may lead to underutilization of drones, while an improper ordering of missions may cause drones to travel too far, wait too long for other drones to arrive, or even causes deadlocks.

The above scenario is formulated as a novel multi-objective optimization problem that generalizes the Vehicle Routing Problem (VRP), which in itself generalizes the Multiple Traveling Salesmen Problem (mTSP) and has many applications in fleet management systems or transportation of people [7]. Our under-capacitated VRP (uVRP) aims at optimal routes for a group of drones to complete a set of joint pickup and delivery missions across different locations, i.e. objects may require more than a single drone to be lifted. Qualitatively, uVRP introduces timing constraints as a major difference to standard

*Authors contributed equally.

This work has been funded by the LOEWE initiative (Hesse, Germany) within the emergenCITY center. The authors are with the Departments of Electrical Engineering and Information Technology, Technische Universität Darmstadt, 64287 Darmstadt, Germany. (e-mail: heinz.koeppl@tu-darmstadt.de).

and capacitated VRPs [8], since it is no longer sufficient only to consider the distances travelled between each mission, but instead one must also consider the time spent waiting for other agents to arrive for a collaborative transportation. Our setting generalizes VRP and is similarly not only relevant for logistics, but of independent interest to, e.g., moving services, automated construction [9], humanitarian supply chains [10] or flood rescue operations [11].

Our primary contributions include (i) formulating the first under-capacitated VRP with optimization objectives of interest; (ii) providing an alternating minimization scheme by encoding the solution in a deadlock-free manner, and solving uVRP in practice using metaheuristics; and (iii) demonstrating the integration of collaborative transportation with collision avoidance both in simulation and on a system of real nano quadcopters with payloads in a controlled environment.

II. Related Work

A concept of aerial highways with shared air spaces was introduced in [12]. However, using single drones for delivering payloads is limiting due to the low thrust-payload ratio of drones, which calls for the swarming of drones to enhance efficiency. Previous works showed how swarms of drones could be used to set up communication networks in areas where communications architecture got damaged due to disasters [13], and how they can help in fire suppression activities [14] or surveillance and weather monitoring [15]. The idea of collaboration between decentralized micro-UAVs to transport fabric was introduced in [16]. Load sharing and energy distribution between 2 UAVs to transport objects was explored in [17]. Further, [18] proposed control systems for collaborative transport of rigidly attached payloads using multiple UAVs. It also facilitates collision avoidance for stationary obstacles by modelling them as constraints. Control for collaborative transportation still remains subject of active research [19]. See also, e.g., [20] for a recent overview. In this work, we enable dynamic collision avoidance for moving obstacles, i.e. other groups of drones.

For coordination, e.g., data ferrying using swarms of drones was proposed in [21] by formulating the problem as an mTSP, a relaxation of the VRP for drone delivery [22]. The proposed mTSP was solved using a genetic algorithm. Variants of TSP are a recurring theme in UAV motion planning [23]. Due to their NP-hardness, various metaheuristic algorithms have been employed to solve mTSP and its variants [24]–[26], while [27] employ graph neural networks and reinforcement learning to solve mTSP variants. In the context of drone deliveries, the problems are a form of asymmetric mTSP, where each location represents both picking up and delivering a package at once. In a capacitated setting, [28] further generalizes VRP by introducing drones with the capability to carry multiple packages. A comprehensive overview of the extensive prior literature is found in [29]. In contrast to existing work, we introduce under-capacitated VRP, where the VRP is generalized to allow for drones with payload capacities that are lower than the mass of some payloads, requiring additional coordination in-between drones.

III. Under-Capacitated Vehicle Routing Problem

As discussed in the introduction, the scenario includes N drones $\mathcal{D} \coloneqq \{1, \dots, N\}$ and M transportation missions $\mathcal{M} \coloneqq \{N+1, \dots, N+M\}$. For a compact Euclidean area of operations $\mathcal{X} \subseteq \mathbb{R}^2$, each drone $d \in \mathcal{D}$ has its depot at location $\mathbf{x}_d \in \mathcal{X}$, while each mission $m \in \mathcal{M}$ consists of an object with weight $w_m \in \mathbb{R}$ at $\mathbf{x}_m \in \mathcal{X}$, which is to be transported to destination $\mathbf{y}_m \in \mathcal{X}$. For simplicity, assume homogeneous drones with payload capacity $C \in \mathbb{R}$ and that each drone can transport at most a single object. Extending to heterogeneous drones or higher capacities is relatively straightforward, but omitted for simplicity of exposition. Then, the number of drones required for mission m is $c_m \coloneqq \lceil \frac{w_m}{C} \rceil$.

