Flight Structure Optimization of Modular Reconfigurable UAVs

Yao Su[1], Ziyuan Jiao[1], Zeyu Zhang[1], Jingwen Zhang[1], Hang Li[1], Meng Wang[1], Hangxin Liu[1†]

***Abstract*— This paper presents a Genetic Algorithm (GA) designed to reconfigure a large group of modular Unmanned Aerial Vehicles (UAVs), each with different weights and inertia parameters, into an over-actuated flight structure with improved dynamic properties. Previous research efforts either utilized expert knowledge to design flight structures for a specific task or relied on enumeration-based algorithms that required extensive computation to find an optimal one. However, both approaches encounter challenges in accommodating the heterogeneity among modules. Our GA addresses these challenges by incorporating the complexities of over-actuation and dynamic properties into its formulation. Additionally, we employ a tree representation and a vector representation to describe flight structures, facilitating efficient crossover operations and fitness evaluations within the GA framework, respectively. Using cubic modular quadcopters capable of functioning as omni-directional thrust generators, we validate that the proposed approach can (i) adeptly identify suboptimal configurations ensuring both over-actuation and trajectory tracking accuracy and (ii) significantly reduce computational costs compared to traditional enumeration-based methods.**

I. Introduction

By docking and undocking with each other, modular UAVs can transform into various structures, exhibiting significant advantages in terms of versatility, robustness, and cost compared to aerial robots with fixed flight configurations [1]. An ideal flight structure is typically achieved by human designs leveraging expert knowledge that accommodates task-specific requirements such as payload transportation [2, 3], object manipulation [4–7], and dynamic exploration [8, 9]. To mitigate the manual efforts involved in the design process, several algorithms have been proposed to systematically enumerate all possible flight structures and select the optimal one based on a given metric [10–14]. However, as the number of modular UAVs in the system increases, designing an optimal flight structure solely based on human expertise could become infeasible, and the computational complexity of finding such one grows exponentially.

Search sub-configurations with a heuristic to reduce the search space [12] and optimization-based methods [15, 16] have recently emerged as promising means to improve the computational efficiency in finding optimal or at least near-optimal flight structures of a modular UAV system. However, when the modules are equipped with different types of sensors, interacting tools, or payloads, the complexity of finding an ideal flight structure through algorithmic solutions escalates because of the **non-identical weights and inertia parameters** among the modules. Such heterogeneity would significantly influence the dynamical properties of the resulting vehicle, leading to stability and trajectory-tracking challenges. Furthermore, the solution space of finding optimal flight structures with heterogeneous modules becomes too complex for existing methods [12, 15, 16].

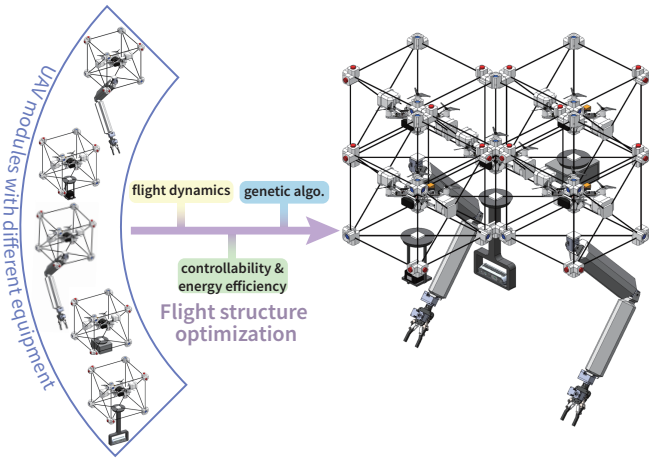


Fig. 1: **The optimal structure configuration with five modular UAVs with different installed equipment.** Each module is equipped with either a manipulator, an RGBD camera, a Lidar, or a computing unit, resulting in different weights and inertia parameters. The proposed algorithm efficiently produces an over-actuated flight structure with optimal dynamical properties.

To tackle the above challenge, we employ Genetic Algorithm (GA) to efficiently find the suboptimal flight structure composed by UAV modules with different weights and inertia to achieve (i) over-actuation with high trust efficiency and (ii) good controllability for trajectory tracking. The GA is a population-based search algorithm capable of addressing both constrained and unconstrained optimization problems through natural selection [17], thus offering a more flexible formulation to solve the structure optimization problem. To support efficient computation with a minimal budget, we leverage two representations to describe the flight structure of a modular UAV system within the GA: the **assembly incidence matrix (AIM)** (tree representation) for generating new structures with specialized crossover operations [18] and the position vectors for evaluate the fitness of each structure.

To evaluate the proposed approach, we adopt a customized modular UAV system. Each module is a quadcopter connected to a cubic docking frame through a 2-Degree-of-freedom (DoF) passive gimbal mechanism and thus can be treated as an **omni-directional thrust generator** after docking with each other horizontally and forming a

This work was supported in part by the National Natural Science Foundation of China (No.62403063, 52305007, 62376031).

† Corresponding author.

[1] State Key Laboratory of General Artificial Intelligence, Beijing Institute for General Artificial Intelligence (BIGAI). Emails: {suyao, jiaoziyuan, zhangzeyu, zhangjingwen, lihang, wangmeng, liuhx}@bigai.ai

flight structure [19, 20]. Specialized equipment with different weights or shapes can also be installed on the bottom of a module's docking frame, resulting in heterogeneous modules. Fig. 1 illustrates the modular UAV system and the flight structure found by the proposed algorithm which is over-actuated, *i.e*. independent position and attitude control, and dynamically optimal.

Our verification first illustrates the convergence process of solving the optimization of flight structures in two cases, each consisting of more than 30 heterogeneous modules. Then, four flight structures at different stages of the optimization process are selected to compare their dynamic properties in trajectory-tracking simulation after implementing a hierarchical controller. Finally, the computational efficiency of the proposed GA is studied and contrasted with that of the enumeration-based method. The results highlight that the proposed approach provides dynamically optimal flight structures for large modular UAV systems with significantly better computational efficiency.

