APIS: Honeybee Foraging Task Assignment for Use in Uncertain and Unreliable Environments

John Pye and Dr. Natalia Alexandrov
NASA Langley Research Center

January 8th, 2024

Presented to: AIAA SciTech Session HMT-02

"Managing Uncertainties in Cyber-Physical Human-Machine-Making"

Overview

Successful Cyber-Physical-Human (CPH) teaming relies on robust autonomy.

- Autonomous actions need to be predictable
- Uncertainty needs to be understood and managed

Present the reference problem of agents servicing objectives in an uncertain environment.

- Common problem formulation to every domain
- Can be simplified for ease of analysis

Walk through our proposed Autonomous Persistent Intelligent Swarm (APIS) approach to robust collaborative autonomy.

- Demonstrate performance under a variety of conditions and parameters
- Compare APIS performance to a baseline collaborative approach

Definitions

Agent: A system operating off the common tasking algorithm (in context, either APIS or Auction)

Objective: A location where an agent may complete a task

Priority: A measure of the urgency for an agent to complete an objective's task

Bid: A measure of the fitness of a particular agent to complete a particular task

Algorithm Effectiveness: Inverse of the mean priority of all objectives in the area

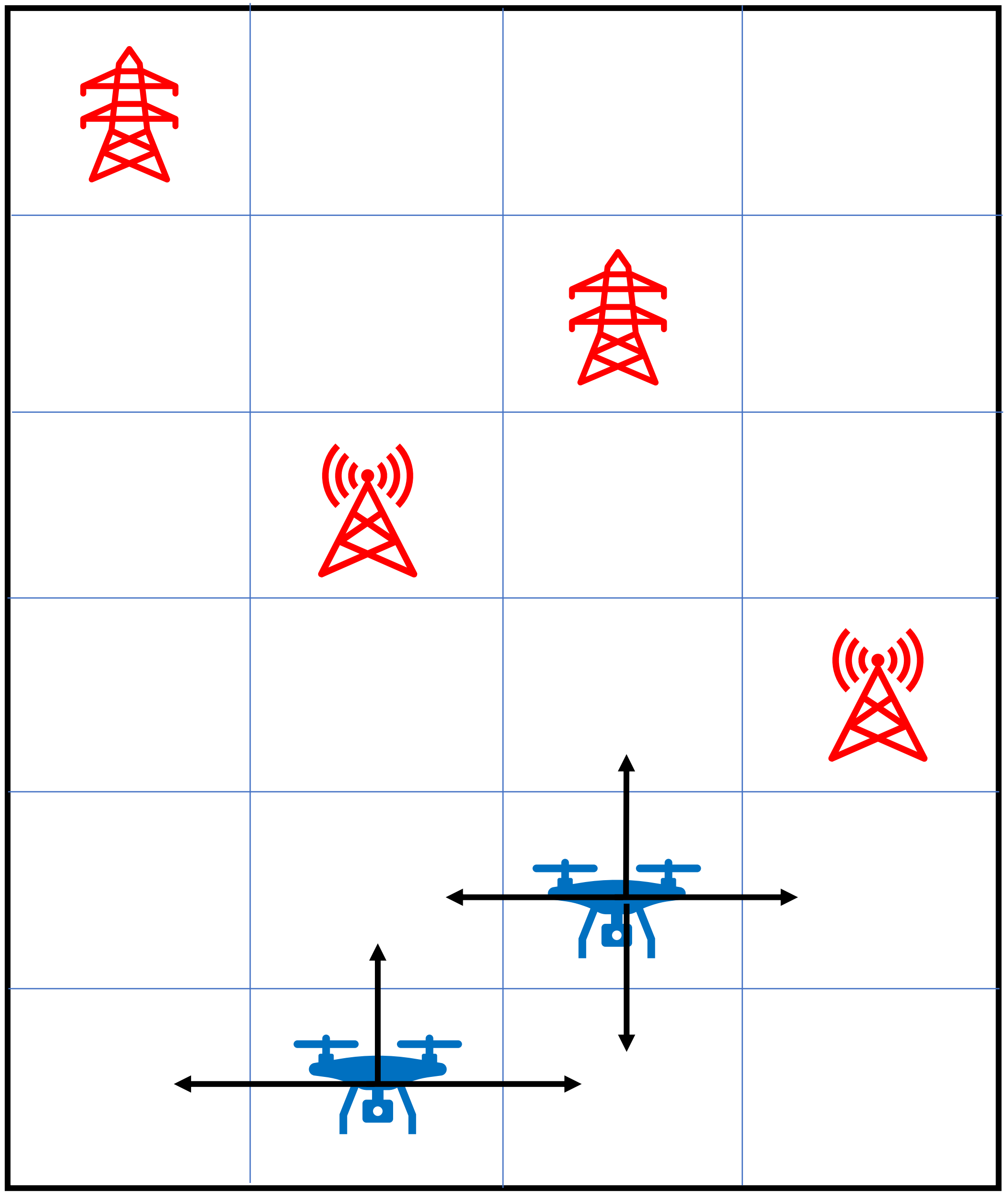
Algorithm Robustness: Inverse of the standard deviation of each trial's mean priority

Simplified Design Reference Mission

Task: Aerial agents inspect degrading platforms in a grid world.

Known: Initial platform location, expected initial platform priority, initial agent locations, possible agent movement.

