Flight Structure Optimization of Modular Reconfigurable UAVs

Yao Su[1], Ziyuan Jiao[1], Zeyu Zhang[1], Chi Chu[1,2], Jiarui Li[1,3], Hang Li[1], Meng Wang[1], Hangxin Liu[1]

***Abstract*— This paper presents a *genetic algorithm* that reconfigures modular Unmanned Aerial Vehicle (UAV) with different installed equipment to a flight structure better suited for task purposes. Existing work either design the flight structure with expert knowledge, which is restricted to a specific task, or search for an optimal one by enumeration-based algorithms that demand heavy computation. Based on a modular quadcopter MARVEL, who connects to a cubic docking frame through a passive gimbal mechanism and serves as an omni-directional thrust generator in a flight structure, the proposed algorithm efficiently finds the sub-optimal configuration that guarantees over-actuation and consumes the minimum control effort. After implementing a hierarchical controller in simulation, we validate that the flight structure generated from the proposed method can (i) effectively track challenging trajectories with coupled position and attitude commands and (ii) significantly reduce computational cost compared with traditional enumeration method.**

I. Introduction

By transforming to different structures, a modular UAV system shows a greater potential in versatility, robustness, and low cost compared with an aerial system with a fixed structure [1]. More specifically, each modular UAV can dock/undock with other modules to form a flight structure that is more efficient in performing tasks such as payload transportation [2, 3], object manipulation [4–7], and dynamic exploration [8, 9]. Designing an efficient flight structure usually requires the knowledge of human experts that accounts for certain task specifications. Automating the design process is also possible using algorithms that enumerate all possible flight structures and select the optimal one evaluated by a given metric [10–13].

However, as the number of modules increases in the modular UAV system, designing an efficient flight structure with human knowledge could become infeasible, and finding that structure through enumeration-based algorithms could suffer an exponential growth in computational cost. In addition, when modules are equipped with different types of sensors, interacting tools, or payloads, the configuration of the flight structure can significantly influence the dynamical properties of the system and the complexity of this problem can be further improved [14, 15]. Consequently, neither human knowledge nor existing algorithmic solutions could produce a desired flight structure of a modular UAV system with satisfactory efficiency.

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

[2] Department of Automation, Tsinghua University.

[3] Department of Advanced Manufacturing and Robotics, College of Engineering, Peking University.

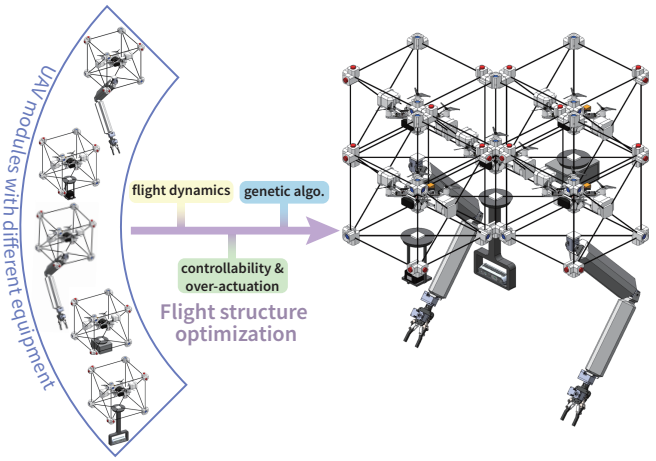


Fig. 1: **The optimal structure configuration with five modular UAVs in four different weights**. Each module is equipped with either a manipulator, an RGBD camera, a Lidar, or a computing unit, resulting in different dynamical properties. The proposed algorithm efficiently produces an over-actuated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

In this work, we first propose to utilize the **assembly incidence matrix (AIM)** representation [16] to describe the flight structure of a modular UAV system as a tree. Then, we transform the problem of finding an efficient flight structure with over-actuation capability and high energy-efficiency into the objective function of an optimization problem. A genetic algorithm with specialized crossover operation on the tree is formulated to accelerate the optimization process, which can find a sub-optimal flight structure with high computational efficiency.