In traditional VRPs and mTSPs, the standard MinSum and MinMax objectives correspond to minimizing distance and time taken, which can be interpreted as energy and opportunity costs respectively. However, in under-capacitated VRP there is a pronounced difference. In contrast to simply optimizing either the sum or maximum of travelled distances for each drone, the objectives corresponding to energy and time costs do not sum or maximize the same terms. Instead, while the MinSum formulation still minimizes the sum of travelled distances, the MinMax formulation should instead minimize the maximum time taken over all drones for all missions to complete, which includes not only travelling time, but also time spent waiting until all other required drones arrive at a collaborative transportation mission.

a) Travelled distance optimization: To formulate the under-capacitated VRP as an optimization problem, define the set of all drone assignments $\mathcal{G} \coloneqq \mathcal{D} \cup \mathcal{M}$. Then, let $\mathbf{t} \coloneqq (t_{ij}^d)_{d \in \mathcal{D}, i,j \in \mathcal{G}}$ be a tuple of binary variables t_{ij}^d, equal one whenever drone d travels from assignment i to j as part of the solution $\mathbf{t}$. We then define distances between depots and missions $i, j \in \mathcal{G}$ as

$$d_{ij} \coloneqq \begin{cases} \|\mathbf{x}_i - \mathbf{x}_j\| & \text{if} \quad i \in \mathcal{D}, \\ \|\mathbf{x}_i - \mathbf{y}_i\| + \|\mathbf{y}_i - \mathbf{x}_j\| & \text{else.} \end{cases} \tag{1}$$

Then, considering for now only the distances travelled by each drone, a possible formulation of the problem is

$$\max_{\mathbf{t},\mathbf{u}} \quad \sum_{d \in \mathcal{D}} \sum_{i,j \in \mathcal{G}} d_{ij} t_{ij}^d =: J_d \tag{2a}$$

$$\text{s.t.} \quad \sum_{d \in \mathcal{D}} \sum_{j \in \mathcal{G}} t_{ij}^d = c_i \quad \forall i \in \mathcal{M}, \tag{2b}$$

$$\sum_{j \in \mathcal{G}} t_{ij}^d = \sum_{j \in \mathcal{G}} t_{ji}^d \leq 1 \quad \forall d \in \mathcal{D}, i \in \mathcal{M}, \tag{2c}$$

$$\sum_{j \in \mathcal{G}} t_{dj}^d = \sum_{j \in \mathcal{G}} t_{jd}^d = 1 \quad \forall d \in \mathcal{D}, \tag{2d}$$

$$u_i - u_j \geq t_{ij}^d - 1 \quad \forall d \in \mathcal{D}, i \in \mathcal{M}, j \in \mathcal{G}, \tag{2e}$$

$$t_{ij}^d \in \{0,1\}, u_i \in [0,1] \quad \forall i, j \in \mathcal{G}, d \in \mathcal{D}. \tag{2f}$$

We note that the above objective is exactly equal to the sum of travelled distances by all drones. Here, constraint (2b) ensures that exactly c_i drones perform any mission i, constraint (2c) ensures that each drone entering a mission must also exit the mission (at most once), and constraint (2d)

ensures that each drone starts and ends at its depot. Finally and most importantly, the constraint (2e) introduces node potentials $\mathbf{u} = (u_i)_{i\in\mathcal{G}}$ as a subtour and deadlock elimination constraint. In other words, it ensures that there is a strict partial order of all missions, according to which the missions can be performed without deadlock. For an example of a deadlock, consider two missions requiring two drones each. If two drones serve both missions but in opposite orders, each drone will forever wait for the other drone, resulting in a deadlock. The existence of node potentials $\mathbf{u}$ ensures that missions can be executed in order of increasing u_i.

b) Total time optimization: Based on the above formulation, optimizing instead the total time to finish all missions and return to depots, leads to the objective

$$J_T \coloneqq \max_{d\in\mathcal{D}} \left\{ \sum_{i,j\in\mathcal{G}} \left(\frac{d_{ij}}{v} + \Delta T_i^d \right) t_{ij}^d \right\} \tag{3}$$

assuming for simplicity a constant drone velocity v, where one additionally minimizes waiting times at missions i,

$$\Delta T_i^d = \max_{d'\in\mathcal{D}:\ \sum_j t_{ij}^{d'}=1} T_i^{d'} - T_i^d \tag{4}$$

until all other drones d' required for mission i arrive, where T_i^d is the time it takes for drone d to reach node i, recursively computed by including previous waiting times.

c) Multi-objective optimization: For achieving an economic trade-off, we propose a multi-objective optimization via scalarization [30], i.e. we optimize the weighted sum

$$J = \mu J_d + (1-\mu) J_T \tag{5}$$

under some scalarization parameter $\mu \in [0,1]$ that trades off energy and time costs depending on the application.

In contrast to capacitated VRP, the under-capacitated VRP adds timing constraints. Instead of simply considering the distances travelled by each agent, one must also consider the time spent waiting for other agents to arrive at an object. This changes the problem qualitatively. As a result, existing algorithmic approaches are inapplicable, and in the following we develop specialized algorithmic solutions.