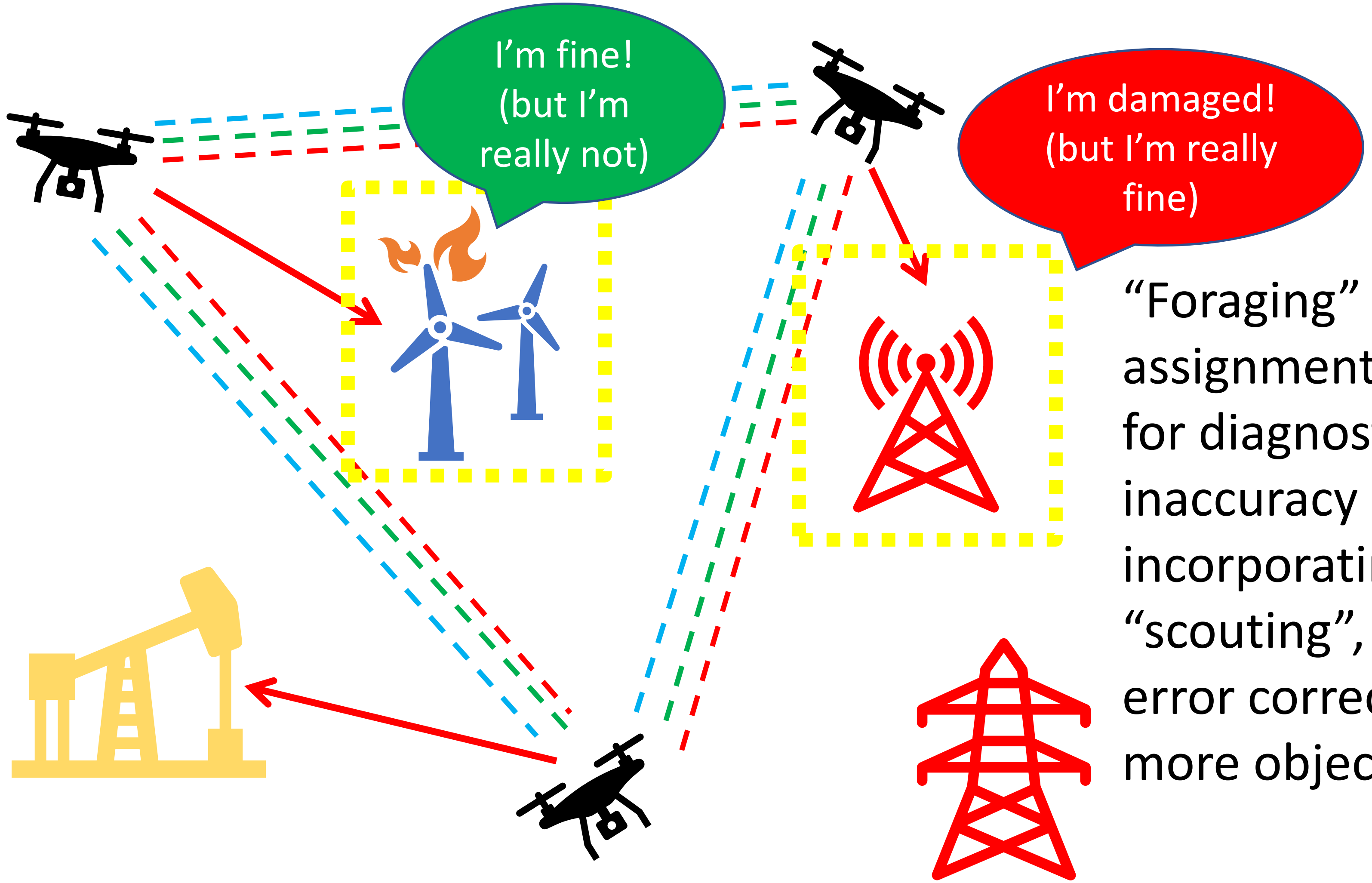
Unknown: True platform priority, ~~servicing time,~~ and agent decisions outside of comms range.



Objective priorities based on self-diagnostics can be misleading. CPH teams need to account for uncertainty in the environment.

Agent inspection provides error correction to self-diagnostics.

Agents may be assigned to tasks based on priority (“harvesting”) or randomly (“scouting”).



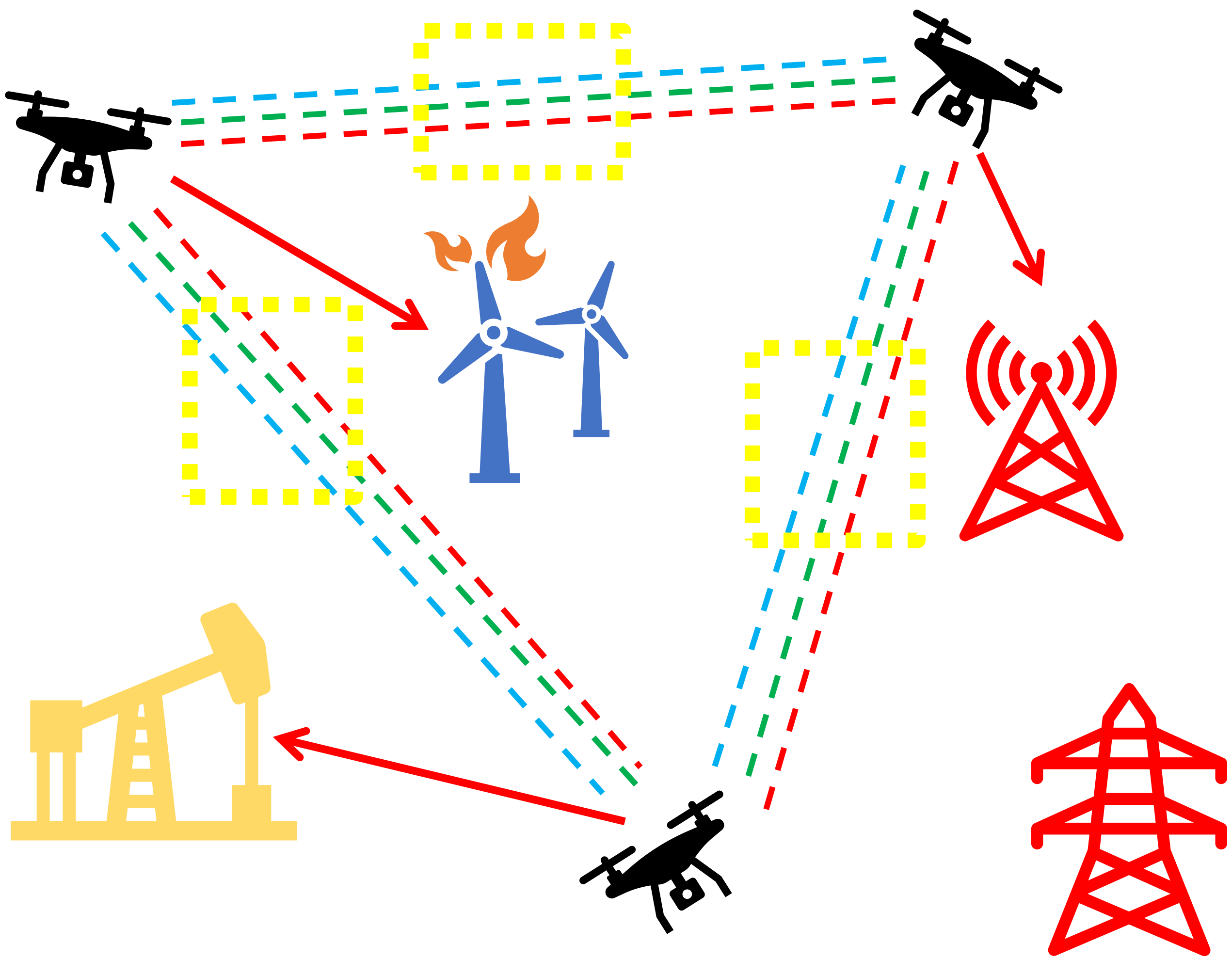
“Foraging” task assignment accounts for diagnostic inaccuracy by incorporating “scouting”, to collect error correction on more objectives.