To evaluate the proposed approach, we adopt a customized modular UAV system, MARVEL. Each module of MARVEL is a quadcopter connecting to a cubic docking frame through a 2-Degree-of-freedom (DoF) passive gimbal mechanism and thus can be treated as a **omni-directional thrust generator** after docking with each other horizontally and forming a flight structure. Specialized equipment with different weights or shapes can be installed on a module's docking frame as well. Fig. 1 illustrates some MARVEL modules and the optimized flight structure that is over-actuated [17, 18] to track challenging trajectories with coupled position and attitude commands.

Combining with a hierarchical controller, we deploy different flight structures in simulation and compare their trajectory tracking performance to validate the proposed optimization method. Then, we present the converging process of solving the optimization of a flight structure composed of up to 37 different-weighted MARVEL modules. The results not only demonstrate that the resulting flight structure can

efficiently achieve over-actuation and consume the least control effort, but also significantly accelerate the computation compared with that of existing enumeration-based methods.

A. 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, 19, 20], dynamics modelling [18, 21, 22], formation control [5], control allocation [17, 23, 24], and docking trajectory planning [25, 26]. In above cases, both the structure and module configurations of the system are designed and fixed. More recent work sought to modify the module configuration by reorienting or swapping them within the flight structure. More specifically, Gabrich *et al*. proposed a heuristic-based subgroup search algorithm that achieved better efficiency than enumerating different types of modules at each module location [14]. A mixed integer linear program was also formulated by Gandhi *et al*. to find the optimal module-to-position arrangement when some rotor faults happened [15]. However, the overall flight structure remained unchanged during the process, which could essentially prohibit it from being over-actuated or achieving higher energy efficiency.

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 the non-isomorphic configurations of a modular manipulator system. Following work extended the scope to various modular robot systems with improved computation efficiency [10–13]. Still, as the number of modules increases, the set of configurations enlarges factorially, and finding an effective configuration through exhaustive search becomes infeasible [16, 27]. On the other hand, utilizing *genetic algorithm* to find a suboptimal configuration of the modular robotic system has been demonstrated on a simple serial connected structure with a much better efficiency [16]. 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 the customized crossover operation for the genetic algorithm, a suboptimal flight structure configuration can be efficiently found.

B. Overview

We organize the remainder of the paper as follows. Sec. II describes the system configuration and the dynamical analysis of the flight structure. Then, we propose the configuration optimization algorithm of the flight structure in Sec. III. Sec. IV shows the simulation results with comprehensive evaluations. Finally, we conclude the paper in Sec. V.

II. System Configuration

A MARVEL 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. 2a, a flight structure can be formed after several MARVEL modules (indexed by $i = 1 \cdots n$) dock with others. As each MARVEL can be

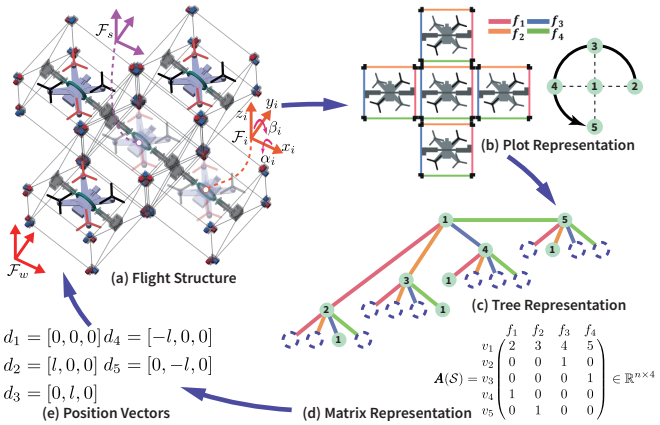


Fig. 2: **Coordinate systems and configuration representations of a flight structure.** A MARVAL module can connect to four others with its docking faces, and each module serves as an omni-directional thrust generator after connections, making the flight structure over-actuated. Different types of representation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

utilized as an omni-directional thrust generator, the resulted multirotor flight structure has the potential of being over-actuated to gain better maneuverability with independent position and orientation tracking. 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. 2b). Then we derive a simplified representation of the flight structure using a matrix and tree for 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 body frame's position as $\boldsymbol{\mathcal{X}}_\mathcal{S} = [x_\mathcal{S}, y_\mathcal{S}, z_\mathcal{S}]^\mathsf{T}$, 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 $\boldsymbol{\Omega}_\mathcal{S} = [p_\mathcal{S}, q_\mathcal{S}, r_\mathcal{S}]^\mathsf{T}$. Frames $\mathcal{F}_i$s are attached to the center of the i-th module. $\boldsymbol{d}_i = [x_i, y_i, 0]$ denotes the vector from module frame $\mathcal{F}_i$ to structure frame $\mathcal{F}_\mathcal{S}$.

