Operator Overload Problem

- Unmanned Aerial Vehicles (UAVs)
  - No on-board pilot
  - Varying degree of remote or autonomous control
  - Trending to smaller, cheaper, w/ greater capability

- One or more operators required to pilot 1 UAV

- How can 1 operator control several UAVs?
Solution: Swarm Intelligence

- Swarm intelligent systems utilize emergence to solve problems
- **Emergence** – whole greater than sum of parts
- Simple local behaviors across many agents lead to complex phenomena
- Inspired by nature
Agent-Based Modeling

- Model system as environment w/ interacting agents
- Bottom-up vs. top-down (equations)
- Captures generative nature of system
- Natural platform for modeling swarms
Dynamic Data-Driven Application System (DDDAS) utilizes two components:

- Incorporate dynamic data into executing application
- Application data steers data-gathering process
- Synergistic feedback control loop

Excellent for modeling non-linear dynamics
Controlling UAV Swarms with DDDAS

Four-Project Survey:
- Flocking Applications
- Communication
- Formation Control
- Mission Scheduling
Flocking Applications
What is Flocking

• BOIDS Reynolds (1987)
  - Separation
  - Cohesion
  - Alignment

• Coordinated Flocking from Local Behavior
• Applied to Particle Swarm Optimization
**Plume Scenario**

- **Contaminant Plume**
  - Highest Concentration at center
  - UAVs flock deployed to measure accident
- **Want diverse measurements**
- **Ground operator controls cohesion**
Plume Scenario

- Ground controller observes standard deviation
- Adjusts cohesion
Vessel Tracking

- 2 UAV Types
  - Searchers
  - Pursuers
- Operator controls:
  - Pursuer to Searcher ratio
  - Number of Pursuers to track each vessel
- 2 Metrics
  - % of vessels detected
  - How well vessels tracked
DDDAS

- Optimization via Simulation
  - Identify swarm metric
    - Coverage
    - Pursuers per Target
  - Calibrate simulation with current data
  - Simulate faster-than-real-time
  - Optimize application
    - UAVs are sensors, so
    - Sensor feedback control through simulation
Icosystem Swarm Game

• Illustrates complex behavior from simple rules

• Each agent non-mutually associated with 1 or 2 other agents

• “Sight” parameter impacts global behavior

- Aggressor: The agent puts its second helper between itself and its first helper.
- Avoider: The agent moves in the opposite direction of its 2nd helper.
- Defender: The agent puts itself between its two helpers.
- Stalker: The agent follows its 1st helper.
Target Search

- Aggressor/Defender behavior implemented for target search
- Augmented with “Leader” that follows pre-assigned way-points
- Objective to maximize coverage, identify unknown targets
**DDDAS Implementation**

- **DDDAS**
  - Inputs “real-world” parameters
  - Execute simulation
  - Outputs results
  - Steers “real-world” application
- **Communicates via webservice**
- **Experiments with intra-swarm communication**
  - Fixed radius vs multi-hop
  - Energy Considerations
Console Screenshot

- Stop server
- Choose system environment
- Set logging level
- Show supervision window
- Generate targets
- Generate UAVs
- Set simulation dimensions
- Set base station
- Take off
- Land
- EMERGENCY

UAV Administrative Console

- INFO 07/24/12 16:27:24: Logging level set to INFO
- INFO 07/24/12 15:27:25: Server started
- INFO 07/24/12 15:27:25: 10 targets generated
- INFO 07/24/12 15:27:25: 10 UAVs generated
- INFO 07/24/12 15:27:29: Server stopped
- INFO 07/24/12 15:27:29: 10 targets generated
- INFO 07/24/12 15:27:29: Server started

UAV Supervision

- Start
- Pause
- Stop
Formation Control
Cooperative Search w/ Swarms

- Developed previous work on cooperative search
- “Dynamic Cleaning” problem
- Two emergent protocols
  - Parallel Paths
  - SWEEP
SWEEP Demo

Capable of “cleaning” diverse topologies
Additional “Sentry” Behavior

- The 2 cooperative search protocols have trade-offs
  - Parallel paths simple
  - SWEEP higher overhead but more applicable

- Add “sentry” to partition
- More efficiently search environment
DDDAS

• Agent-based simulation for ground control
• Highly dynamic environment
  – Topology change
  – UAV failure/reassignment
• Evaluate on different parameters
  – Time/steps
  – Cost/communication
• Implement within DDDAS framework
Mission Scheduling
Scheduling Framework

- A **swarm** consists of many agents
- A **mission** consists of many tasks
- Tasks are associated with a cost
- Some tasks may depend on other tasks
- The problem is to efficiently assign tasks to agents

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>A UAV swarm.</td>
</tr>
<tr>
<td>$v_i$</td>
<td>The ith UAV in the swarm.</td>
</tr>
<tr>
<td>$v_i(t)$</td>
<td>The state of the ith UAV at time t.</td>
</tr>
<tr>
<td>n</td>
<td>Total number of UAVs in the swarm.</td>
</tr>
<tr>
<td>MS</td>
<td>A mission.</td>
</tr>
<tr>
<td>T</td>
<td>The set of tasks in the mission.</td>
</tr>
<tr>
<td>$t_i$</td>
<td>The ith task.</td>
</tr>
<tr>
<td>D</td>
<td>The set of task dependencies in the mission.</td>
</tr>
<tr>
<td>${t_i, t_j}$</td>
<td>A task dependency pair specifying task $t_j$ has a dependency on task $t_i$.</td>
</tr>
<tr>
<td>$MS_t$</td>
<td>The state of a mission at time t.</td>
</tr>
<tr>
<td>C</td>
<td>The set of completed tasks in the mission state.</td>
</tr>
<tr>
<td>P</td>
<td>The set of in-progress tasks in the mission state.</td>
</tr>
<tr>
<td>W</td>
<td>The set of not-started tasks in the mission state.</td>
</tr>
<tr>
<td>M</td>
<td>Scheduling mapping from the set of tasks to the swarm.</td>
</tr>
</tbody>
</table>
Solution Approach

• Global-local hybrid planning and scheduling
  – 1 Swarm Control Agent (SCA)
    • Interface between ground operator and UAVs
    • Assigns new mission tasks to UAVs
  – UAV agent
    • Service running on UAV
    • Communicates with SCA

• SCA attempts to minimize cumulative cost
  – Maintain task queue
  – Assign tasks as they become available

• UAV agent decides order of execution
DDDAS

• Agent-based simulation developed in Java
• Several local assignment algorithms implemented
  – First-come first-serve
  – Travelling Salesman
  – Insertion-based
  – Adaptive
• Simulation executed by ground control upon arrival of new missions

Relative Performance to First-come First-serve
Scheduling GUI
Acknowledgements

Research was conducted under support from the Air Force Office of Scientific Research

Co-authors and Contributors:

• Dr. David Wei, Microsoft
• Rachael Purta, Notre Dame
• Alex Madey, Trinity School
• Dr. Brian Blake, University of Miami
• M. Dobski, A. Jaworski, and S. Nagrecha, ND
Thank you! Questions?