IV. Algorithmic Solution

In this work, the primary algorithmic contribution is in formulating an alternating minimization scheme under a natural encoding of valid solutions for the uVRP, i.e. solutions without deadlocks. Numerically, since the uVRP is NP-hard as a generalization of TSP [29], in this work we employ metaheuristics. Unfortunately, due to the potential for deadlocks, it is not possible to decompose uVRP into independent sub-instances of TSP, in contrast to e.g. mTSP [27], which is a special case of uVRP. Therefore, we design a novel algorithm based on a decomposition of the solution into two parts, in turn avoiding any deadlocks.

TABLE I

Representation of an example solution ((π, Ψ), bold).

Order of Mission Execution (π)	Assignments (Ψ) of Drones i to Missions				
	$i=1$	$i=2$	$i=3$	$\dots$	$i=N$
1st Mission: **4**	**1**	**0**	**1**	$\dots$	**0**
2nd Mission: **2**	**0**	**1**	**0**	$\dots$	**1**
3rd Mission: **7**	**1**	**1**	**0**	$\dots$	**0**
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Mth Mission: **3**	**0**	**0**	**1**	$\dots$	**1**

A. Alternating minimization algorithm

The solutions generated by algorithms will be of the form as shown in Table I. Here, π is the order in which payloads are delivered, and Ψ encodes which drones deliver which payload. Consider permutations $\pi \in \mathrm{Sym}(\mathcal{M})$ of missions $\mathcal{M}$ from the symmetric group $\mathrm{Sym}(\mathcal{M})$ (all bijections $\mathcal{M} \to \mathcal{M}$) of $\mathcal{M}$, which define the mission execution order $\pi(1), \dots, \pi(m)$ and realize potentials in (2) by $u_i = \pi(i)/m$. Then, $\Psi \in \mathcal{P}(\mathcal{D})^{\mathcal{M}}$ defines which drones $\Psi(i)$ perform missions i, where $\mathcal{P}$ is the power set. Permutations π and assignments Ψ are implemented as vectors and $\{0,1\}$-valued matrices seen in Table I. As the objective (5) is a function $J = J(\pi, \Psi)$ of (π, Ψ), any tuple (π, Ψ) is thus a solution to the uVRP.

This decomposition allows us to avoid deadlocks by a strict ordering of missions in which they can always be executed, and establishes an algorithm by alternating minimization (AM) of π and Ψ, i.e. starting with $(\pi^{(0)}, \Psi^{(0)})$, in each iteration $q = 0, 1, \dots$ of the AM algorithm, let

$$\Psi^{(q+1)} \coloneqq \operatorname*{arg\,max}_{\Psi\in\mathcal{P}(\mathcal{D})^{\mathcal{M}}} J(\pi^{(q)}, \Psi), \tag{6}$$

$$\pi^{(q+1)} \coloneqq \operatorname*{arg\,max}_{\pi\in\mathrm{Sym}(\mathcal{M})} J(\pi, \Psi^{(q+1)}). \tag{7}$$

The AM algorithm converges to a coordinate-wise optimal solution by the monotone convergence theorem, as $J \geq 0$.

Proposition 1: The AM algorithm given by (6)-(7) converges in $J(\pi^{(q)}, \Psi^{(q)})$ as $q \to \infty$.

B. Metaheuristics

Since the optimization of each subproblem (6) and (7) remains difficult, the alternating minimization algorithm instantiates two metaheuristic algorithms, ρ_Ψ and ρ_π, for the combinatorial optimization problems (6) and (7). Simulated annealing can be better than Genetic Algorithms for TSP [25], while genetic algorithms are widely used to solve mTSP [24]. Combining both algorithms has been proven effective also in other variants of the mTSP [26]. Hence, in this work, the SAGA algorithm in Algorithm 1 implements the alternating optimization scheme by combining a genetic algorithm (GA) ρ_Ψ and a simulated annealing (SA) algorithm ρ_π. The SA algorithm ρ_π is tasked with finding the optimal assignment of payloads to each mission. The GA ρ_Ψ finds the optimal assignment of drones for each mission. Depending on the requirements and constraints, the optimal fitness/cost function can be chosen by adjusting the trade-off between the total time to complete all missions and the total distance travelled by all drones combined. See also Section VI.

Algorithm 1 SAGA for uVRP

Randomly generate initial π.
Randomly generate an initial population of drone assignments $P = \{\Psi_1, \ldots, \Psi_{P_{\mathrm{GA}}}\}$.
for iterations $n = 1, \ldots, N_{\mathrm{alt}}$ **do**
    Optimize $P = \rho_\Psi(\pi)$ using GA ρ_Ψ; N_G iterations
    Optimize $\pi = \rho_\pi(P)$ using SA ρ_π; N_S iterations
return π and $\arg\max_{\Psi_i \in P} J(\pi, \Psi_i)$ with best cost.

a) Genetic Algorithm: Genetic algorithms work by initialising a random population and scoring it. Multiple iterations of mutation and crossover are performed, followed by selection and reinsertion in the population based on the fitness of each genotype $\Psi_1, \ldots, \Psi_{P_{\mathrm{GA}}}$ (individuals in the population P). This ensures that the population is optimized over its iterations. The encoding for the uVRP solution genotype is as in Table I, assuming homogeneous drones.