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*. [16], we describe the flight structure as a tree where the module **1** is chosen as the root. Then the structure configuration can be characterized by the assembly incidence matrix (AIM).

Using the configuration in Fig. 2a as an example, the corresponding AIM is shown in Fig. 2d, where each row represents a module, while each column represents a docking face. $A_{ij} = k$ indicates the j-th face of module i is connected to module k. Of note, $A(\mathcal{S})$ is a normalized representation, which is independent of module size l.

We can calculate $\boldsymbol{d}_i$ from $A(\mathcal{S})$ with the following steps:

1) assuming the module **1** is at the origin $\boldsymbol{d}_1 = [0,0,0]$ and determining the position of each module $\boldsymbol{d}_i$ recursively with the *Step* matrix that describes the relative position

between two modules according to the docked face:

$$Step = \begin{matrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{matrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{pmatrix} \times l \tag{1}$$

2) calculating the geometric center position with $\boldsymbol{d}_o = \frac{1}{M}\sum_{i=1}^{n} m_i \boldsymbol{d}_i$;
3) shifting the geometric center of $\mathcal{S}$ to the origin with $\boldsymbol{d}_i = \boldsymbol{d}_i - \boldsymbol{d}_o$.

C. Flight Structure Dynamics

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

$$M\,\ddot{\boldsymbol{\mathcal{X}}}_{\mathcal{S}} = {}^{W}_{S}\boldsymbol{R}\Big(\sum_{i=1}^{n} {}^{S}_{i}\boldsymbol{R}\,T_i\hat{\boldsymbol{z}}\Big) + M\,g\hat{\boldsymbol{z}}, \tag{2}$$

where $M = \sum_{i=1}^{n} m_i$ is the total mass of the flight structure, with m_i the mass of module i, $\ddot{\boldsymbol{\mathcal{X}}}_{\mathcal{S}}$ 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:

$$\boldsymbol{J}_{\mathcal{S}}\,\dot{\boldsymbol{\Omega}}_{\mathcal{S}} = -\boldsymbol{\Omega}_{\mathcal{S}} \times \boldsymbol{J}_{\mathcal{S}}\boldsymbol{\Omega}_{\mathcal{S}} + \sum_{i=1}^{n} (\boldsymbol{d}_i \times {}^{S}_{i}\boldsymbol{R}\,T_i\hat{\boldsymbol{z}}), \tag{3}$$

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

$$\boldsymbol{J}_{\mathcal{S}} = \sum_{i=1}^{n} \boldsymbol{J}_i + \sum_{i=1}^{n} \begin{bmatrix} m_i y_i^2 & 0 & 0 \\ 0 & m_i x_i^2 & 0 \\ 0 & 0 & m_i(x_i^2 + y_i^2) \end{bmatrix}, \tag{4}$$

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:

$$\begin{bmatrix} \ddot{\boldsymbol{\mathcal{X}}}_{\mathcal{S}} \\ \dot{\boldsymbol{\Omega}}_{\mathcal{S}} \end{bmatrix} = \begin{bmatrix} \frac{1}{M}\,{}^{W}_{S}\boldsymbol{R} & 0 \\ 0 & \boldsymbol{J}_{\mathcal{S}}^{-1} \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} g\hat{\boldsymbol{z}} \\ -\boldsymbol{\Omega}_{\mathcal{S}} \times \boldsymbol{J}_{\mathcal{S}}\boldsymbol{\Omega}_{\mathcal{S}} \end{bmatrix}, \tag{5}$$

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

$$\boldsymbol{u} = \begin{bmatrix} \sum_{i=1}^{n} {}^{S}_{i}\boldsymbol{R}\,T_i\hat{\boldsymbol{z}} \\ \sum_{i=1}^{n} (\boldsymbol{d}_i \times {}^{S}_{i}\boldsymbol{R}\,T_i\hat{\boldsymbol{z}}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_{\mathcal{X}}(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \boldsymbol{J}_{\Omega}(\boldsymbol{\alpha},\boldsymbol{\beta}) \end{bmatrix} \boldsymbol{T}. \tag{6}$$