II. Related Work

Developing **aerial modular robots** that can adapt to various mission settings flexibly through reconfiguration has attracted a lot of research attention, and progress has been made in modular UAV system hardware design [2, 3, 5, 21, 22], dynamics modelling [20, 23, 24], formation control [5], control allocation [19, 25, 26], and docking trajectory planning [27, 28]. In the above cases, the system's structure and module configurations are manually designed and fixed. More recent work sought to find the optimal/suboptimal flight structure for modular UAVs. Specifically, Gandhi *et al*. formulated a mixed integer linear programming to reconfigure the flight structure under propeller faults on some modules [15]. Gabrich *et al*. proposed a heuristic-based subgroup search algorithm that achieved better computation efficiency than enumerating different types of modules at each module location [12]. Xu *et al*. evaluated a structure's actuation capability by solving an optimization problem [16]. However, the dynamic discrepancy between different modules in weight and inertia parameters is still not considered by these works, prohibiting the fly structure from achieving superior dynamic properties in different task environments.

The problem of **structure optimization** for a modular robotic system was first investigated by Chen *et al*. in [10], where an algorithm was proposed to enumerate all non-isomorphic configurations of a modular manipulator system. The following work extended the scope to various modular robot systems with improved computation efficiency [10, 11, 13, 14]. However, as the number of modules increases, the set of configurations enlarges factorially, and finding an effective configuration through an exhaustive search becomes infeasible [18, 29]. On the other hand, utilizing **GA** to find a suboptimal configuration of the modular robotic system has been demonstrated in a simple serial connected structure with much better computation efficiency [18]. In this paper, we generalize this approach to a modular UAV system with a more complicated **tree structure**. Taking advantage of our proposed fitness evaluation function and crossover operation for the GA, a suboptimal flight structure configuration can be efficiently found.

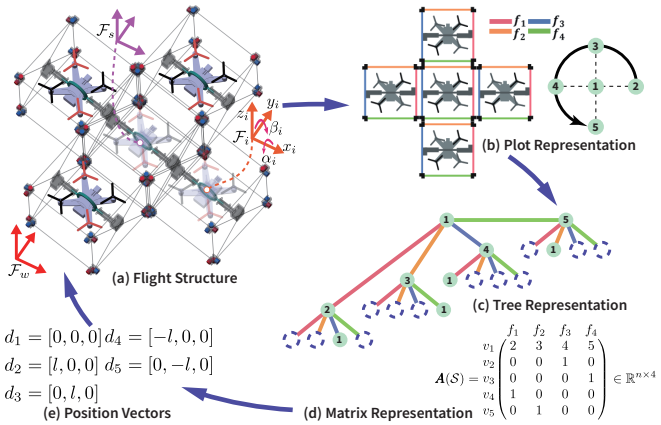


Fig. 2: **Coordinate systems and configuration representations of a flight structure.** Each quadcopter module can connect to four others with its docking faces, and each module serves as an omni-directional thrust generator after connections, making the flight structure over-actuated. Different types of representation of the flight structure are incorporated to support subsequent optimization.

III. System Configuration

Each modular UAV is constructed by connecting a customized quadcopter to the cubic docking frame (size l) by a 2-DoF passive gimbal mechanism that has no limits in rotation angle. As shown in Fig. 2(a), a flight structure can be formed with several modules (indexed by $i = 1, \cdots, n$) by docking with each other. As each quadcopter module can be utilized as an omni-directional thrust generator, the resulting multirotor flight structure has the potential of being over-actuated to gain better maneuverability with independent position and orientation tracking capabilty [30]. To represent the entire flight structure, we first index the four docking faces of one module by $\{f_1, f_2, f_3, f_4\}$ (see Fig. 2(b)). Then we derive a simplified representation of the flight structure using a matrix (tree) for the subsequent computation applied to the rest of this paper.

A. System Frames Definition & Notation

Let $\mathcal{F}_W$ denote the world coordinate frame and $\mathcal{F}_\mathcal{S}$ be the structure frame attached to the geometric center of the flight structure. We define the position of the structure as $\boldsymbol{X}_\mathcal{S} = [x_\mathcal{S}, y_\mathcal{S}, z_\mathcal{S}]^\mathsf{T}$, the attitude of the structure in the roll-pitch-yaw convention as $\boldsymbol{\Theta}_\mathcal{S} = [\phi_\mathcal{S}, \theta_\mathcal{S}, \psi_\mathcal{S}]^\mathsf{T}$, and the angular velocity $\boldsymbol{\Omega}_\mathcal{S} = [p_\mathcal{S}, q_\mathcal{S}, r_\mathcal{S}]^\mathsf{T}$. The frames $\mathcal{F}_i$s are attached to the center of the i-th module. $\boldsymbol{d}_i = [x_i, y_i, 0]^\mathsf{T}$ denotes the vector from $\mathcal{F}_i$ to $\mathcal{F}_\mathcal{S}$.

B. Flight Structure Configuration

The configuration of the flight structure refers to a set of points ($\boldsymbol{d}_i$) in $\mathcal{S}$ that represents the positions of n docked modules w.r.t the structure frame $\mathcal{F}_\mathcal{S}$. Following the idea introduced by Chen *et al*. [18], we describe the flight structure as a tree where the module **1** is chosen as the root. The structure configuration can then be characterized by the assembly incidence matrix (AIM).

Using the configuration in Fig. 2(a) as an example, the corresponding AIM is shown in Fig. 2(d), where each row

represents a module, while each column represents a docking face.

$$A_{ip} = j, A_{jq} = i, p, q \in \{1, 2, 3, 4\}, \tag{1}$$

indicates the p-th face of module i is connected to the q-th face of module j. Of note, $\boldsymbol{A}(\mathcal{S})$ is a normalized representation that is independent of module dimension l.