“Scouting” agents collect information on objectives which may not be serviced through priority-based “harvesting”.

Objective priorities based on self-diagnostics can be misleading. CPH teams need to account for uncertainty in the environment.

Agents communicate information on objectives and other agents through multiple modes.

"Dancing" agents rendezvous at set points to guarantee communication at short range.



Communication improves efficiency, but agents can function independently.

Each agent maintains a memory model of objectives and other agents, allowing for inferences of the behavior of other agents.

"Dancing" agents guarantee a baseline level of communication.

Bee-Inspired Aspects of Swarm Operations

Diverse Comm. Techniques (“dancing”)

Short-range (e.g., visual)

Long-range (e.g., radio)

Foraging Behaviors (“scouting”, “harvesting”)

“Harvesters” greedily complete tasks

“Scouts” randomly explore tasks

Memory Models (common to all roles)

Agents construct models of targets and agents

Models corrected with other agents

Robust-to-Uncertainty Autonomy

Auction and APIS Pseudocode

Initialization

A: Set of all agents in mission area
T: Set of all targets in mission area
B: Bounds of the mission area
u: Baseline urgency of all targets in mission area
σ: Standard deviation from baseline urgency of all targets in mission area
t: Timespan of mission
t_{reset}: Time before target priority reset

Initialize:

$t_i = 0$

For each target j in **T**:
  $x_j \leftarrow \mathrm{U}(\mathbf{B})$
  $p_j \leftarrow u$
For each agent i in **A**:
  $x_i \leftarrow \mathrm{U}(\mathbf{B})$

Distribute objectives and agents uniformly in arena

For each agent i in **A**:
  $i.agentModels \leftarrow \mathbf{A}$
  $i.targetModels \leftarrow \mathbf{T}$

For each target j in **T**
  $\Delta_j \leftarrow \mathrm{N}(0, \sigma)$
  $p_j^t \leftarrow \Delta_j + u$

Generate random normal unknown bias and apply to ideal priority to yield true priority

Operational Loop (Standard Auction)

Do until $t_i = t$

$\quad$ For each agent i in $\mathbf{A}$:

$\quad\quad$ If $\text{Bid}(i, i.target) = 0$:

$\quad\quad\quad$ $i.target \leftarrow None$

Commitment subroutine (only abandon if 0 bid)

$\quad$ For each agent i in $\mathbf{A}$:

$\quad\quad$ If $i.target = None$:

$\quad\quad\quad$ For each target j in $\mathbf{T}$:

$\quad\quad\quad\quad$ $i.bids[j] \leftarrow \text{Bid}(i, j)$

$\quad\quad\quad$ For each other agent i_{other} in $\mathbf{A}$:

$\quad\quad\quad\quad$ If $\text{InCommsRange}(i, i_{other})$:

$\quad\quad\quad\quad\quad$ $i.bids[i_{other}.target] \leftarrow None$

$\quad\quad\quad$ $i.target \leftarrow \max(i.bids)$

Cooperative subroutine (only service if currently unassigned)

$\quad$ EvolveTargets($\mathbf{T}, \mathbf{A}$)

$\quad$ For each agent i in $\mathbf{A}$:

$\quad\quad$ $x_i^{+} \leftarrow ConvergeToTarget(x_i, i.target)$

End Do

Matarić, M.J., Sukhatme, G.S. & Østergaard, E.H,

"Multi-Robot Task Allocation in Uncertain Environments," in Autonomous Robots, vol. 14, pp. 255–263, 2003,

https://doi.org/10.1023/A:1022291921717.

Bidding Function

Function 1: Bidding function Bid()

Agent i : argument 1

Target j : argument 2

$B_i^j \leftarrow P_i^j - d(x_i, x_j)$

return B_i^j

End Function

Base bid on "efficiency": benefit (dictated by priority) and cost (dictated by distance)

Matarić, M.J., Sukhatme, G.S. & Østergaard, E.H,

"Multi-Robot Task Allocation in Uncertain Environments," in Autonomous Robots, vol. 14, pp. 255–263, 2003,

https://doi.org/10.1023/A:1022291921717.

Operational Loop (APIS)

Do until $t_i = t$

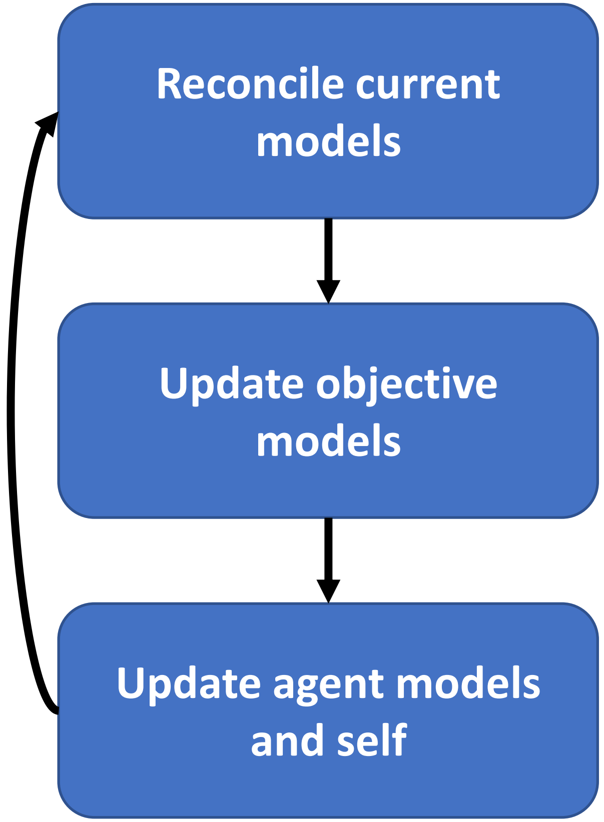
    For each agent i in $\mathbf{A}$:
        For each other agent i_{other} in $\mathbf{A}$:
            ReconcileModels(i, i_{other})