D. Force Decomposition-based Analysis

Utilizing the force decomposition method, we can transform the nonlinear relationship in Eq. (6) to a linear one by defining $\boldsymbol{F}$ as an intermediate variable [18, 24]:

$$\boldsymbol{F} = \begin{bmatrix} F_1 \\ \vdots \\ F_n \end{bmatrix} \in \mathbb{R}^{3n\times 1},\ F_i = \begin{bmatrix} \sin\beta_i \\ -\sin\alpha_i\cos\beta_i \\ \cos\alpha_i\cos\beta_i \end{bmatrix} T_i, \tag{7}$$

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

$$\boldsymbol{u}^d = \begin{bmatrix} \boldsymbol{J}_{\mathcal{X}}(\boldsymbol{\alpha},\boldsymbol{\beta}) \\ \boldsymbol{J}_{\Omega}(\boldsymbol{\alpha},\boldsymbol{\beta}) \end{bmatrix} \boldsymbol{T} = \boldsymbol{W}\boldsymbol{F}, \tag{8}$$

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 [28].

III. Flight Structure Optimization

To optimize the flight structure and generate an over-actuated platform with high energy efficiency, 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 objective function with them from the control perspective. Finally, a genetic algorithm is implemented for improved efficiency.

A. General Optimization Formulation

We formulate the general optimization problem of a flight structure as:

$$\underset{\boldsymbol{d}_1,\cdots,\boldsymbol{d}_n}{\operatorname{argmin}} \quad \|\boldsymbol{T}\|^2 \quad \forall \boldsymbol{u}^d \tag{9}$$

$$s.t \qquad rank(\boldsymbol{W}) = 6 \tag{10}$$

$$\frac{1}{M}\sum_{i=1}^{n} m_i \boldsymbol{d}_i = 0 \tag{11}$$

$$\mathcal{S}(\boldsymbol{d}_1, \cdots, \boldsymbol{d}_n) = 1 \tag{12}$$

where the objective function Eq. (9) is defined as minimizing the required thrust energy index $\|\boldsymbol{T}\|^2$ for all possible desired acceleration command. And the constraint Eq. (10) ensures over-actuation, while Eq. (11) ensures the geometric center of the structure is at the origin. Feasibility constraint Eq. (12) considers that all modules are connected with no overlaps, which is hard to derive in the analytical form.

B. Combinational Optimization Formulation

Looking at the derivation of allocation matrix $\boldsymbol{W}$, we can have [14]:

$$\boldsymbol{W} = \bar{\boldsymbol{P}}\bar{\boldsymbol{R}} = \begin{bmatrix} \boldsymbol{I}_3 & \cdots & \boldsymbol{I}_3 \\ \hat{\boldsymbol{d}}_1 & \cdots & \hat{\boldsymbol{d}}_n \end{bmatrix} \begin{bmatrix} {}^{S}_{1}\boldsymbol{R} & \cdots & 0 \\ & \ddots & \\ 0 & \cdots & {}^{S}_{n}\boldsymbol{R} \end{bmatrix}, \tag{13}$$

where $\bar{\boldsymbol{P}} \in \mathbb{R}^{b\times 3n}$, $\bar{\boldsymbol{R}} \in \mathbb{R}^{3n\times 3n}$, $\hat{\boldsymbol{d}}_i$ is the skew symmetric matrix of $\boldsymbol{d}_i$. And we can easily find that the translational dynamics are independent of the configuration. Therefore, we only focus on the rotational dynamics part in this optimization problem. Besides, the total inertia $\boldsymbol{J}_{\mathcal{S}}$ is also related to the structure configuration (see Eq. (4)) which is not neglectable. Therefore, we define a matrix $\bar{\boldsymbol{D}}$ as:

$$\bar{\boldsymbol{D}} = \boldsymbol{J}_{\mathcal{S}}^{-1}\begin{bmatrix} \hat{\boldsymbol{d}}_1 & \cdots & \hat{\boldsymbol{d}}_n \end{bmatrix}, \tag{14}$$

then Eq. (9) can be simplified as:

$$\begin{aligned} &\underset{\hat{\boldsymbol{d}}_1,\cdots,\hat{\boldsymbol{d}}_n}{\operatorname{argmin}} \quad \boldsymbol{u}_{\Omega}^{T}\bar{\boldsymbol{D}}^{\dagger T}\bar{\boldsymbol{D}}^{\dagger}\boldsymbol{u}_{\Omega} \\ \leqslant\ &\underset{\hat{\boldsymbol{d}}_1,\cdots,\hat{\boldsymbol{d}}_n}{\operatorname{argmin}} \sigma_{max}((\bar{\boldsymbol{D}}\bar{\boldsymbol{D}}^{T})^{-1})\|\boldsymbol{u}_{\Omega}\|^2 \quad \forall \boldsymbol{u}_{\Omega} \end{aligned} \tag{15}$$

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

Utilizing the AIM representation, we can reformulate Eq. (9)-Eq. (12) as a combinational optimization problem:

$$\underset{\boldsymbol{A}(\mathcal{S})}{\operatorname{argmax}} \quad -cond((\bar{\boldsymbol{D}}\bar{\boldsymbol{D}}^{T})^{-1}) - \sigma_{max}((\bar{\boldsymbol{D}}\bar{\boldsymbol{D}}^{T})^{-1}), \tag{16}$$

where $cond(\cdot)$ is the condition number of a matrix. The left half of the objective function considers the controllability

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); see Alg. 3
            else
                NewPop(:, :, end + 1) ← Chromo;
            end
        end
    end
    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
    end
end
A* ← Pop(:, :, 1);
```

where the full-actuation constraint (Eq. (10)) is implicitly included; the right half characterizes the thrust minimization criteria (Eq. (15)).

Given the number of modules n, the total number of different structures is analytically calculated [29], 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 Genetic Algorithm to find a suboptimal flight structure configuration with better efficiency.

C. Genetic Algorithm

Inspired by the biological evolution process, the Genetic Algorithm is a population based search algorithm that solves both constrained and unconstrained optimization problems with natural selection [30, 31]. Different from the serially connected structure handled by [16], we implement the genetic Algorithm (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. 3a. In each generation, (i) we first evaluate the fitness of all 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 2: computeFitness

Input : Pop, $m_{1..n}$, $J_{1..n}$
Ouput : $Fitness$
Params: Matrix from Eq. (1): $Step$
The root of tree representation: $root$