We can calculate $\boldsymbol{d}_i$ from $\boldsymbol{A}(\mathcal{S})$ with the following steps (refers as **PosTreeSearch** function in the rest of the paper) :

1) assuming the module **1** is at the origin $\boldsymbol{d}_1 = [0, 0, 0]^\mathsf{T}$ and determining the position of each module $\boldsymbol{d}_i$ recursively with the Depth-First tree traversal algorithm and the **Step** matrix that describes the relative position between two modules according to the docked face:

$$\mathbf{Step} = \begin{matrix} x \\ y \\ z \end{matrix} \begin{pmatrix} f_1 & f_2 & f_3 & f_4 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \times l \in \mathbb{R}^{3\times 4}. \tag{2}$$

   Specifically, if $\boldsymbol{d}_i$ is known and Eq. (1) satisfied, then the position of the module j is calculated as:

$$\boldsymbol{d}_j = \boldsymbol{d}_i + \mathbf{Step}(:, p). \tag{3}$$

2) calculating the geometric center position of the structure $\mathcal{S}$ with $\boldsymbol{d}_o = \frac{1}{M}\sum_{i=1}^{n} m_i \boldsymbol{d}_i$, where $M = \sum_{i=1}^{n} m_i$ is the total mass of the flight structure with m_i represents the mass of module i;
3) shifting the geometric center of $\mathcal{S}$ to the origin with $\boldsymbol{d}_i = \boldsymbol{d}_i - \boldsymbol{d}_o$, which ensures $\frac{1}{M}\sum_{i=1}^{n} m_i \boldsymbol{d}_i = 0$.

The reason for having both position vectors $\boldsymbol{d}_i$ and AIM as two representations for the flight structure is twofold: (i) the dynamic property evaluation of each flight structure requires $\boldsymbol{d}_i$, but checking the feasibility of forming a connected structure with $\boldsymbol{d}_i$ of each module can be a tedious process [10]; (ii) describing the flight structure as a tree with the AIM representation naturally ensures all the modules are connected. Additionally, dividing the tree into two sub-trees and reconnecting them is an efficient way to generate new structures. This approach simplifies the feasibility check process as only the overlap between modules needs to be considered. Therefore, during flight structure optimization, we maintain two representations of each structure to efficiently generate new structures and evaluate their dynamic properties.

C. Flight Structure Dynamics

Given a flight structure, its translational dynamics can be described as [31]:

$$M\,\ddot{\boldsymbol{X}}_{\mathcal{S}} = {}^{W}_{S}\boldsymbol{R}\Big(\sum_{i=1}^{n} {}^{S}_{i}\boldsymbol{R}\, T_i \hat{\boldsymbol{z}}\Big) + M\, g \hat{\boldsymbol{z}}, \tag{4}$$

where $\ddot{\boldsymbol{X}}_{\mathcal{S}}$ is the linear acceleration of the flight structure, g is the gravitational acceleration, ${}^{S}_{i}\boldsymbol{R}$ is a function of tilting and twisting angles (α_i and β_i) of ith module, and T_i refers to the magnitude of thrust generated by ith module, $\hat{\boldsymbol{z}} = [0,\ 0,\ 1]^\mathsf{T}$.

Its rotational dynamics can be described as [32]:

$$\boldsymbol{J}_{\mathcal{S}}\,\dot{\boldsymbol{\Omega}}_{\mathcal{S}} = -\boldsymbol{\Omega}_{\mathcal{S}} \times \boldsymbol{J}_{\mathcal{S}}\boldsymbol{\Omega}_{\mathcal{S}} + \sum_{i=1}^{n} (\boldsymbol{d}_i \times {}^{S}_{i}\boldsymbol{R}\, T_i \hat{\boldsymbol{z}}), \tag{5}$$

where $\boldsymbol{J}_{\mathcal{S}}$ is the total inertia matrix of the structure:

$$\boldsymbol{J}_{\mathcal{S}} = \sum_{i=1}^{n} \boldsymbol{J}_i + \sum_{i=1}^{n} m_i \begin{bmatrix} y_i^2 & -x_i y_i & 0 \\ -x_i y_i & x_i^2 & 0 \\ 0 & 0 & x_i^2 + y_i^2 \end{bmatrix}, \tag{6}$$

with $\boldsymbol{J}_i = Diag(J_{ix},\, J_{iy},\, J_{iz})$ is the inertia matrix of each the module, $\dot{\boldsymbol{\Omega}}_{\mathcal{S}}$ is the angular acceleration of the structure.

Taking together, the complete dynamics model of the flight structure is:

$$\begin{bmatrix} \ddot{\boldsymbol{X}}_{\mathcal{S}} \\ \dot{\boldsymbol{\Omega}}_{\mathcal{S}} \end{bmatrix} = \begin{bmatrix} \frac{1}{M}\,{}^{W}_{S}\boldsymbol{R} & 0 \\ 0 & \boldsymbol{J}_{\mathcal{S}}^{-1} \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} g\hat{\boldsymbol{z}} \\ -\boldsymbol{J}_{\mathcal{S}}^{-1}(\boldsymbol{\Omega}_{\mathcal{S}} \times \boldsymbol{J}_{\mathcal{S}}\boldsymbol{\Omega}_{\mathcal{S}}) \end{bmatrix}, \tag{7}$$

where $\boldsymbol{u}$ is the 6-DoF wrench generated by all modules with

$$\begin{aligned} &\boldsymbol{u} = \begin{bmatrix} \boldsymbol{u}_X \\ \boldsymbol{u}_\Omega \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} {}^{S}_{i}\boldsymbol{R}\, T_i \hat{\boldsymbol{z}} \\ \sum_{i=1}^{n} (\boldsymbol{d}_i \times {}^{S}_{i}\boldsymbol{R}\, T_i \hat{\boldsymbol{z}}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_X(\boldsymbol{\alpha}, \boldsymbol{\beta}) \\ \boldsymbol{J}_\Omega(\boldsymbol{\alpha}, \boldsymbol{\beta}) \end{bmatrix} \boldsymbol{T}, \\ &\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_n]^\mathsf{T}, \boldsymbol{\beta} = [\beta_1, \cdots, \beta_n]^\mathsf{T}, \boldsymbol{T} = [T_1, \cdots, T_n]^\mathsf{T}. \end{aligned} \tag{8}$$

Of note, due to the passive gimbal mechanism design and the low-level control of each module eliminating rotating torques in the z-axis [20], there is **no rotation-induced torque** of propellers pass to the mainframe. Therefore, in the simplified dynamics model Eq. (7), the input of each module is only considered as a 3-axis force vector. Besides, the total inertia matrix is approximated as a constant matrix.

