From Docking Station to Docking Station: Completing Tasks in Minimum Time by Cooperative UAV Fleets

Baixin Wan, Feng Shan, Jianping Huang

baixinwan@seu.edu.cn

School of Computer Science and Engineering
Southeast University, Nanjing, China

ICPADS 2025, December

Outline

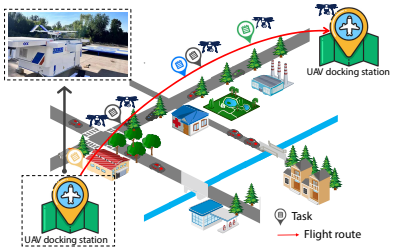
Outline

Problem Background

- Cooperative UAV fleets Scenarios:
  - Infrastructure inspection
  - Logistics and delivery
  - Environmental monitoring
- Fixed endurance $\rightarrow$ Fixed route
- Along the route: A variety of collaborative tasks

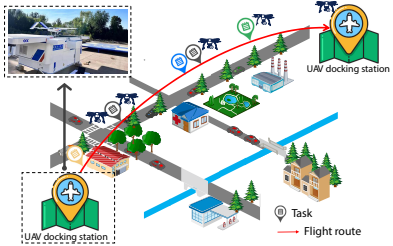
Challenges

- Temporal Dependencies Complexity
  - Tasks must be executed sequentially
  - One delay influences others



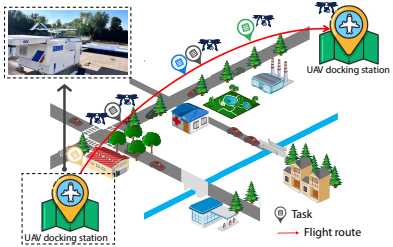
Challenges

- Temporal Dependencies Complexity
  - Tasks must be executed sequentially
  - One delay influences others
- Synchronization Complexity
  - Synchronization conduction
  - Fast UAVs must wait for slower ones → Resource Waste.



Challenges

- Temporal Dependencies Complexity
  - Tasks must be executed sequentially
  - One delay influences others
- Synchronization Complexity
  - Synchronization conduction
  - Fast UAVs must wait for slower ones → Resource Waste.
- Huge Search Space (NP-Hard)
  - Search space grows exponentially
  - Critical: Time is continuous → Infinite possibilities.



Challenges

How to find out the optimal schedule efficiently?

Contributions

- Theoretical Foundation
  - SR-MUCSP Formulation
  - Decoupling: Physical Movement $\leftrightarrow$ Logical Scheduling
- Core Methodology
  - “Skyline” Geometric Representation
  - Algorithms: S-EDP, S-G, S-SDP, S-HDP
  - Pruning and Theoretical Analysis: State Dominance & CPPs
- Simulation
  - Performance: Optimal or Efficient Near-optimal
  - Experiment found: the potential of a state is closely related to its fill rate.

Outline

Problem Formulation

Input: Resources & Tasks

- Fleet: M Homogeneous UAVs
- Tasks: $\mathcal{T} = \{t_1, t_2, ..., t_n\}$
- Tuple: $t_i \rightarrow (d_i, m_i)$

Output: Schedule

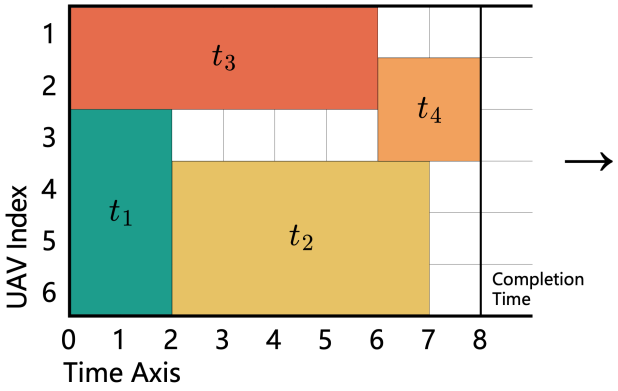
- Output: Schedule P
- Goal: Minimize $\mathbf{C_{max}}(\mathbf{P})$