For initialization, the population is chosen such that each genotype is a valid solution. Drone assignments for each payload m are obtained by sampling c_m indices from $\{1, \ldots, n\}$, and using those drones for the payload, where we recall that c_m is the number of drones required for mission m of the selected payload, and n is the number of drones. The two-parent uniform crossover [31] is used as depicted by Fig. 2, with a mutation operator that randomly replaces the drones in use with the same number of randomly sampled drones to ensure that the drones can carry the payload. The mutation probability was tuned to get the best outcome. The GA uses elitist reinsertion and roulette wheel selection to ensure optimal population generation.

b) Simulated Annealing: The Simulated Annealing (SA) algorithm has been shown to be effective in combinatorial optimization problems where permutations have to be optimized. It works by simulating the heat treatment process of annealing, which alters the properties of a material to make it more suitable to work on, by heating the material to a certain temperature and cooling it to room temperature. As the material (solution) cools down, it obtains more desirable properties (better cost J). Starting with a very high temperature, the SA simulates cooling with a cooling rate and works by transfiguring the payload order by performing one of the following randomly sampled mutation techniques:

- Swap Mutation: Two payloads are chosen randomly in the mission order and swapped.
- Reverse Mutation: Two indices which are $m/2$ apart are chosen randomly, and the slice between them is reversed.
- Insertion Mutation: A payload is randomly removed from π and inserted into a different index.
- Insert Slice Mutation: A random slice is chosen, removed from π and inserted into a random index.

SA avoids converging at a local optimum by using the Metropolis criterion to select a solution with a certain probability even if it is worse than the current solution. At higher temperatures, the SA explores the solution space more, using the probability of accepting a worse solution (π_p, Ψ_p),

$$M_P = \min\left\{1, \exp\left(-\frac{J(\pi_p, \Psi_p) - J(\pi_c, \Psi_c)}{T}\right)\right\} \quad (8)$$

when π_c and Ψ_c are the current solution, and T is the temperature, which decays by multiplying with $\gamma \in (0, 1)$ at each step. Here, Ψ_p and Ψ_c are selected as the best (three) assignments for π_p and π_c respectively from current population P (and averaging the three costs).

It is important to note that any changes by GA to Ψ in matrix form (in Table I) will not affect π as long as the drones assigned to the payloads can transport them. However, any changes to the permutation in π without corresponding changes to Ψ might render the solution invalid. To prevent this, any transfiguration applied to π by ρ_π is also applied to Ψ, effectively ensuring that changing the payload order does not change which drones will carry it. In other words, mutation operations applied to the payload order are also applied to all corresponding indices on each genome in the GA. The temperature is reset to the original starting temperature for each new alternating iteration, since each iteration of alternating minimization fully minimizes in each step. Table II shows the parameters used in this work.

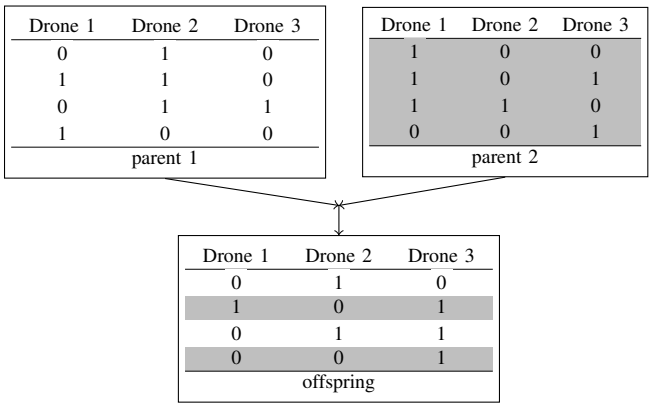

Fig. 2. **Uniform two-parent crossbreeding.** Shown in the figure are 2 parent Ψs from the GA population creating an offspring Ψ. The example problem statement requires the transportation of 4 packages using 3 drones. The drone configuration for each payload is randomly chosen from one of parents 1 and 2, to make it into the offspring.

