SAND2022-13365C

Applications of Autonomous Sensor Task Planner for Intelligence, Surveillance and Reconnaissance

Sarah Johnson, Kem Cespedes, Rachel Schlossman, John Richards

LDC: NONE – Publicly Releasable After Review (Formerly DISTRO A)

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525.

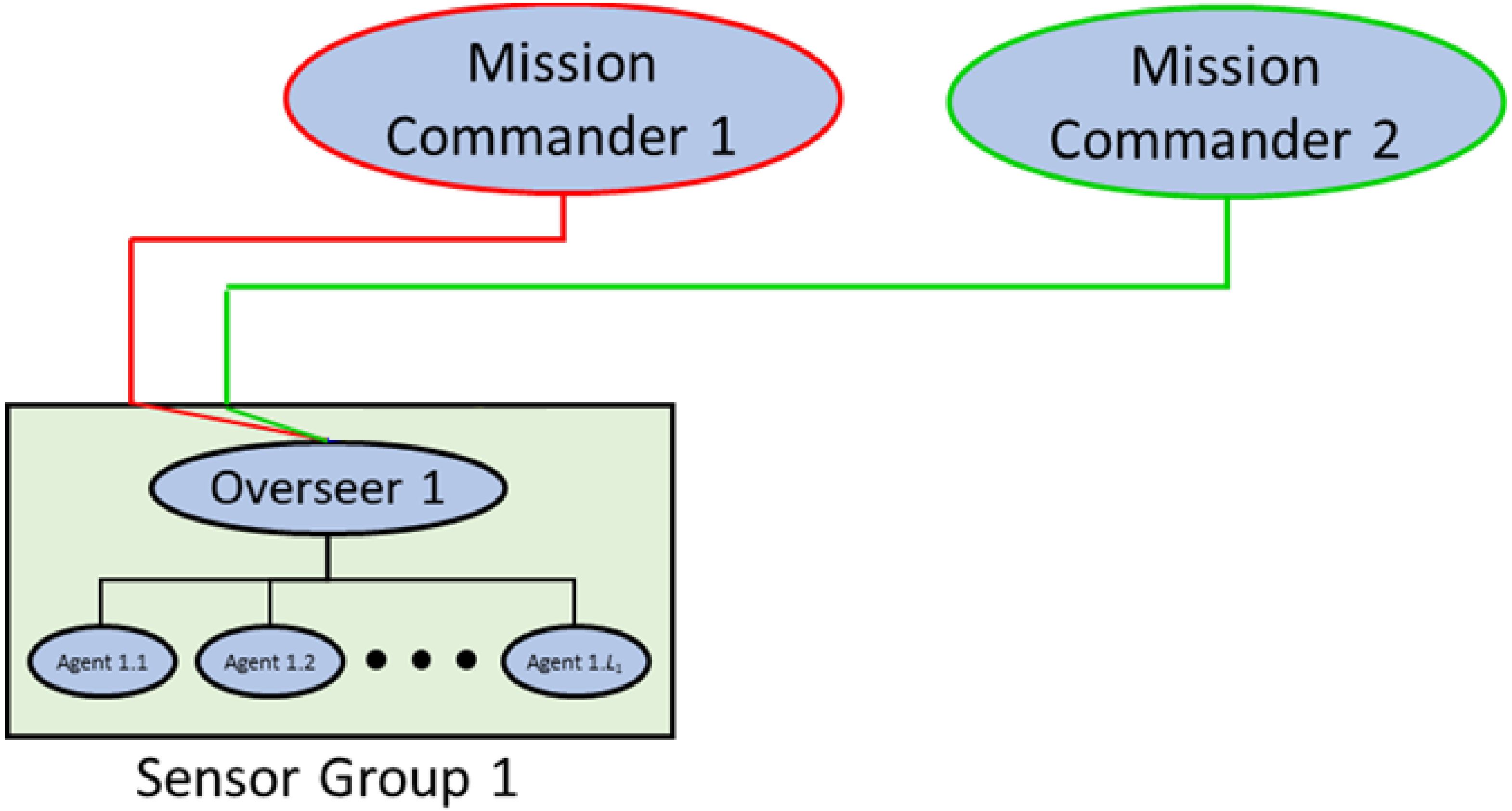
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Program Overview