```
Fitness ← ∅;
for Chromo ∈ Pop do
    // Start with root and evaluate positions
     of all nodes
    d ← PosTreeSearch(root, Chromo, Step);
    Center ← sum(d. * m, 1)/M;
    d ← d − Center ;        // Shift center to (0,0)
    Js ←Eq. (4), D ←Eq. (14);
    FValue ← −cond((DD^T)^{−1}) − max(svd((DD^T)^{−1}));
    Fitness ← Fitness ∪ {FValue}
end
```

Algorithm 3: Crossover

Input : $Chromo$
Ouput : $NewChromo$

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

Fitness evaluation: To evaluate the optimality of different structure configurations, we choose Eq. (16) as the fitness function, which requires both the position vector $\boldsymbol{d_i}$ and total inertia matrix $\boldsymbol{J_S}$ from the AIM of a structure for evaluation. Following the method introduced in Sec. II-B, the position of each module is first decided utilizing the Depth-First tree traversal algorithm. And then $\bar{\boldsymbol{D}}$ matrix is build with Eqs. (4) and (14) for fitness value calculation. The 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. IV.

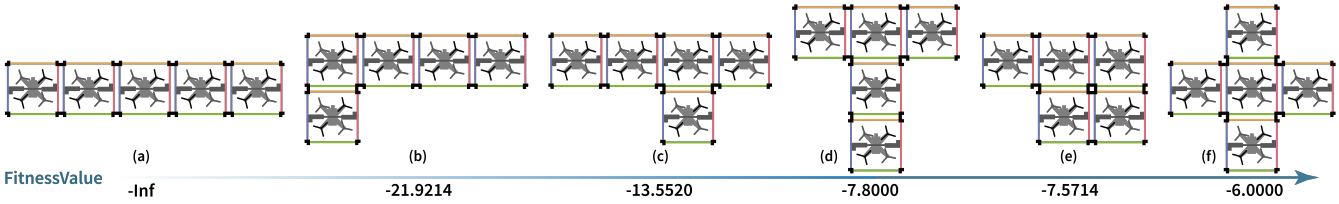
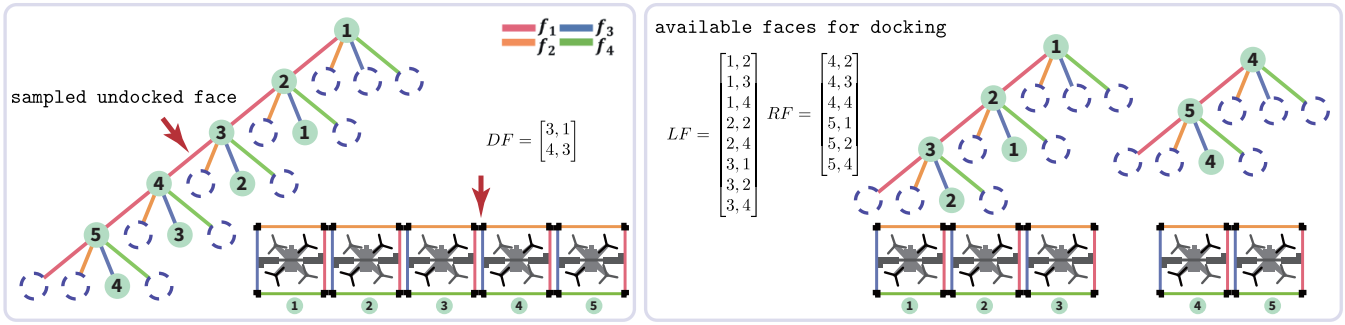
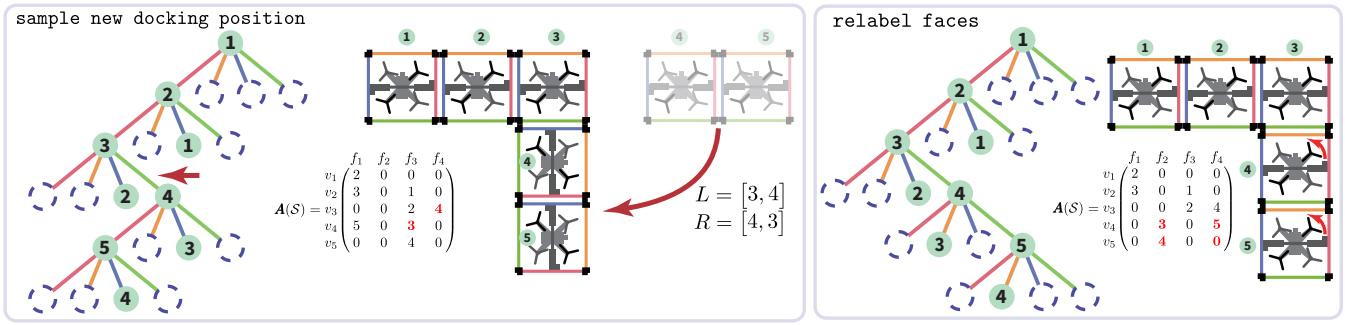


Fig. 3: **The fitness values for different flight structures composed by five same-weighted modules.** (a) The configuration with $-Inf$ fitness value as it is under-actuated, while (f) has the maximum value because its symmetric configuration leads to better.



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



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

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

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.

IV. Simulation

Through a series of simulated studies, we demonstrated that the proposed method (i) successfully generated an over-actuated flight structure with good energy efficiency when possible, (ii) effectively solved the flight structure optimization problem a large number of modules with different weights, 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 realistic simulation platform where the characteristics of real system were included, such as control frequencies, motor dynamics, thrust saturation, and measurement noise. Implementing the hierarchical controller developed in our previous work [17, 18, 32], we could check whether or not the generated flight structures were desired by testing their flying performance in simulation.

B. Simulation Results

Generating effective flight structure: Five flight structures generated by the optimization process (see Fig. 3b-f), who have a increasing fitness score, were selected to evaluate their capability and efficiency in tracking a challenging 6-DoF trajectory shown in Fig. 5. The thrust energy consumed by the platforms to track the trajectory and the corresponding tracking errors were plotted in Fig. 5. Tab. I further listed the accumulated energy cost derived from thrust and the tracking Root-Mean-Square (RMS) error of each structure along the entire trajectory.

Despite these 5 flight structures are over-actuated and could independently track position and orientation com-

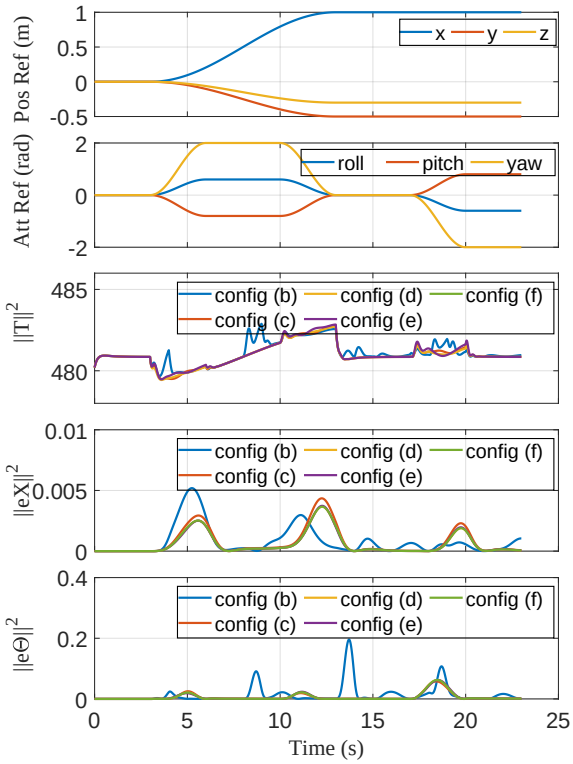


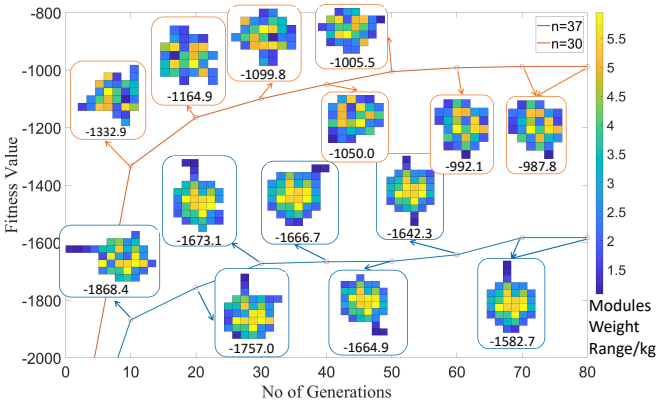