TABLE II
PARAMETERS OF SAGA

Algorithm Parameters	**Value**
GA iterations (N_G)	200
SA iterations (N_S)	500
Alternating optimizer iterations (N_{alt})	3
Population size (P_{GA})	50
Individuals per parents	3
Selection ratio	0.8
Mutation rate	0.05
Reinsertion ratio	0.7
SA cooling rate (γ)	0.97
SA starting temperature	15000

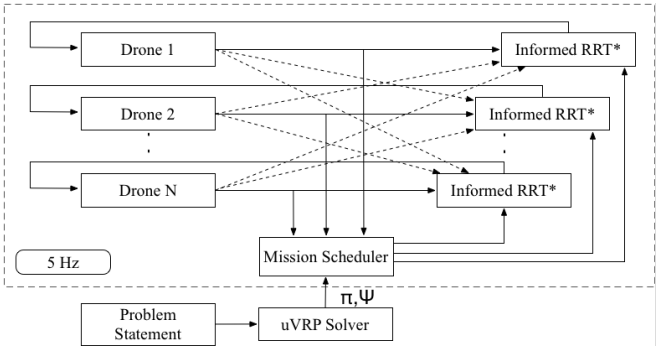

Fig. 3. **Integration of algorithms for swarm delivery.** The uVRP solver generates a near-optimal solution, which is parsed by the mission scheduler to assign drone goals, i.e. pick up, drop off, and transportation. The high-level goals are passed to the Informed RRT* path planner, which plans optimal motion sequences for the drones to execute to complete their missions.

V. Implementation and Methodology

Overall, the uVRP algorithm is integrated into a real system as described in the following, by introducing collision-free path planning and drone controllers, see also Fig. 3.

A. Hardware

This work utilises Crazyflie 2.0 nano-drones equipped with a Lighthouse positioning system (Steam VR Base Station 2.0). Communicating positions and velocities of drones between the central PC (AMD Ryzen 9-5900HX CPU and 16GB RAM) and each drone is done at 5 Hz. A single payload consists of a 3D-printed block with freely moving, cylindrical permanent magnets that are attracted to the iron plate on the payload docker of drones. We attach multiple payloads using carbon fibre rods for heavier and larger, collaboratively transported payloads. The drop-off point employs a stronger magnet that detaches the payload via a stronger magnetic force than the force between the metallic plate and the payload, facilitating successful payload drop-offs. See also Fig. 4 for images.

B. Control and Mission Tracking

The uVRP solver assigns a group of drones to each payload, along with the order in which the payloads must be delivered, as shown in Table I. The drone group may consist of only a single drone, provided that the drone is capable of transporting the payload. When multiple drones are in a group, the radius is increased to a circle, enclosing all drones in the group. A single thread is used to plan a path for the group. A ROS2 node/topic framework is used to integrate all the subsystems. The path planner receives information about the radius, velocity and the current position of each drone group. The high-level commander assigns set points to each drone to make each group follow the path via a cascaded PID controller, which enables position control of each drone.

a) Pick-up mechanism: A pickup manoeuvre cannot be executed unless all drones arrive at the payload. Upon arrival, drones move synchronously to pick up a payload. Upon completion, the path planner is notified to create a group with updated radius, encompassing all involved drones.

b) Drop-off mechanism: The drone group descends synchronously to drop the package. Once done, the group is split into separate groups for each drone, with their corresponding radius and positions being tracked.

c) Mission scheduler: The algorithm to schedule drones iterates over $\pi(1), \ldots, \pi(m)$. The corresponding missions are added to a queue for each drone. The missions to be added to the drones' queues are decided by Ψ. Drones execute a mission when all drones arrive at the payload pickup location. Once payload drop-off is complete, the corresponding mission is removed from the queues of all involved drones. A drone returns to its depot once its mission queue is empty.

C. Path Planning

A decentralised path planning algorithm based on the Informed RRT* [32] planner has been implemented. The sampling-based path planner on each drone [33] facilitates collision avoidance, and was tested in a multi-threaded platform where each thread ran on its corresponding drone. Introducing a latency of up to $3.94\,\mathrm{ms}$ (wireless delays) in the information shared between threads did not affect the performance. Decentralised communication and embedded systems implementations for the Crazyflie 2.1 drones are out of scope. Data shared between drones consists of positions, headings, velocities, and the radius of each drone group.

The SE(2) space was used for the planner to ensure all drones use the same altitude in the airspace to avoid the downwash effect. Extending the algorithm to SE(3) is straightforward and uncomplicated if required. Sampling-based path planners work by sampling a set of states in the SE(2) workspace, checking their validity and choosing a subset of the valid samples (V) for optimal motion planning for the drone. All the sampled states are validated by checking for possible collisions with all active drones. Shadows are projected based on the current velocity of each drone to avoid deadlocks and future collisions. The sampled states which pass the above checks make it into V. Paths are re-planned in case the drones stray too far from the previous path or if the drones' new velocity projections predict collisions.

VI. Experiments and Results

For evaluation in simulation, problem statements are generated by adding drones and payloads (whose weights are sampled uniformly from $\{1, 2, 3\}$) randomly in a workspace of size $4\,\mathrm{m} \times 4\,\mathrm{m}$. The delivery location of each payload is a randomly generated point with a distance X from the starting point such that $X \sim \mathcal{N}(2\,\mathrm{m},\, 4\,\mathrm{m}^2)$.