1. Ordering Constraint
   - UAVs must finish previous task before the next

2. Task Atomicity Constraint
   - All UAVs must start & end simultaneously.

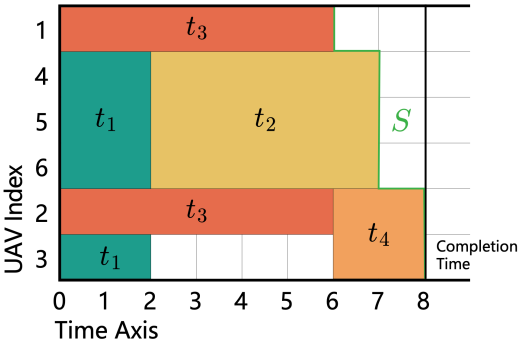
Visualize: 2D Strip

- Y-Axis: Resources (Fixed Capacity)
- X-Axis: Time (Minimize Length)



The Scheduling “Canvas”

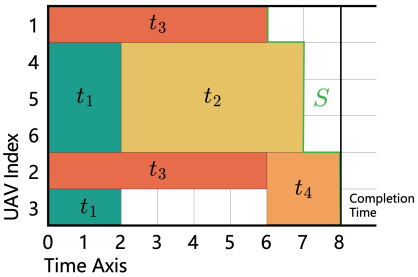
Visualize: Composite Rectangles



- Shape = Load
  - Width: Duration (d_i)
  - Height: UAV Requirement (m_i)
- "Composite"
  - Composed of m_i unit blocks.
  - Key: Can be placed on any m_i available rows.
  - (UAVs are homogeneous!)

Visualize: Review Constraints

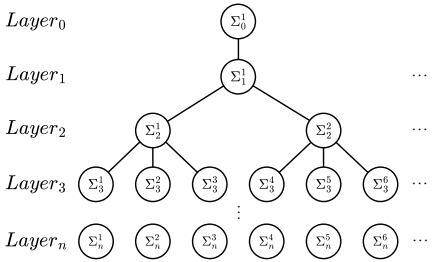
Physical Rules $\Rightarrow$ Geometric Rules



- No Overlap
  $\Leftrightarrow$ A UAV cannot do 2 tasks at once.
- Sequential Placement
  $\Leftrightarrow$ For one UAV, t_i must at the front of t_{i+1}.
- Vertical Alignment
  $\Leftrightarrow$ All m_i parts of the rectangle must start and end together.

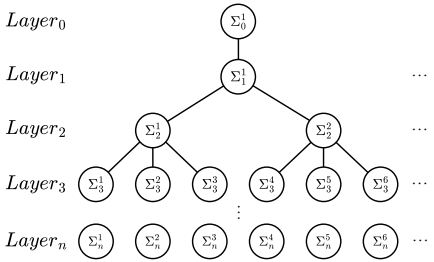
Outline

Transition Graph



- Layered Structure
  Layer $i \Rightarrow$ Scheduling Task t_i
- State Definition (Σ_i)
  Defined by its Skyline (Resource Availability)
- The Crisis: Infinite Branching
  - Continuous Time = Infinite positions
  - Impossible to search exhaustively.

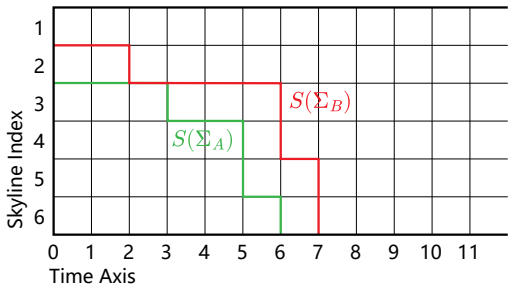
Transition Graph



- Layered Structure
  Layer $i \Rightarrow$ Scheduling Task t_i
- State Definition (Σ_i)
  Defined by its Skyline (Resource Availability)
- The Crisis: Infinite Branching
  - Continuous Time = Infinite positions
  - Impossible to search exhaustively.

Requirement: We need Pruning Strategies.