D. Force Decomposition-based Analysis

Utilizing the force decomposition method, we can transform the nonlinear relationship in Eq. (8) to a linear one by defining $\boldsymbol{F}$ as an intermediate variable [20, 26]:

$$\boldsymbol{F} = \begin{bmatrix} F_1 \\ \vdots \\ F_n \end{bmatrix} \in \mathbb{R}^{3n\times 1}, F_i = \begin{bmatrix} \sin\beta_i \\ -\sin\alpha_i \cos\beta_i \\ \cos\alpha_i \cos\beta_i \end{bmatrix} T_i, \tag{9}$$

Then, the 6-DoF wrench $\boldsymbol{u}$ can be rewritten as:

$$\boldsymbol{u}^d = \begin{bmatrix} \boldsymbol{J}_X(\boldsymbol{\alpha}, \boldsymbol{\beta}) \\ \boldsymbol{J}_\Omega(\boldsymbol{\alpha}, \boldsymbol{\beta}) \end{bmatrix} \boldsymbol{T} = \boldsymbol{W}\boldsymbol{F}, \tag{10}$$

where $\boldsymbol{W} \in \mathbb{R}^{6\times 3n}$ is a constant allocation matrix with full row rank. Treating $\boldsymbol{F}$ as inputs to the system, its dynamics can be analyzed from a linear perspective. The real inputs tilting angle α_i, twisting angle β_i, and thrust force T_i of each module are computed from $\boldsymbol{F}$ using inverse kinematics.

IV. Flight Structure Optimization

To find the optimal flight structure that is over-actuated and dynamically optimized, we first isolate the configuration-related factors from the dynamics equations (allocation matrix $\boldsymbol{W}$ and total inertia matrix $\boldsymbol{J}_s$). Then we design an evaluation function of each flight structure from the dynamics/control perspective. Finally, a GA is implemented to find the suboptimal flight structure with improved computation efficiency.

A. Dynamic Property Evaluation

The allocation matrix $\boldsymbol{W}$ is derived as [12]:

$$\boldsymbol{W} = \bar{\boldsymbol{P}}\bar{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{I}_3 & \cdots & \boldsymbol{I}_3 \\ \hat{\boldsymbol{d}}_1 & \cdots & \hat{\boldsymbol{d}}_n \end{bmatrix} \begin{bmatrix} {}^{S}_{1}\boldsymbol{R} & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & {}^{S}_{n}\boldsymbol{R} \end{bmatrix}, \tag{11}$$

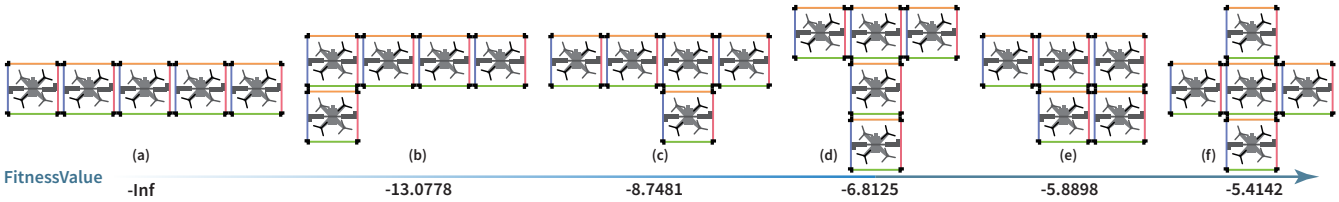


Fig. 3: **The fitness values for different flight structures composed of five same-weighted modules.** Configuration (a) has $-$Inf fitness value as it is under-actuated, while (f) has the maximum fitness value (optimal dynamics property) due to its symmetric configuration.

where $\bar{\boldsymbol{P}} \in \mathbb{R}^{6\times 3n}$, $\bar{\boldsymbol{R}} \in \mathbb{R}^{3n\times 3n}$, $\hat{\boldsymbol{d}}_i$ is the skew symmetric matrix of $\boldsymbol{d}_i$. From the formulation of $\bar{\boldsymbol{P}}$, we can easily find that the translational dynamics (the first three rows of $\boldsymbol{W}$) are independent of the configuration. Therefore, we only focus on the rotational dynamics (the last three rows of $\boldsymbol{W}$) in this optimization problem. Besides, the total inertia $\boldsymbol{J}_{\mathcal{S}}$ is also related to the structure configuration (see Eq. (6)) which is not neglectable. Therefore, we define a matrix $\bar{\boldsymbol{D}}$ as:

$$\bar{\boldsymbol{D}} = \boldsymbol{J}_{\mathcal{S}}^{-1}\hat{\boldsymbol{D}} = \boldsymbol{J}_{\mathcal{S}}^{-1}\begin{bmatrix}\hat{\boldsymbol{d}}_1 & \cdots & \hat{\boldsymbol{d}}_n\end{bmatrix}. \tag{12}$$

Calculating the required thrust energy index $\|\boldsymbol{T}\|^2$ and neglecting $\boldsymbol{\Omega}_{\mathcal{S}} \times \boldsymbol{J}_{\mathcal{S}}\boldsymbol{\Omega}_{\mathcal{S}}$, we have:

$$\begin{aligned}\|\boldsymbol{T}\|^2 = \|\boldsymbol{F}\|^2 &= \boldsymbol{u}_{\Omega}^{\mathsf{T}}\hat{\boldsymbol{D}}^{\dagger T}\bar{\boldsymbol{R}}\bar{\boldsymbol{R}}^{\mathsf{T}}\hat{\boldsymbol{D}}^{\dagger}\boldsymbol{u}_{\Omega} = \dot{\boldsymbol{\Omega}}_{\mathcal{S}}^{\mathsf{T}}\bar{\boldsymbol{D}}^{\dagger T}\bar{\boldsymbol{D}}^{\dagger}\dot{\boldsymbol{\Omega}}_{\mathcal{S}} \\ &\leqslant \sigma_{\max}(\bar{\boldsymbol{D}}^{\dagger})^2\|\dot{\boldsymbol{\Omega}}_{\mathcal{S}}\|^2 \quad \forall \dot{\boldsymbol{\Omega}}_{\mathcal{S}},\end{aligned} \tag{13}$$

where $(\cdot)^{\dagger}$ is the Moore–Penrose inverse of a matrix, and $\sigma_{\max}(\cdot)$ is the maximum singular value of a matrix.