Fig. 5: **Comparison of the tracking performance of different flight structures composed by five same-weighted modules.** Although all 5 structures are over-actuated, some outperform others due to better controllability.

TABLE I: **Correspondence between structures' fitness value Rank and their flying performance**. The structure f has the highest fitness value and indeed performs better than the others.

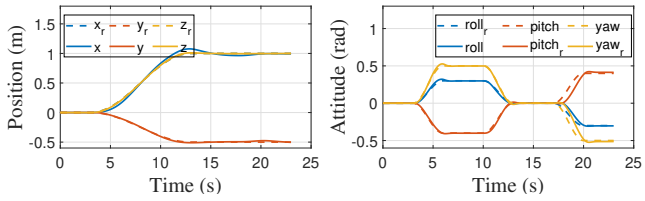
Structure	b	c	d	e	f
Fitness value	-21.9214	-13.5520	-7.8000	-7.5714	-6.0000
$\sum\|\boldsymbol{T}\|^2(10^6)$	1.1070	1.1069	1.1067	1.1068	1.1066
Pos RMS error (mm)	28.2400	24.7375	24.7348	24.7049	24.7007
Att RMS error (rad)	0.1156	0.0773	0.0768	0.0768	0.075

mands, the configuration in Fig. 3f performed the best in terms of the least energy cost and RMS errors. This configuration also yield the highest fitness value, demonstrating its efficacy in ensuring over-actuation with better controllability while reducing energy consumption.

Optimizing a complex structure: In this study, we demonstrated that the proposed genetic algorithm could scale up to optimize flight structure configuration for a large swarm with 30 modules (*Case 1*) and another with 37 (*Case 2*). The weight of each module was randomly sampled from $1-5.5\ kg$, indicated by the color bar. As seen in Fig. 6a, During the evolution of 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 a approximately symmetric configuration with heavier modules in the middle, which had a better controllability across all directions. A similar result could be observed for *Case 2* ($n=37$) as well. The final flight structure of *Case 2* was further deployed in the simulation to validate its tracking performance, as shown in Fig. 6b.



(a) The optimization process for two large swarms.



(b) Position and attitude tracking of the converged configuration.

Fig. 6: **Generating an efficient flight structure for a large modular UAV system swarm with different-weighted modules.** (a) Two cases with $n=30$ and $n=37$ are presented to illustrate the evolution of the optimization process. (b) The trajectory tracking performance of the converged flight structure of *Case 2*.

TABLE II: **Genetic Algorithm Parameters**

Parameter	*PopSize*	*GSize*	*TSize*	*CSize*	*CrossP*	*K*	*Set*
Value	3000	100	100	30	0.95	10	*GA*1
	1000	100	100	30	0.95	10	*GA*2

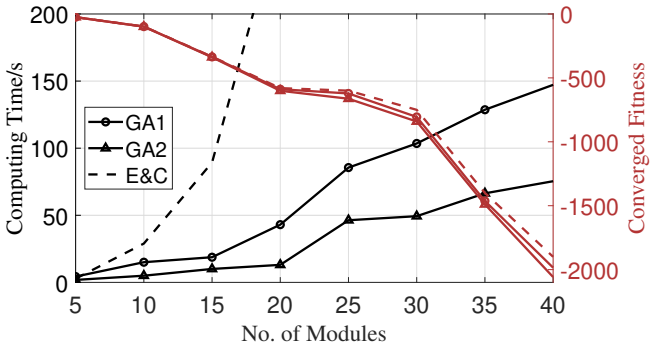


Fig. 7: **Computation speed and optimality of the proposed genetic algorithm and the baseline enumeration method.** The genetic algorithm with two population sizes performs significantly faster than that of the baseline. Though the global optimality is not guaranteed, the genetic algorithm could still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

Computation speed: To demonstrate the proposed method's improvement in computation efficiency, we implemented an classic emulation-based method [10, 11] as the baseline and compared its computing time and converged fitness with those of the proposed genetic algorithm with two population sizes ($GA1$ and $GA2$; see Tab. II for detailed parameters). Using a desktop with AMD Ryzen9 5950X CPU and 64.00 GB RAM, the results are shown in Fig. 7. The proposed genetic algorithm significantly shortened the