Deconflict memory models with other agents

    EvolveTargetDiagnostics($\mathbf{T}, \mathbf{A}$)
    For each agent i in $\mathbf{A}$:
        For each target j in $\mathbf{T}$:
            UpdateTargetModel(i, j)
        For each agent model i_{model} in $\{i, i.agentModels\}$:
            UpdateAgentModel(i, i_{model}, t_i)
    EvolveTargetReset($\mathbf{T}$)
    For each agent i in $\mathbf{A}$:
        EvolveTargetReset($i.targetModels$)

Update memory models and self, simultaneously

End Do



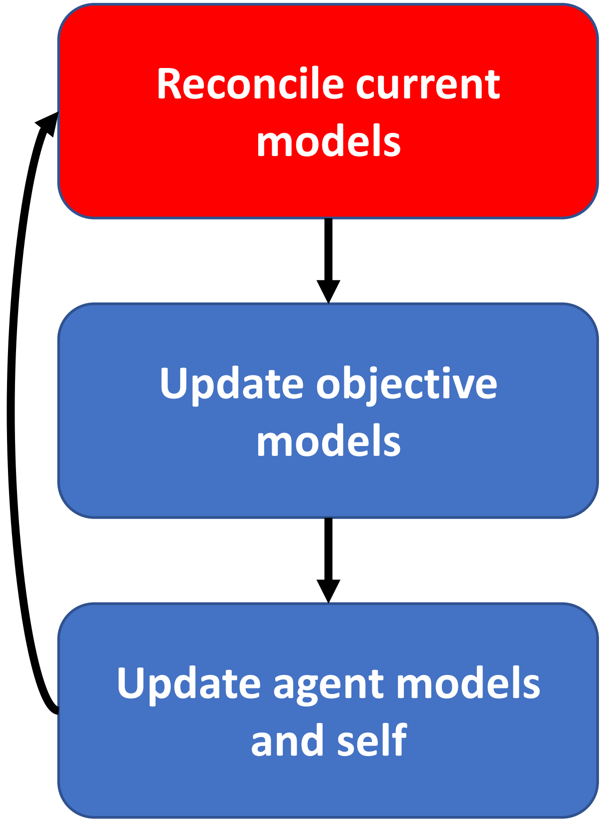
Model Reconciliation Function

Function 6: Reconcile sets of models function ReconcileModels()

Agent i : Argument 1
Other agent i_{other}: Argument 2
If $|x_i - x_{other}| > communicationRange$: Return
For each agent model m in $i.agentModels$:
If $m.id = i_{other}.id$:
$m_i \leftarrow i_{other}$
Else:
If $m_i.age > m_{other}.age$:
$m_i \leftarrow m_{other}$
For each target model m in $i.targetModels$:
If $\Delta_{m_{other}} \neq 0$:
$\Delta_m \leftarrow \Delta_{m_{other}}$
If $m_i.age > m_{other}.age$:
$m_i \leftarrow m_{other}$
End Function

Prioritize firsthand models (models of the other agent), and more recent models

Prioritize disseminating target priority bias factor – independent of model age



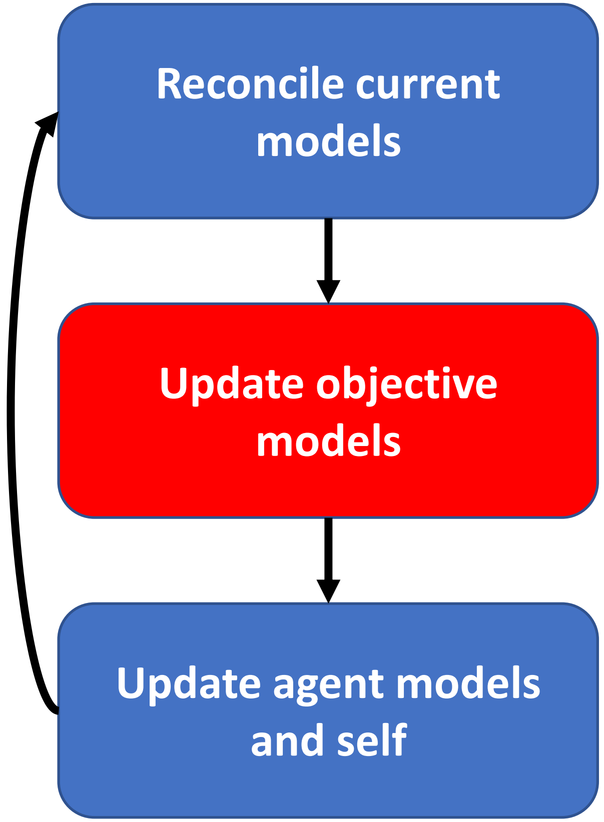
Target Model Update Function

Function 7: Evolve models of targets function UpdateTargetModel()

Agent i : argument 1
Target j : argument 2
If $|x_i - x_j| > communicationRange$:
    $i.m_j.age \leftarrow i.m_j.age + 1$
    EvolveTargets($i.targetModels, i.agentModels$)
Else:
    $i.m_j.age \leftarrow 0$
    $i.m_j.resetTime \leftarrow j.resetTime$
    $P_{i.m_j} \leftarrow P_j.$
    If $|x_i - x_j| = 0$:
        $i.m_j.\Delta_j \leftarrow \Delta_j$
End Function

Increase model age if out of update range

Otherwise, communicate with target to receive firsthand diagnostics (apply bias correction if possible)



Agent Update Function (abridged)

If $i_{model}.currentJob = Harvest$ OR $Scout$:
  $jobDone \leftarrow (P^t_{i_{model}.target} = 0)$
Else If $i_{model}.currentJob = Dance$:
  If $i_{model}.danceTime = goalDanceTime$:
    $jobDone \leftarrow True$
    $i_{model}.danceTime \leftarrow 0$
  Else:
    $i_{model}.danceTime \leftarrow i_{model}.danceTime + 1$

Check completion conditions for each job type (harvest, scout, dance)

If $jobDone$:

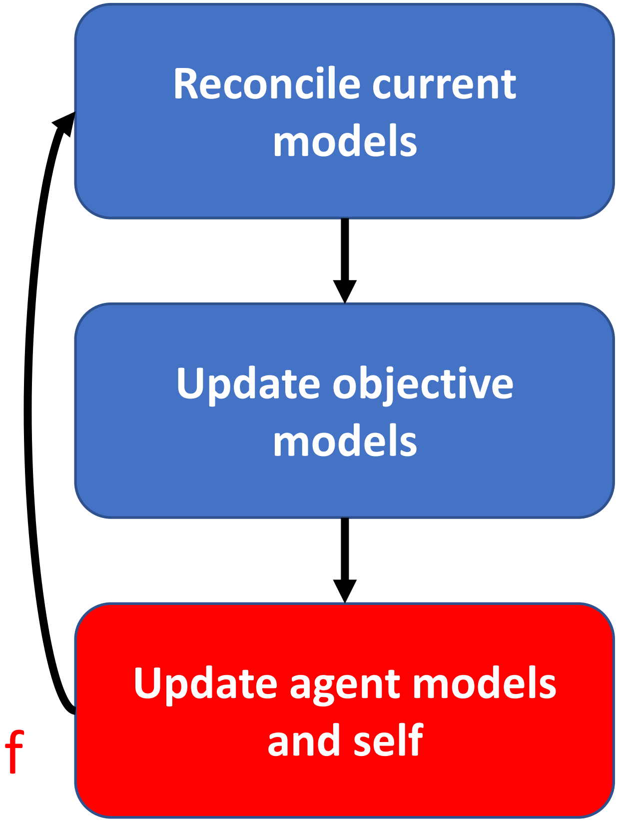
Randomly choose new job based on APIS parameters

$i_{model}.currentJob \leftarrow \text{choose}(\{Harvest, Scout, Dance\})$
If $i_{model}.currentJob = Scout$:
  $i_{model}.target \leftarrow \text{choose}(i.freeTargetModels)$

Randomly pick open objective if job is scout

Else If $i_{model}.currentJob = Dance$:
  $i_{model}.target \leftarrow danceRallyPoint$
  $i_{model}.danceTime \leftarrow 0$

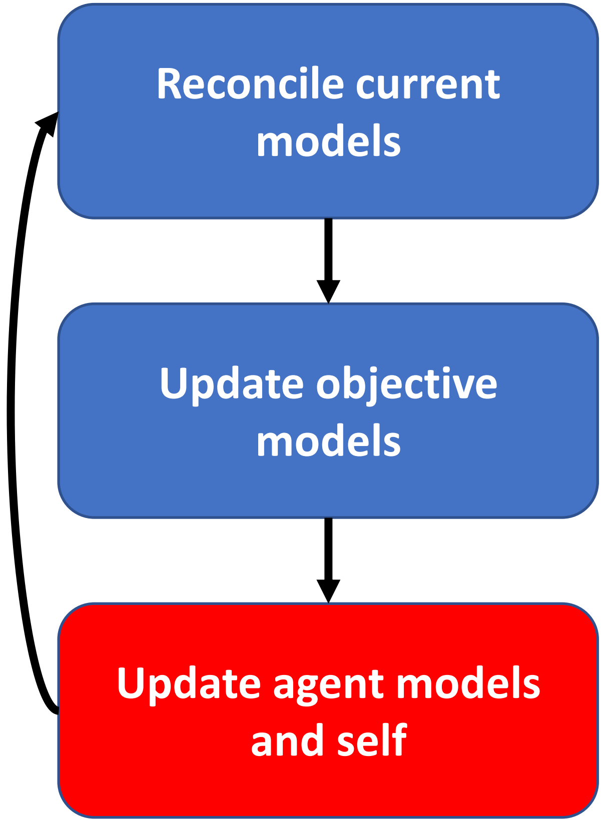
Converge to preset rally point if job is dance



Agent Update Function: Perform Job

If $i_{model}.currentJob = Harvest$:
    $switchTarget \leftarrow False$
    If $i_{model}.target = None$:
        $switchTarget \leftarrow True$
    Else:
        If $(i_{model}.target \in \{i.otherAgentModels.target\})$ OR $(P^t_{i_{model}.target} = 0)$:
            $switchTarget \leftarrow True$
    If $switchTarget$:
        $i_{model}.target \leftarrow None$
        For each model m in $i.freeTargetModels$:
            $i_{model}.bids[m] = \mathrm{Bid}(i_{model}, m)$
        $i_{model}.target = \max(i_{model}.bids)$

Always reconsider chosen objective if job is harvest: pick best unoccupied



APIS and Auction Test Results

Baseline Comparison Test Results (mean total priority)

Algorithm effectiveness: Measured as mean priority of all objectives across all trials (lower number indicates more effective tasking)

Standard APIS settings of 80% chance for harvest, 10% chance for scout, 10% chance for dance

Trial	CC	CN	NC	NN	APIS
$\sigma = 0$, 2 agents, 4 targets	1059.667	1477.988	2451.295	3545.324	520.066
$\sigma = 4$, 2 agents, 4 targets	1133.942	1646.819	2710.702	3884.058	550.199
$\sigma = 10$, 2 agents, 4 targets	1089.404	1530.468	2553.710	3701.541	535.719
$\sigma = 0$, 10 agents, 20 targets	6253.692	6972.121	11768.047	19732.141	3448.477
$\sigma = 4$, 10 agents, 20 targets	6214.143	6948.015	11719.990	19528.225	3471.483
$\sigma = 10$, 10 agents, 20 targets	6322.831	7059.089	11833.616	19724.153	3373.418

Best standard auction performance: fully cooperative and committed

Best overall performance: ~50% reduction in test priority mean

Conclusion: APIS more efficient than standard auction task assignment.

Baseline Comparison Test Results (mean trial priority standard deviation)

Algorithm reliability: Measured as standard deviation of mean priority of all objectives across all trials (lower number indicates less variance from random trials)

Standard APIS settings of 80% chance for harvest, 10% chance for scout, 10% chance for dance

Trial	**CC**	**CN**	**NC**	**NN**	**APIS**
$\sigma = 0$, 2 agents, 4 targets	487.595	678.301	851.527	991.106	218.478
$\sigma = 4$, 2 agents, 4 targets	557.374	774.681	952.250	1101.224	248.056
$\sigma = 10$, 2 agents, 4 targets	565.056	809.849	1048.445	1193.280	234.622
$\sigma = 0$, 10 agents, 20 targets	1717.756	1735.272	1842.847	1977.284	859.093
$\sigma = 4$, 10 agents, 20 targets	1904.007	1861.644	1872.522	1937.558	874.616
$\sigma = 10$, 10 agents, 20 targets	2000.557	1956.513	1971.477	2063.461	933.994

Best standard auction performance: fully cooperative and committed

Best overall performance: ~50% reduction in test standard deviation

Conclusion: APIS more reliable than standard auction task assignment.