Based on the above analysis, we formulate the flight structure evaluation function as :

$$\underset{\bar{\boldsymbol{D}}}{\operatorname{argmax}} \quad -\lambda_1 \operatorname{cond}(\bar{\boldsymbol{D}}) - \lambda_2 \sigma_{\max}(\bar{\boldsymbol{D}}^{\dagger})^2, \tag{14}$$

where $\operatorname{cond}(\cdot)$ is the condition number of a matrix. The left half of the objective function $-\operatorname{cond}(\bar{\boldsymbol{D}})$ considers the controllability where the full-actuation constraint is implicitly included ($\operatorname{cond}(\bar{\boldsymbol{D}}) = \text{Inf}$ if $\operatorname{rank}(\bar{\boldsymbol{D}}) < 3$); the right half $-\sigma_{\max}(\bar{\boldsymbol{D}}^{\dagger})^2$ characterizes the thrust minimization criteria (Eq. (13)), λ_{1-2} are the weighting parameters.

B. Genetic Algorithm

Given the number of modules n, the total number of different structures can be analytically calculated [33], and the method of enumerating all feasible $\boldsymbol{A}$ to find an optimal flight structure has been introduced in [10]. However, as n increases, the number of $\boldsymbol{A}$ grows factorially and thus it demands excessive computation. To overcome the limitation of existing methods, we propose a GA to find a suboptimal flight structure configuration with better efficiency. Different from the serially connected structure handled by [18], we implement the GA (see Alg. 1) to deal with more complicated tree representation in this work.

At first, each chromosome in the population is randomly initialized as a serial chain, *e.g.* a chain with five modules in Fig. 3(a). In each generation, (i) we first evaluate the fitness of the population; then (ii) we pick some chromosome from the population through a tournament; (iii) every picked chromosome is utilized to generate new children by crossover operation; (iv) we evaluate the existing population along with their children and select new population from them as the next generation. The fitness evaluation and crossover operation as two main components will be introduced in detail next, which correspond to **computeFitness** (Alg. 2) and **Crossover** (Alg. 3), respectively.

Algorithm 1: Genetic Algorithm

Input : Number of UAV modules: n,
Mass of each module: $m_{1..n}$,
Inertia of each module: $J_{1..n}$
Ouput : Optimized structure: A^*
Params: Maximum population size: PopSize,
Maximum generations: GSize,
Tournament size: TSize,
Number of children: CSize,
Crossover probability: CrossP,
Number of generations for convergence check: K

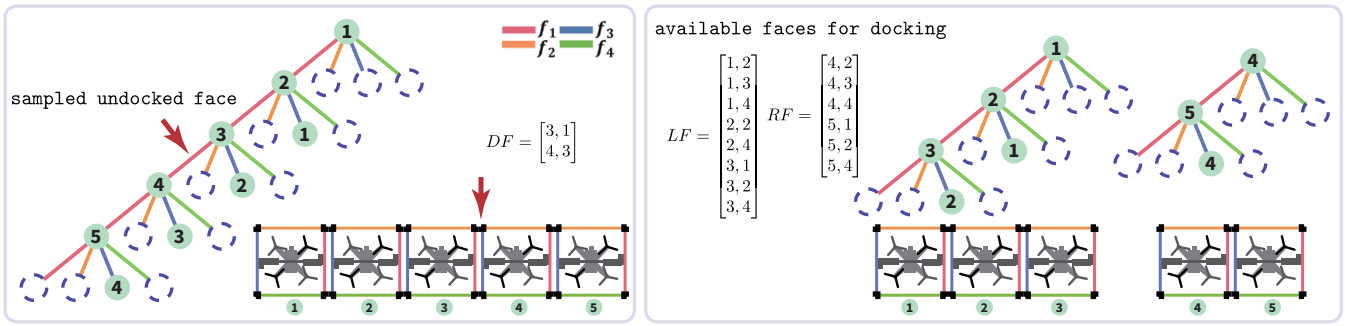
```
// Initialize population and fitness
Pop ← Initialize(PopSize, n) ∈ ℝ^{n×4×PopSize};
Fit ← computeFitness(Pop); see Alg. 2
// Generation iteration
for G_i ∈ 1 ··· GSize do
    NewPop ← Pop;
    for i ∈ 1 ··· TSize do
        // Tournament selection
        idx ← TSelect(Pop, Fit, TSize);
        Chromo ← Pop(:, :, idx);
        // Generate children
        for j ∈ 1 ··· CSize do
            if rand < CrossP then
                NewPop(:, :, end + 1) ← Crossover(Chromo)
            else
                NewPop(:, :, end + 1) ← Chromo;
    NewFit ← computeFitness(NewPop, m_{1..n}, J_{1..n});
    // Select new population
    Pop, Fit ← PopSelect(NewPop, NewFit, PopSize);
    if max(Fit) repeat for K generations then
        break // Optimization converges
A* ← Pop(:, :, 1);
```

Algorithm 2: computeFitness

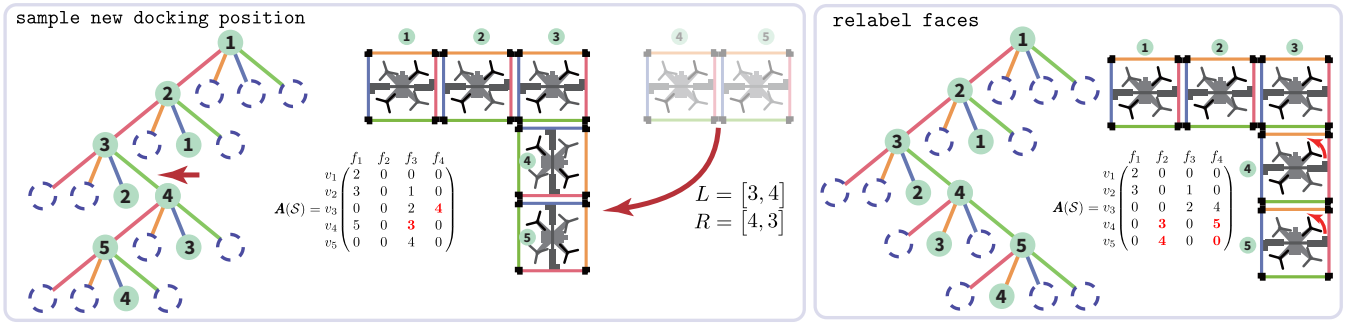
Input : Pop, $m_{1..n}$, $J_{1..n}$
Ouput : Fitness
Params: Quadcopter size in Eq. (2): l
The root of tree representation: root
Weighing parameters: λ_1, λ_1

```
Fitness ← ∅;
for Chromo ∈ Pop do
    d ← PosTreeSearch(root, Chromo, Step);
    Js ← Eq. (6), D ← Eq. (12);
    FValue ← −λ1 cond(D) − λ2 max(svd(D))^2;
    Fitness ← Fitness ∪ {FValue}
```