TABLE III

Comparing SAGA and simulation with path planning on a single problem instance ($\pm$ 95% confidence interval, 50 trials).

n, m	SAGA Predicted Output		Actual output	
	Distance(m)	Time(sec)	Distance(m)	Time(sec)
3, 15	302.0 $\pm$ 4.6	245.7 $\pm$ 5.1	377.0 $\pm$ 0.6	256.2 $\pm$ 0.9
4, 20	419.6 $\pm$ 4.5	295.2 $\pm$ 5.1	509.3 $\pm$ 1.0	336.8 $\pm$ 1.7
5, 25	378.4 $\pm$ 2.6	202.8 $\pm$ 2.8	495.3 $\pm$ 1.1	208.1 $\pm$ 2.3
5, 30	465.0 $\pm$ 4.2	259.3 $\pm$ 5.1	665.5 $\pm$2.8	298.2 $\pm$ 5.2

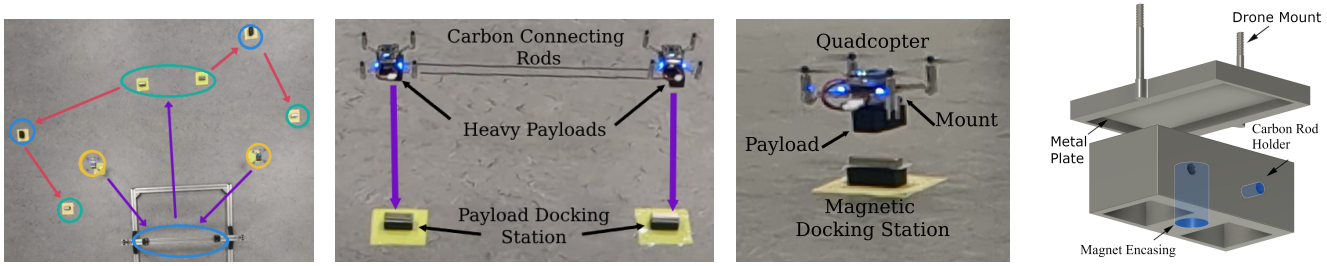

Fig. 4. **Real world system integration.** The integrated system is successfully verified in practice on real nano quadcopters while avoiding collisions between drones. Left: Example demonstration for two drones that first transport an object collaboratively, and then transport two single drone payloads, colors as in Fig. 1; Right: Hardware setup, including the 3D-printed payloads and docking stations.

The trade-off between the total distance travelled and time taken is shown as a Pareto frontier in Fig. 5. The value of μ can be adjusted according to the curve for a given use case to minimize the economic cost (here, we show the results for $n = 4$ drones and $m = 100$ payloads).

Table III displays the difference in the distance and total time predicted by SAGA and uVRP against the integrated system when simulated with path planning, collision avoidance, using a single integrator model in the bullet physics library and the open motion planning library [34]. As seen, the difference is relatively small, i.e. the uVRP model is accurate for practical purposes.

In Table IV, the performance of SAGA was compared to random scheduling, i.e. choosing the best solution from the initial population of SAGA. Since to the best of our knowledge, there exist no algorithms for the novel uVRP formulation, comparisons can only be drawn between SAGA and RA. Other solvers available for VRPs or mTSP cannot be compared due to the uVRP's unique necessity of avoiding deadlocks, which the other solvers do not address. It can be seen that the algorithm successfully optimizes the solution beyond random initializations.

Lastly, an exemplary deployment in the real world is shown in Fig. 4 and the supplementary video[1], where we successfully deploy our results in a real system of nano quadcopters and show the applicability of our developed framework.

[1]https://www.youtube.com/watch?v=SAPVUDS24mA

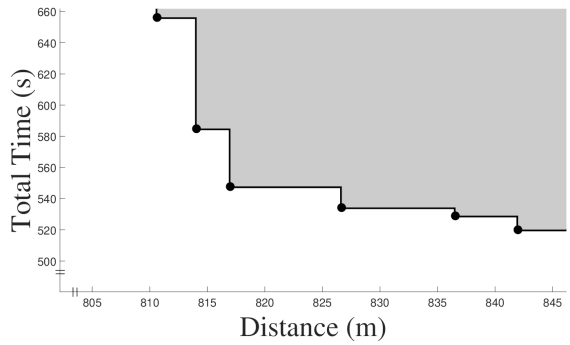

Fig. 5. **Pareto Frontier.** A parameter sweep of μ was performed in the interval $[0.15, 0.9]$ with increments of 0.15. For each value of μ, the algorithm was run 800 times.