APIS Parameter Tuning Test Results (mean total priority)

Where H = Harvest % Chance, S = Scout % Chance, and D = Dance % Chance

Trial	80H, 10S, 10D	40H, 30S, 30D	100H	100S	80H, 20S	80H, 20D
$\sigma = 4$, 10 agents, 20 targets	3471.483	3730.266	3126.123	3474.689	3353.603	3176.460
$\sigma = 10$, 10 agents, 20 targets	3373.418	3769.991	3046.632	3430.168	3451.245	3251.597

Balanced parameters not optimal effectiveness; indicates incomplete integration between jobs.

100% harvest agents most effective

Elimination of scouting role improves effectiveness over balanced case

Key takeaways:

1. Dancing agents increase effectiveness.
2. Too many dancing/scouting agents can reduce effectiveness.
3. Scouting agents have minimal impact on effectiveness.

Conclusion: APIS efficiency stems primarily from proportion of harvesting agents.

APIS Parameter Tuning Test Results (mean trial priority standard deviation)

Where H = Harvest % Chance, S = Scout % Chance, and D = Dance % Chance

Trial	80H, 10S, 10D	40H, 30S, 30D	100H	100S	80H, 20S	80H, 20D
$\sigma = 4$, 10 agents, 20 targets	874.616	597.441	1069.471	435.743	994.257	848.346
$\sigma = 10$, 10 agents, 20 targets	933.994	1060.595	1086.235	411.931	924.536	859.197

Key takeaways:

1. Dancing agents significantly increase robustness by decreasing trial performance variation.
2. Too many dancing/scouting agents can reduce robustness.

100% harvest agents extremely unreliable

Highest reliability comes from pure scouting; random objective task selection masks platform uncertainty

Balanced parameters show good reliability

Conclusion: APIS reliability stems primarily from proportion of scouting and dancing agents.

Closing Remarks

Conclusions

APIS demonstrates promising solution to swarm coordination in uncertain environments.

- More efficient (lower mean objective priority)
- More reliable (lower trial performance standard deviation)
- Advantage in efficiency and reliability contingent on uncertain environment

APIS behavior can be meaningfully altered by varying parameters.

- Higher harvesting chance: more efficient in ideal environment
- Higher scout/dance chance: more reliable in uncertain environment
- **Key takeaway: increased communication alleviates impact of uncertain conditions**

Abnormally high-performance scout-only APIS variation indicates further role integration required.

Future Work

Increase design reference mission fidelity – eliminate "simplified" traits.

- Continuous state space
- More realistic communication modeling
- More sources of uncertainty in objective

Compare APIS to more specialized state-of-the-art task assignment strategies.

- Apply state-of-the-art solutions to design reference mission
- Simulate against APIS to identify strengths and weaknesses

Test APIS applicability in both air and space domain missions to test algorithm applicability and identify more use cases.

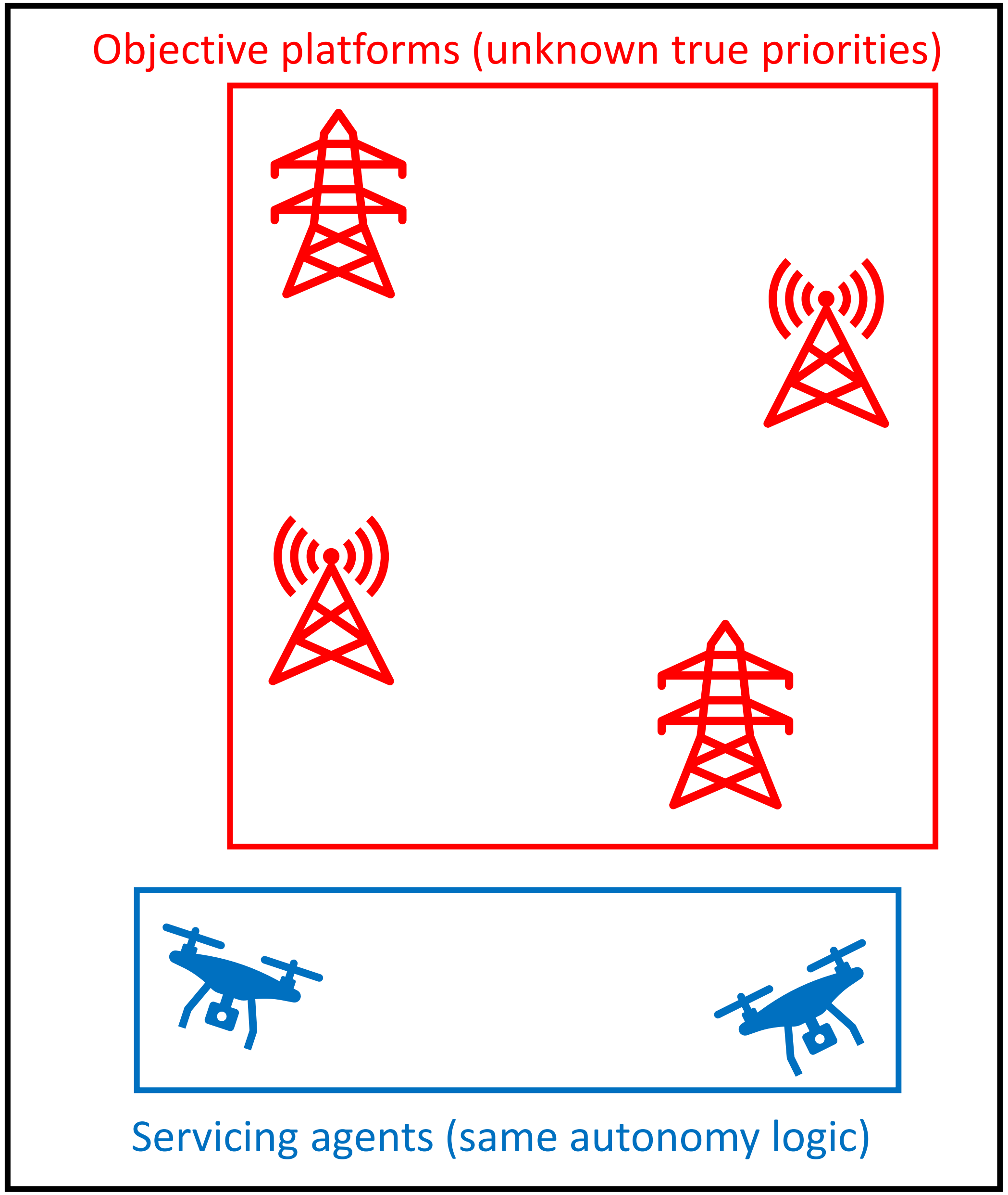
Backup materials

Design Reference Mission

Task: Aerial agents inspecting degrading platforms.