Fitness evaluation: To evaluate the optimality of different structure configurations, we choose Eq. (14) as the fitness function, which requires both the position vector $\boldsymbol{d_i}$ and total inertia matrix $\boldsymbol{J}_{\mathcal{S}}$ derived from the AIM of a structure for evaluation. Following the method introduced in Sec. III-B, the position of each module is first decided with the Depth-First tree traversal algorithm. And then $\bar{\boldsymbol{D}}$ matrix is built with Eqs. (6) and (12) for fitness value calculation. The



(a) Breaking the initial configuration into two sub-trees.



(b) Forming a new configuration by connecting them with a new label assignment.

Fig. 4: **Steps of a crossover operation.** The crossover operation divides the original tree into two small trees and reconnects them to build a new one. Through the crossover operation, all the feasible configurations can be acquired.

Algorithm 3: Crossover

Input : Chromo
Ouput : NewChromo
Feasible ← False;
while ¬*Feasible* **do**
  $ChR \leftarrow$ Chromo, $r \leftarrow$ randomInt$(1, n)$;
  // Find feasible docking faces, and current docking face pairs
  $[LF, RF, DF] \leftarrow$ DFSTreeSearch(r, ChR);
  // Delete current docking connections
  $ChR(DF(1,:)) \leftarrow 0$, $ChR(DF(2,:)) \leftarrow 0$;
  // Random select a connecting face L
  $L \leftarrow LF($randomInt$(1,$ length$(LF)), :)$;
  // Random select a connecting face R
  $R \leftarrow RF($randomInt$(1,$ length$(RF)), :)$;
  // Connect as a new tree
  $ChR(L) \leftarrow R(1,1)$, $ChR(R) \leftarrow L(1,1)$;
  // Rotate small tree
  $ChR \leftarrow$ RotateFaces(ChR, RF);
  // Check new tree with no overlap
  Feasible ← CheckFeasible(ChR);
NewChromo ← ChR;

detail of this function is described in Alg. 2. In Fig. 3, some structure configurations with five same-weighted modules are plotted and evaluated with our proposed fitness function as examples. We can easily find the Fig. 3(a) has the minimum fitness value due to under-actuation, while plus shape configuration Fig. 3(f) has the maximum fitness value. The fitness function will be studied and discussed in Sec. V.

Crossover operation: As demonstrated in Fig. 4, we define the crossover operation as breaking the tree structure into two subtrees and reconnecting them into a new tree. For a given configuration input, (i) we first randomly pick a module r as the breaking point, and utilize a Depth-First Tree Search algorithm to break the tree into two small trees, then we collect all the possible docking faces on the left tree and the right tree, as LF and RF respectively (see Fig. 4(a)); (ii) With one randomly picked element from both sets (L and R) as the new pair of docking face, we connect two subtrees into a new one, then we rotate the faces of the small tree to make sure the position relationship is identical to the Step matrix Fig. 4(b) and check the feasibility of the new structure. This function is summarized in Alg. 3.

V. Evaluation

Through a series of studies, we demonstrated that the proposed method (i) effectively solved the flight structure optimization problem with a large number of modules with different weights, (ii) gradually improved the dynamic properties of an over-actuated flight structure through the proposed crossover operation and fitness evaluation, and (iii) significantly outperformed a traditional emulation-based algorithm in terms of computing time.

A. Simulation Setup

Utilizing the Simscape module of Matlab Simulink, we developed a simulator where the characteristics of real systems were included, such as control frequencies, motor dynamics, thrust saturation, and measurement noise. Implementing the hierarchical controller developed in our previous work [19, 20, 34, 35], we could compare the dynamic properties of generated flight structures by testing their trajectory-tracking performance in simulation.

B. Evaluation Results

Optimizing a complex structure: In this study, we demonstrated that the proposed GA could scale up to optimize flight structure configuration for a large swarm with 30 heterogeneous modules (*Case 1*) and another with 37 modules (*Case 2*). The weight of each module was randomly

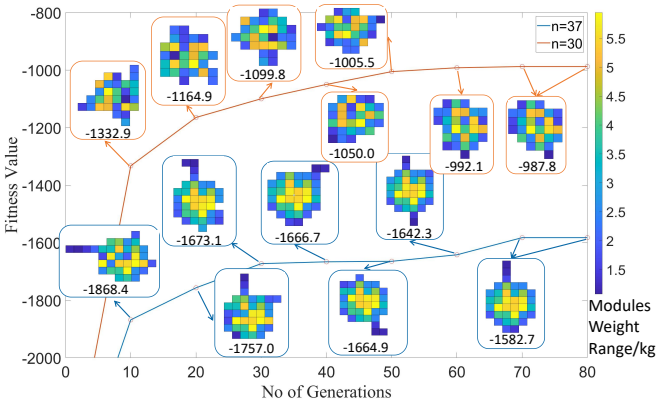


Fig. 5: **The optimization process of generating an optimal flight structure for a large modular UAV system swarm with different-weighted modules.** Two cases with $n = 30$ and $n = 37$ are presented to illustrate the evolution of the optimization process.

sampled from $1-5.5$ kg, indicated by the color bar. As seen in Fig. 5, During the optimization process, the fitness value for *Case 1* ($n = 30$) gradually improved from $-Inf$ and eventually converged to -1582.7 after 80 generations. The final flight structure was over-actuated and formed an approximately symmetric configuration with heavier modules in the middle, which had better controllability across all directions. A similar result could be observed for *Case 2* ($n = 37$) as well.