VII. Discussion and Conclusion

Overall, we have provided an approach to efficient collaborative transportation via homogeneous drone swarms by formulating a novel extension of VRPs with algorithmic solutions, and successfully providing system integrations for both simulated and real aerial drones. The framework could be expanded upon in future works: The payloads designed for demonstration are known, uniform masses, alleviating the requirement to design specialized control. However, in practice payloads may be unknown. Online system identification could determine package parameters, and coordination algorithms for partial information could be developed. Additionally, when drones collaborate to pick up a package, a single path planning instance is used to plan the path of the entire group. Instead, the state sampling and validation process can be delegated to make motion planning more decentralized and efficient for larger swarms. Finally, the uVRP solver is based on metaheuristics. Future works could explore solving the problems by leveraging attention mechanisms in graph neural networks and reinforcement learning, or by applying linear programming relaxations.

TABLE IV

Comparison of SAGA against Random Assignment (RA): Expected performances with 95% confidence interval ($\pm$) averaged over 100 randomly generated problem instances.

n, m	Alg	J_{time} (sec)	J_{dist} (m)	J ($\mu = 0.2$)
$4, 60$	RA	378.48 $\pm$ 24.14	506.31 $\pm$ 30.37	403.6 $\pm$ 25.4
	SAGA	**310.08** $\pm$ **9.0**	**495.46** $\pm$ **11.28**	**347** $\pm$ **9.5**
$5, 60$	RA	334.98 $\pm$ 25.39	509.19 $\pm$ 31.06	369.8 $\pm$ 26.5
	SAGA	**242.49** $\pm$ **6.59**	**491.08** $\pm$ **10.60**	**292.6** $\pm$ **7.4**
$4, 100$	RA	644.58 $\pm$ 9.21	854.42 $\pm$ 10.27	686 $\pm$ 9.4
	SAGA	**520.03** $\pm$ **9.55**	**828.08** $\pm$ **9.34**	**581.6** $\pm$ **9.5**
$5, 100$	RA	573.60 $\pm$ 9.84	850.42 $\pm$ 13.52	628.4 $\pm$ 10.8
	SAGA	**411.63** $\pm$ **9.047**	**829.00** $\pm$ **14.16**	**494.8** $\pm$ **10.1**
$4, 300$	RA	2044.24 $\pm$ 22.58	2641.69 $\pm$ 27.75	2163.4 $\pm$ 23.6
	SAGA	**1716.57** $\pm$ **29.35**	**2613.04** $\pm$ **32.59**	**1895.4** $\pm$ **29.9**
$8, 300$	RA	1378.08 $\pm$ 17.98	2651.36 $\pm$ 29.711	1632.6 $\pm$ 20.3
	SAGA	**911.13** $\pm$ **18.12**	**2551.94** $\pm$ **30.14**	**1422** $\pm$ **20.524**
$20, 300$	RA	699.15 $\pm$ 11.84	2650.66 $\pm$ 34.11	1089.2 $\pm$ 16.3
	SAGA	**515.20** $\pm$ **14.97**	**2487.90** $\pm$ **39.05**	**909.7** $\pm$ **19.8**

REFERENCES

[1] M. Półka, S. Ptak, and Ł. Kuziora, "The use of UAV's for search and rescue operations," *Procedia Engineering*, vol. 192, pp. 748–752, 2017.

[2] J. Xing, G. Cioffi, J. Hidalgo-Carrió, and D. Scaramuzza, "Autonomous power line inspection with drones via perception-aware MPC," *arXiv preprint arXiv:2304.00959*, 2023.

[3] H. D. Yoo and S. M. Chankov, "Drone-delivery using autonomous mobility: An innovative approach to future last-mile delivery problems," in *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. IEEE, 2018, pp. 1216–1220.

[4] D. Bamburry, "Drones: Designed for product delivery," *Design Management Review*, vol. 26, no. 1, pp. 40–48, 2015.

[5] J. Janszen, B. Shahzaad, B. Alkouz, and A. Bouguettaya, "Constraint-aware trajectory for drone delivery services," in *International Conference on Service-Oriented Computing*. Springer, 2021, pp. 306–310.

[6] S. H. Alsamhi, O. Ma, M. S. Ansari, and F. A. Almalki, "Survey on collaborative smart drones and internet of things for improving smartness of smart cities," *IEEE Access*, vol. 7, pp. 128 125–128 152, 2019.

[7] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013.

[8] T. K. Ralphs, L. Kopman, W. R. Pulleyblank, and L. E. Trotter, "On the capacitated vehicle routing problem," *Mathematical programming*, vol. 94, pp. 343–359, 2003.

[9] F. Augugliaro, S. Lupashin, M. Hamer, C. Male, M. Hehn, M. W. Mueller, J. S. Willmann, F. Gramazio, M. Kohler, and R. D'Andrea, "The flight assembled architecture installation: Cooperative construction with flying machines," *IEEE Control Systems Magazine*, vol. 34, no. 4, pp. 46–64, 2014.

[10] N. A. Kyriakakis, I. Sevastopoulos, M. Marinaki, and Y. Marinakis, "A hybrid tabu search–variable neighborhood descent algorithm for the cumulative capacitated vehicle routing problem with time windows in humanitarian applications," *Computers & Industrial Engineering*, vol. 164, p. 107868, 2022.