Known: Initial platform location, expected initial platform priority, initial agent locations.

Unknown: True platform priority, servicing time, and agent decisions outside of comms range.



Operational Uncertainty

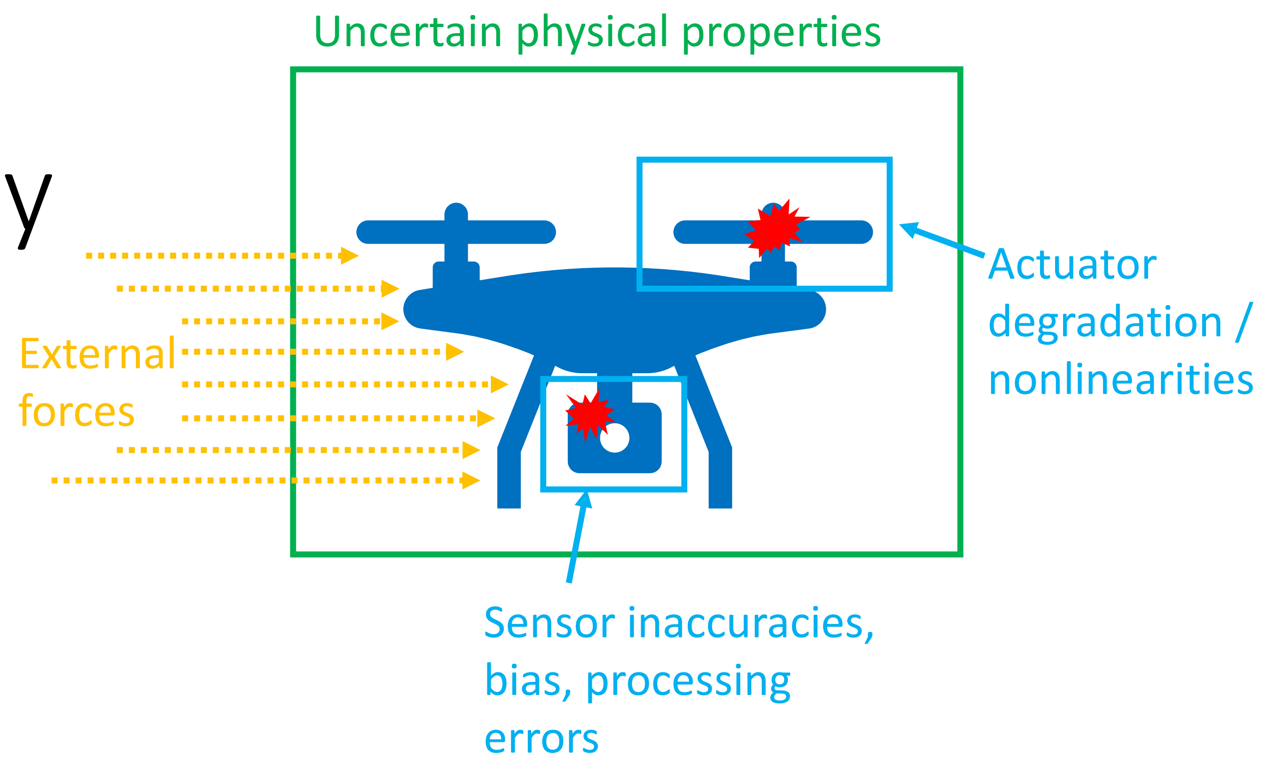
Uncertainty in Cyber-Physical-Human teams can reduce trust, and harm effectiveness.

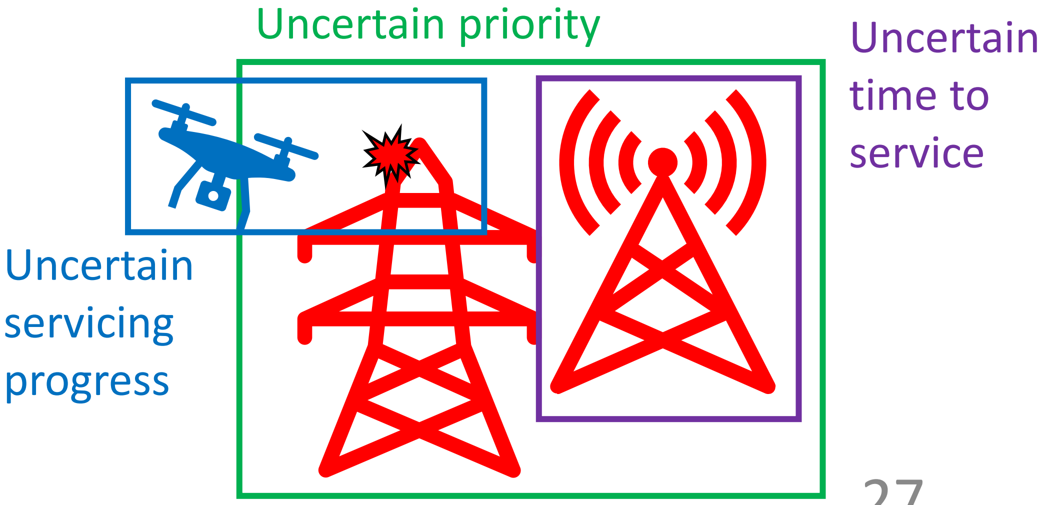
Sources of agent uncertainty:

- Physical breakdown
- Informational errors
- Communication degradation

Sources of task uncertainty:

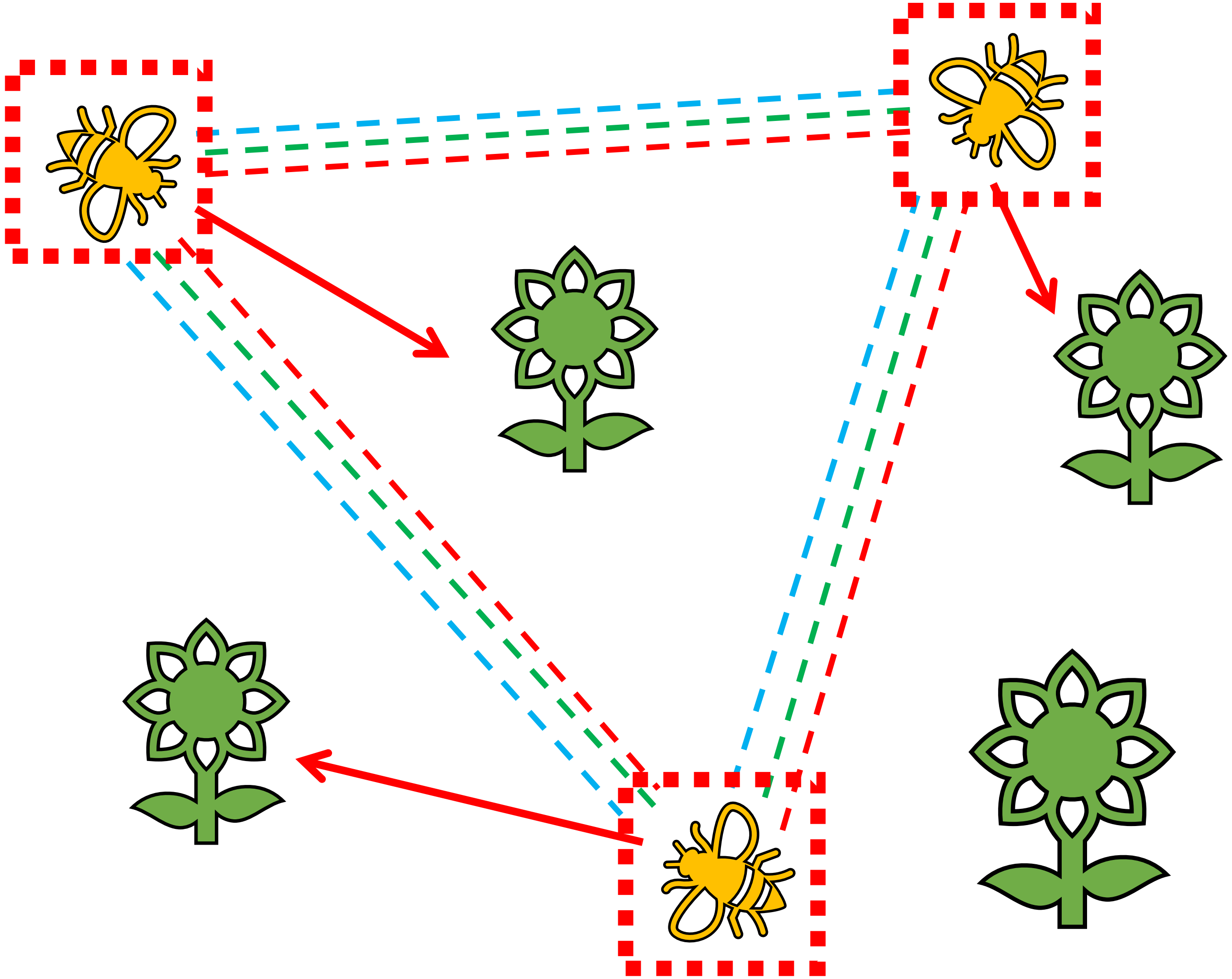
- Diagnostic uncertainty
- Servicing uncertainty





Bee foraging behavior is well-suited for coordinating swarms in environments with <u>uncertain communication</u> and <u>unreliable information</u>.

Bees share information about target position, quality, and assignment with local agents. Information then propagates.



Bees may choose to become "scouts" and ignore target fitness. These scouts randomly explore objectives, allowing for target verification.

Bee foraging discovers inaccurate objective information and disseminates it to other agents.

Convergence Function

Converge to target along grid world

Simplify simulation dynamics by ignoring collision avoidance (assume large grid cells)

Function 2: State evolution function ConvergeToTarget()

Agent state x_i : argument 1

Target state x_{target} : argument 2

$\Delta = x_i - x_{target}$

If $\Delta.\mathrm{x} = \Delta.\mathrm{y}$ and $\Delta \neq 0$:

  If $\Delta.\mathrm{y} > 0$:

    $x_i^+.y \leftarrow x_i.y - 1$

  Else:

    $x_i^+.y \leftarrow x_i.y + 1$

Else If $\Delta.\mathrm{x} > \Delta.\mathrm{y}$:

  If $\Delta.\mathrm{x} > 0$:

    $x_i^+.x \leftarrow x_i.x - 1$

  Else:

    $x_i^+.x \leftarrow x_i.x + 1$

Else If $\Delta.\mathrm{x} < \Delta.\mathrm{y}$:

  If $\Delta.\mathrm{y} > 0$:

    $x_i^+.y \leftarrow x_i.y - 1$

  Else:

    $x_i^+.y \leftarrow x_i.y + 1$

$x_i \leftarrow x_i^+$

End Function

Objective Evolution Function

Assume servicing progresses in predictable fashion for simplicity

Assume degradation also progresses in predictable fashion for simplicity

Function 3: Priority evolution function EvolveTargets()

Set of all targets **T**: argument 1
Set of all agents **A**: argument 2
For each target j in **T**:
$oldPriority \leftarrow P_j^t$
$beingServiced \leftarrow False$
For each agent i in **A**:
If $x_i - x_j = \mathbf{0}$:
$beingServiced \leftarrow True$
If $beingServiced$**:**
$P_j{}^+ \leftarrow \max(P_j - 1, 0)$
$P_j^{t+} \leftarrow \max(P_j^t - 1, 0)$
Else:
$P_j{}^+ \leftarrow P_j + 1$
$P_j^{t+} \leftarrow P_j^t + 1$
If $oldPriority = P_j^{t+}$:
$j.resetTime \leftarrow j.resetTime + 1$
If $j.resetTime = t_{reset}$:
$P_j{}^+ \leftarrow u$
$P_j^{t+} \leftarrow \Delta_j + u$
End Function

Objective Diagnostic Function

Function 4: Priority evolution without reset function EvolveTargetDiagnostics()

Set of all targets **T**: argument 1
Set of all agents **A**: argument 2
For each target j in **T**:
  $oldPriority \leftarrow P_j^t$
  $beingServiced \leftarrow False$
  For each agent i in **A**:
    If $x_i - x_j = \mathbf{0}$:
      $beingServiced \leftarrow True$
  If $beingServiced$:
    ${P_j}^+ \leftarrow \max(P_j - 1, 0)$
    ${P_j^t}^+ \leftarrow \max(P_j^t - 1, 0)$
  Else:
    ${P_j}^+ \leftarrow P_j + 1$
    ${P_j^t}^+ \leftarrow P_j^t + 1$
End Function

Assume servicing progresses in predictable fashion for simplicity

Assume degradation also progresses in predictable fashion for simplicity