Generating efficient flight structure: Four flight structures (the 10th, 20th, 40th, and 80th generations) generated along the optimization process of *Case 1* in Fig. 5, who have an increasing fitness score, were selected to evaluate their dynamics properties in tracking a challenging 6-DoF trajectory. The thrust energy consumed by the structures to track the trajectory and the corresponding tracking errors were plotted Fig. 6. Tab. I further listed the accumulated energy cost derived from thrust and the tracking RMS errors of each structure along the entire trajectory. Despite these 4 flight structures being over-actuated and could independently track position and orientation commands, the converged structure configuration ($G_i(80)$) performed the best in terms of the least energy cost and RMS errors. This configuration also yields the highest fitness value, demonstrating its efficacy in ensuring over-actuation with better dynamics properties.

Computation speed: To demonstrate the proposed method's improvement in computation efficiency, we implemented a classic emulation-based method [10, 11] as the baseline and compared its computing time and converged fitness with those of the proposed GA with two population sizes ($GA1$ and $GA2$; see Tab. II for detailed parameters).

TABLE I: **Correspondence between structures' fitness value rank and their flying performance**. Four structures (the 10th, 20th, 40th, and 80th generations) that correspond to the *Case 1* of Fig. 5 are selected to compare their dynamics properties in simulation, and the converged structure indeed performs better than the others.

Structure	$G_i(10)$	$G_i(20)$	$G_i(40)$	$G_i(80)$
Fitness value	-1332.9	-1164.9	-1050.0	-987.8
$\sum\|\boldsymbol{T}\|^2(10^7)$	5.8587	5.8521	5.8432	5.8401
Pos RMS error (m)	0.1889	0.1883	0.1878	0.1874
Att RMS error (rad)	0.1693	0.1407	0.1221	0.1171

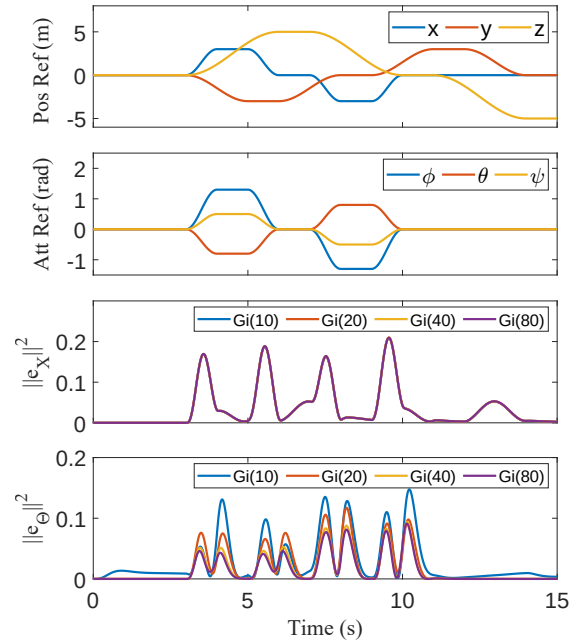


Fig. 6: **Comparison of the tracking performance of different flight structures composed of 30 different-weighted modules.** Although all 4 structures are over-actuated, some outperform others in terms of trajectory tracking accuracy due to better controllability.

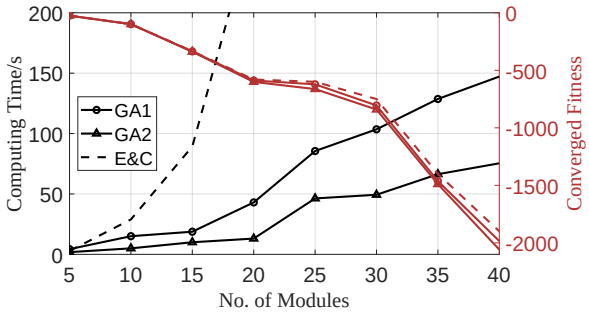


Fig. 7: **Computation speed and optimality of the proposed GA and the baseline enumeration method.** The GA with two population sizes performs significantly faster than that of the baseline. Though the global optimality is not guaranteed, the GA could still converge to a suboptimal solution, showing a good trade-off between computation efficiency and optimality.

Using a desktop with AMD Ryzen9 5950X CPU and 64.00 GB RAM, the results are shown in Fig. 7. The proposed GA significantly shortened the required computing time compared with the baseline, especially when the number of modules n became large. Meanwhile, the converged suboptimal configuration found by both settings was close to the global optimal one in terms of fitness. Of note, lowering the population size (*i.e.* $GA1$ *vs.* $GA2$) could reduce computation, but would sacrifice the optimality, which implies the necessity of balancing optimality and computation efficiency.

TABLE II: **Genetic Algorithm Parameters**

Parameter	PopSize	GSize	TSize	CSize	CrossP	K	Set
Value	3000	100	100	30	0.95	10	$GA1$
	1000	100	100	30	0.95	10	$GA2$

VI. Conclusion

In this paper, we devised a novel optimization method aimed at finding efficient flight structures for modular UAV systems. This method utilized genetic algorithms (GA) to address an energy-minimizing objective function while satisfying over-actuation constraints. By representing the configuration as a tree structure for customized crossover operations and as a position vector for fitness evaluation, the GA effectively tackled the optimization problem with significantly reduced computational costs compared to existing emulation-based algorithms for flight structure optimization. Based on our customized modular UAV system, simulation studies illustrated (i) the algorithm's ability to identify efficient flight structures for a large number of modules, (ii) the superior dynamic performance of the resulted flight structure in tracking challenging trajectories, and (iii) the improvement in computational efficiency. Looking ahead, our future plan includes conducting experiments with physical platforms and further enhancing computational capabilities to enable online reconfiguration based on varying task constraints.

References

[1] J. Seo, J. Paik, and M. Yim, "Modular reconfigurable robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 63–88, 2019.

[2] R. Oung and R. D'Andrea, "The distributed flight array: Design, implementation, and analysis of a modular vertical take-off and landing vehicle," *International Journal of Robotics Research (IJRR)*, vol. 33, no. 3, pp. 375–400, 2014.

[3] B. Mu and P. Chirarattananon, "Universal flying objects: Modular multirotor system for flight of rigid objects," *IEEE Transactions on Robotics (T-RO)*, vol. 36, no. 2, pp. 458–471, 2019.

[4] D. Saldana, B. Gabrich, G. Li, M. Yim, and V. Kumar, "Modquad: The flying modular structure that self-assembles in midair," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[5] D. Saldana, P. M. Gupta, and V. Kumar, "Design and control of aerial modules for inflight self-disassembly," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 3410–3417, 2019.