Skyline Dominance



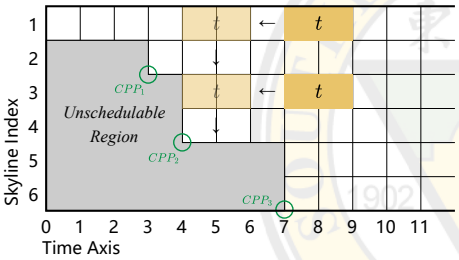
The Skyline Vector

Sorted availability times:
$S(\Sigma) = (s_1, s_2, \ldots, s_M)$

- Dominance Rule:
  If $S_A \leq S_B \Rightarrow$ State A is "better".
- Pruning Strategy:
  Discard State B (It can never beat A).

Candidate Placement Points

- Problem:
  Infinite possible start times on the Time Axis.
- Insight:
  Optimal schedules are always “Left-Aligned” and “Down-Aligned”.
- Solution: CPPs
  Only check placements at “Corners” of the skyline.
  $\infty \rightarrow O(M)$ choices.



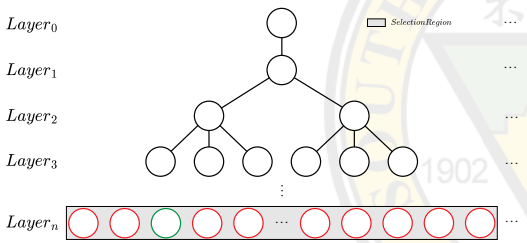
Exact Algorithm: S-EDP

Mechanism: Layer DP

- Transition: $\Sigma_i \xrightarrow{CPP} \Sigma_{i+1}$
  (Generate child states using Corners)
- Discard dominated states.
  (Keep all non-dominated skylines)

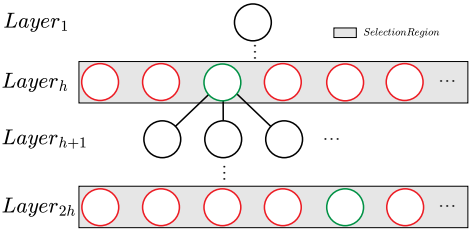
Result

- Selection: In last layer.
- Cost: $O(M^n)$ (Exponential).
  $\Rightarrow$ Only for small scale.



Heuristics: S-SDP & S-G

Idea: Sacrifice global optimality for speed.



- S-SDP (Segmented DP)
  - Divide tasks into segments of size h.
  - Aggressive Pruning: Keep ONLY the best state at boundaries.
- S-G (Greedy)
  - Special Case: $h = 1$.
  - Schedule tasks to the earliest position.
- Selection: In boundary layers

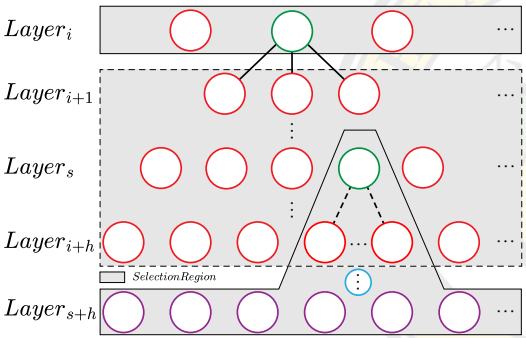
Advanced Heuristic: S-HDP

- Mechanism: Lookahead

  Peek into a window of h future tasks before transitioning.

- Selection by Scoring:
  - $M_{C_{max}}$ (Time Increase)
  - $M_{profile}$ (Profile Flatness)
  - M_{waste} (Waste Rate) $\leftarrow$ Crucial!
- Selection:

  In whole lookahead tree

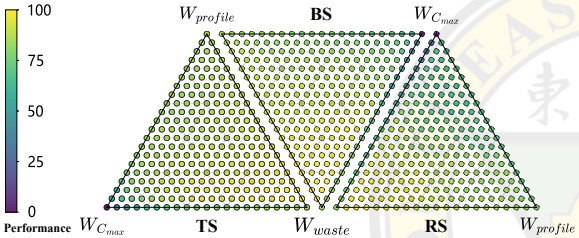


Advanced Heuristic: S-HDP

Scoring Function

$$\text{Score} = \mathbf{W} \cdot \mathbf{M}^T = W_C M_{\Delta C_{max}} + W_W M_{waste} + W_P M_{profile}$$

- $M_{C_{max}}$ (Time)
- $M_{profile}$ (Profile Flatness)
- M_{waste} (Waste Rate)
  ↑ Crucial!