[11] F. Dubois, P. Renaud-Goud, and P. Stolf, "Capacitated vehicle routing problem under deadlines: An application to flooding crisis," *IEEE Access*, vol. 10, pp. 45 629–45 642, 2022.

[12] B. Alkouz, B. Shahzaad, and A. Bouguettaya, "Service-based drone delivery," in *2021 IEEE 7th International Conference on Collaboration and Internet Computing (CIC)*. IEEE, 2021, pp. 68–76.

[13] K. Cui, L. Baumgärtner, M. B. Yilmaz, M. Li, C. Fabian, B. Becker, L. Xiang, M. Bauer, and H. Koeppl, "UAV swarms for joint data ferrying and dynamic cell coverage via optimal transport descent and quadratic assignment," in *2023 IEEE 48th Conference on Local Computer Networks (LCN)*. IEEE, 2023, pp. 1–8.

[14] E. Ausonio, P. Bagnerini, and M. Ghio, "Drone swarms in fire suppression activities: A conceptual framework," *Drones*, vol. 5, no. 1, p. 17, 2021.

[15] H. Hildmann, E. Kovacs, F. Saffre, and A. Isakovic, "Nature-inspired drone swarming for real-time aerial data-collection under dynamic operational constraints," *Drones*, vol. 3, no. 3, p. 71, 2019.

[16] R. Cotsakis, D. St-Onge, and G. Beltrame, "Decentralized collaborative transport of fabrics using micro-UAVs," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 7734–7740.

[17] A. Mohiuddin, Y. Zweiri, R. Almadhoun, T. Taha, and D. Gan, "Energy distribution in Dual-UAV collaborative transportation through load sharing," *Journal of Mechanisms and Robotics*, pp. 1–14, 2020.

[18] A. Hegde and D. Ghose, "Multi-UAV collaborative transportation of payloads with obstacle avoidance," *IEEE Control Systems Letters*, vol. 6, pp. 926–931, 2021.

[19] Y. Ping, M. Wang, J. Qi, C. Wu, and J. Guo, "Collaborative control based on payload-leading for the multi-quadrotor transportation systems," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5304–5309.

[20] F. Schiano, P. M. Kornatowski, L. Cencetti, and D. Floreano, "Reconfigurable drone system for transportation of parcels with variable mass and size," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 150–12 157, 2022.

[21] M. Harounabadi, M. Bocksberger, and A. Mitschele-Thiel, "Evolutionary path planning for multiple UAVs in message ferry networks applying genetic algorithm," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*. IEEE, 2018, pp. 1–7.

[22] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85, 2016.

[23] J. Chen, F. Ye, and Y. Li, "Travelling salesman problem for UAV path planning with two parallel optimization algorithms," in *2017 Progress in Electromagnetics Research Symposium - Fall (PIERS - FALL)*. IEEE, 2017, pp. 832–837.

[24] S. Mahmoudinazlou and C. Kwon, "A hybrid genetic algorithm for the min-max multiple traveling salesman problem," *arXiv preprint arXiv:2307.07120*, 2023.

[25] A. Karevan, M. Homayouni, A. Hesami, and S. Larki, "The comparison between simulated annealing algorithm and genetic algorithm in order to solve traveling salesman problem in Esfahan telecommunications companies," in *Proc., 3rd Int. Conf. on Industrial Engineering and Sustainable Management. Pisa, Italy: Sabanet Company*, 2016.

[26] J. Li, M. Zhou, Q. Sun, X. Dai, and X. Yu, "Colored traveling salesman problem," *IEEE Transactions on Cybernetics*, vol. 45, no. 11, pp. 2390–2401, 2014.

[27] Y. Hu, Y. Yao, and W. S. Lee, "A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs," *Knowledge-Based Systems*, vol. 204, p. 106244, 2020.

[28] M. A. Masmoudi, S. Mancini, R. Baldacci, and Y.-H. Kuo, "Vehicle routing problems with drones equipped with multi-package payload compartments," *Transportation Research Part E: Logistics and Transportation Review*, vol. 164, p. 102757, 2022.

[29] O. Cheikhrouhou and I. Khoufi, "A comprehensive survey on the multiple traveling salesman problem: Applications, approaches and taxonomy," *Computer Science Review*, vol. 40, p. 100369, 2021.

[30] M. Ehrgott, "A discussion of scalarization techniques for multiple objective integer programming," *Annals of Operations Research*, vol. 147, no. 1, pp. 343–360, 2006.

[31] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia tools and applications*, vol. 80, pp. 8091–8126, 2021.

[32] M.-C. Kim and J.-B. Song, "Informed RRT* with improved converging rate by adopting wrapping procedure," *Intelligent Service Robotics*, vol. 11, pp. 53–60, 2018.

[33] Y. Lin and S. Saripalli, "Sampling-based path planning for UAV collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3179–3192, 2017.

[34] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, https://ompl.kavrakilab.org.