[6] B. Gabrich, D. Saldana, V. Kumar, and M. Yim, "A flying gripper based on cuboid modular robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[7] M. Zhao, K. Kawasaki, K. Okada, and M. Inaba, "Transformable multirotor with two-dimensional multilinks: modeling, control, and motion planning for aerial transformation," *Advanced Robotics*, vol. 30, no. 13, pp. 825–845, 2016.

[8] J. Xu, D. S. D'Antonio, and D. Saldaña, "Modular multi-rotors: From quadrotors to fully-actuated aerial vehicles," *arXiv preprint arXiv:2202.00788*, 2022.

[9] J. Xu, D. S. D'Antonio, and D. Saldaña, "H-modquad: Modular multi-rotors with 4, 5, and 6 controllable dof," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[10] I.-M. Chen and J. W. Burdick, "Enumerating the non-isomorphic assembly configurations of modular robotic systems," *International Journal of Robotics Research (IJRR)*, vol. 17, no. 7, pp. 702–719, 1998.

[11] J. Liu, Y. Wang, S. Ma, and Y. Li, "Enumeration of the non-isomorphic configurations for a reconfigurable modular robot with square-cubic-cell modules," *International Journal of Advanced Robotic Systems*, vol. 7, no. 4, p. 31, 2010.

[12] B. Gabrich, D. Saldana, and M. Yim, "Finding structure configurations for flying modular robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[13] J. Feng and J. Liu, "Enumerating the nonisomorphic configurations of a modular reconfigurable robot," *ASME Journal of Mechanisms and Robotics*, vol. 14, no. 6, p. 064503, 2022.

[14] K. Stoy and D. Brandt, "Efficient enumeration of modular robot configurations and shapes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[15] N. Gandhi, D. Saldana, V. Kumar, and L. T. X. Phan, "Self-reconfiguration in response to faults in modular aerial systems," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 2522–2529, 2020.

[16] J. Xu and D. Saldaña, "Finding optimal modular robots for aerial tasks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[17] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, 2021.

[18] I.-M. Chen and J. W. Burdick, "Determining task optimal modular robot assembly configurations," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1995.

[19] Y. Su, P. Yu, M. Gerber, L. Ruan, and T.-C. Tsao, "Nullspace-based control allocation of overactuated uav platforms," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 4, pp. 8094–8101, 2021.

[20] P. Yu, Y. Su, M. J. Gerber, L. Ruan, and T.-C. Tsao, "An over-actuated multi-rotor aerial vehicle with unconstrained attitude angles and high thrust efficiencies," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 4, pp. 6828–6835, 2021.

[21] B. Li, L. Ma, D. Huang, and Y. Sun, "A flexibly assembled and maneuverable reconfigurable modular multi-rotor aerial vehicle," *IEEE/ASME Transactions on Mechatronics (TMECH)*, vol. 27, no. 3, pp. 1704–1714, 2021.

[22] B. Gabrich, G. Li, and M. Yim, "Modquad-dof: A novel yaw actuation for modular quadrotors," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[23] M. Zhao, K. Kawasaki, T. Anzai, X. Chen, S. Noda, F. Shi, K. Okada, and M. Inaba, "Transformable multirotor with two-dimensional multilinks: Modeling, control, and whole-body aerial manipulation," *International Journal of Robotics Research (IJRR)*, vol. 37, no. 9, pp. 1085–1112, 2018.

[24] H.-N. Nguyen, S. Park, J. Park, and D. Lee, "A novel robotic platform for aerial manipulation using quadrotors as rotating thrust generators," *IEEE Transactions on Robotics (T-RO)*, vol. 34, no. 2, pp. 353–369, 2018.

[25] Y. Su, L. Ruan, P. Yu, C.-H. Pi, M. J. Gerber, and T.-C. Tsao, "A fast and efficient attitude control algorithm of a tilt-rotor aerial platform using inputs redundancies," *IEEE Robotics and Automation Letters (RA-L)*, vol. 7, no. 2, pp. 1214–1221, 2021.

[26] Y. Su, P. Yu, M. Gerber, L. Ruan, and T. Tsao, "Fault-tolerant control of an over-actuated uav platform built on quadcopters and passive hinges," *IEEE/ASME Transactions on Mechatronics (TMECH)*, vol. 29, no. 1, pp. 602–613, 2024.

[27] G. Li, B. Gabrich, D. Saldana, J. Das, V. Kumar, and M. Yim, "Modquad-vi: A vision-based self-assembling modular quadrotor," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[28] Y. Litman, N. Gandhi, L. T. X. Phan, and D. Saldaña, "Vision-based self-assembly for modular multirotor structures," *IEEE Robotics and Automation Letters (RA-L)*, vol. 6, no. 2, pp. 2202–2208, 2021.

[29] C. Leger and J. Bares, "Automated synthesis and optimization of robot configurations," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, 1998.

[30] P. Yu, Y. Su, L. Ruan, and T.-C. Tsao, "Compensating aerodynamics of over-actuated multi-rotor aerial platform with data-driven iterative learning control," *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 10, pp. 6187–6194, 2023.

[31] L. Ruan, C.-H. Pi, Y. Su, P. Yu, S. Cheng, and T.-C. Tsao, "Control and experiments of a novel tiltable-rotor aerial platform comprising quadcopters and passive hinges," *Mechatronics*, vol. 89, p. 102927, 2023.

[32] Y. Su, J. Li, Z. Jiao, M. Wang, C. Chu, H. Li, Y. Zhu, and H. Liu, "Sequential manipulation planning for over-actuated unmanned aerial manipulators," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[33] L. J. White, "Introduction to combinatorial mathematics (cl liu)," *SIAM Review*, vol. 11, no. 4, p. 634, 1969.

[34] Y. Su, C. Chu, M. Wang, J. Li, L. Yang, Y. Zhu, and H. Liu, "Downwash-aware control allocation for over-actuated uav platforms," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[35] Y. Su, J. Zhang, Z. Jiao, H. Li, M. Wang, and H. Liu, "Real-time dynamic-consistent motion planning for over-actuated uavs," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.