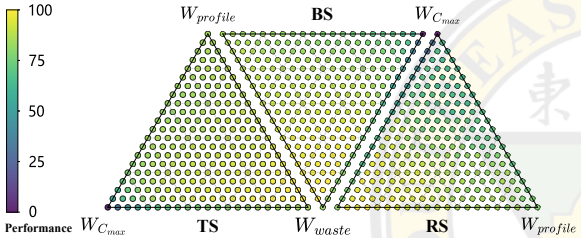


Advanced Heuristic: S-HDP

Scoring Function

$$\text{Score} = \mathbf{W} \cdot \mathbf{M}^T = W_C M_{\Delta C_{max}} + W_W M_{waste} + W_P M_{profile}$$

- $M_{C_{max}}$ (Time)
- $M_{profile}$ (Profile Flatness)
- M_{waste} (Waste Rate)
  ↑ Crucial!



Optimal Configuration

Via Search (Step 0.05):

$$\mathbf{W}^* = (0.10, \mathbf{0.85}, 0.05)$$

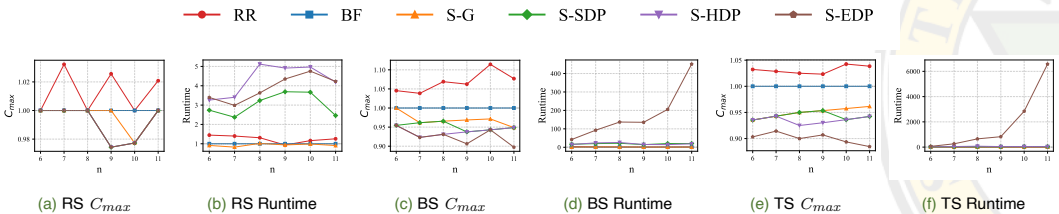
Insight: Minimizing Waste $\gg$ Greedily Minimizing Time.

SOUTHEAST UNIVERSITY

Outline

Small Scale

S-HDP achieves near-optimal results with $<$1% deviation from the ground truth (S-EDP).

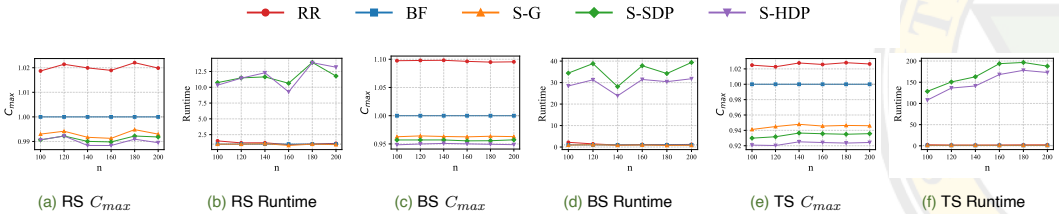


(a) RS C_{max} (b) RS Runtime (c) BS C_{max} (d) BS Runtime (e) TS C_{max} (f) TS Runtime

Fig. 1: Performance comparison in small-scale scenarios ($n \leq 11$). The y-axis represents the average completion time ($C_{\max}$) and runtime, with all values normalized by the results of BF.

Large Scale

S-Algorithms all maintain high performance. S-HDP consistently outperforms baselines, reducing mission time by up to 10% in complex scenarios.



(a) RS C_{max} (b) RS Runtime (c) BS C_{max} (d) BS Runtime (e) TS C_{max} (f) TS Runtime

Fig. 2: Performance comparison in large-scale scenarios ($n \geq 100$). The y-axis represents the average completion time ($C_{\max}$) and runtime, normalized by BF. S-EDP is excluded due to its computational infeasibility at this scale.

Outline

Conclusion

- Problem Formulation
  Defined SR-MUCSP: Single-Route, Cooperative, Sequential.
- Methodological Innovation
  Proposed the Skyline framework to schedule tasks efficiently.
  Developed S-EDP (Optimal) and other heuristics.
- Performance
  Achieved near-optimal solution quality with short runtimes.
- Next Step: Extend framework to Heterogeneous UAVs.

Thank you!

baixinwan@seu.edu.cn