required computing time compared with the baseline, especially when the number of modules n became large. Meanwhile, the converged sub-optimal configuration found by the 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 of the optimality, which implies the necessity of balancing optimality and efficiency.

V. Conclusion

In this paper, we explored a new topic in modular UAV system—finding an efficient flight structure to be composed to with an acceptable computational budget. Utilizing our customized MARVEL system, we first mathematically formulated this optimization problem with designed energy minimizing objective function and over-actuation constraints. Representing the configuration as a tree structure and combining with customized crossover operation, a genetic algorithm is further proposed to solve the optimization problem. Our simulation studies demonstrated (i) the effectiveness of the proposed method, (ii) the capability for finding an efficient flight structure for a large number of modules, and (iii) the improvement in computational efficiency. Moving forward, we plan to conduce experiment with physical platform and to further improve computation efficiency to achieve online reconfiguration given different task constraints.

Acknowledgement: The authors thank Miss. Zhen Chen at BIGAI for the help on figures.

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 *Proceedings of 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 *Proceedings of 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 *Proceedings of 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] 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.

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

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

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

[17] 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.

[18] 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.

[19] 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)*, 2021.

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

[21] 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.

[22] 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.

[23] 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.

[24] Y. Su, P. Yu, M. Gerber, L. Ruan, and T. Tsao, "Fault-tolerant control of overactuated multirotor uav platform under propeller failure," *IEEE/ASME Transactions on Mechatronics (TMECH) (submitted)*, 2022.

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

[26] 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.

[27] 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.

[28] Y. Su, *Compensation and Control Allocation with Input Saturation Limits and Rotor Faults for Multi-Rotor Copters with Redundant Actuations*. PhD thesis, University of California, Los Angeles, 2021.

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

[30] D. Whitley, "A genetic algorithm tutorial," *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[31] 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.

[32] 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 *Proceedings of International Conference on Intelligent Robots and Systems (IROS)*, 2022.