- Increasing demand for sensing resources in various warfighting domains and ISR missions
- Allocation of sensor tasks may be impossible for human operator to deconflict and prioritize in timely manner
- This work extends Sandia's legacy autonomous sensor scheduling algorithm [1][2] formulated with mixed-integer linear programming (MILP) [3][4] by:
    - Leveraging realistic simulation data
    - Incorporating operational constraints (i.e. sensor availability, access, and confidence)
    - Implementing a waypoint generation algorithm to discretize large search areas of interest
- This work results from an ongoing collaboration between Sandia National Labs and the Naval Postgraduate School

Sensor Scheduling Hierarchy

- Mission commanders (of differing ranks) send task requests to overseers
- Overseers are responsible for their group of sensing agents
- Overseers balances the load of incoming requests via an optimization problem



Methodology

Input Generation

- Mission commanders are responsible for providing task requests
- Overseers have knowledge of their respective sensor groups

Input 1: Task/Tip requests
Look at this search area in the coming time window (Tips, Locations, Collection Values)

Input 2: States of available sensors
List of available sensors including sensor type, average availability, range of sight, footprint size, and position

Mixed-Integer Linear Program

Output: Sensor Employment Plan / Actual Tasks to Execute
This schedule includes sensor ID, sensor footprint location, range bin classification, and specific times of execution in the coming time window

- All inputs are used to devise sensor-waypoint utility values. Each utility value is associated with a specific sensor looking at an individual point.
- Optimization is performed to maximize overall utility

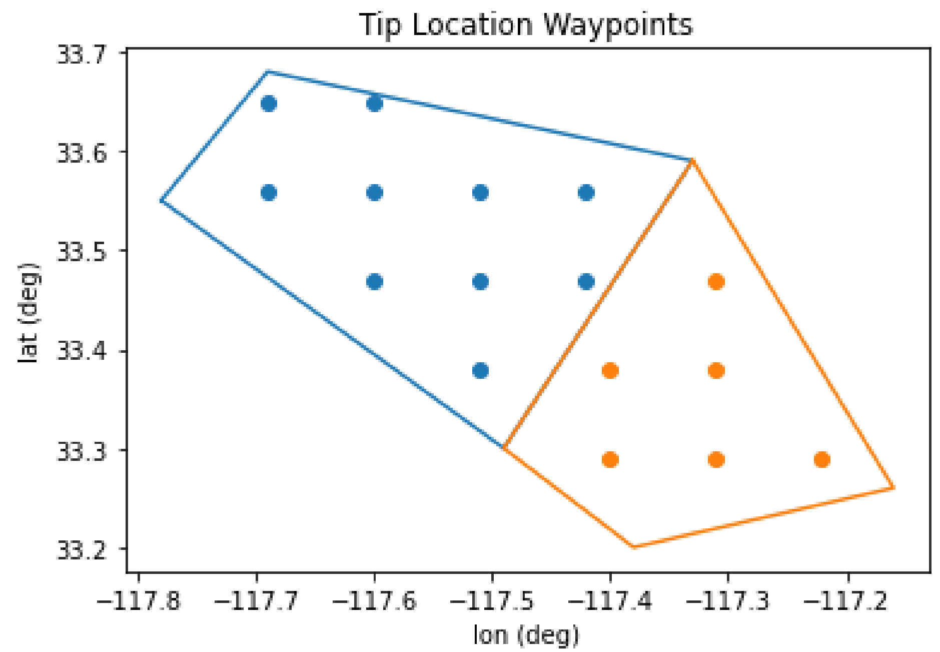
Mission Commander Input

- Tips: Requested activity/entity to be scanned
- Locations: Search areas (polygons) where the desired tips are predicted to be found
- Collections: Potential sensing combinations to achieve a mission commander's desired tip (desired information outcome)

Tip ID	Collection ID	Sensor Type	Number of Timesteps	Collection Value
1	1	Electro-Optical	0	0.21
		SAR	1	
	2	Electro-Optical	1	0.14
		SAR	0	
	3	Electro-Optical	1	0.31
		SAR	1	
2	4	Electro-Optical	0	0.29
		SAR	1	
	5	Electro-Optical	1	0.20
		SAR	0	
	6	Electro-Optical	1	0.44
		SAR	1	

Polygon Waypoint Generation

- Number of waypoints per search area (polygon) depends on polygon size
- Waypoint layout determined by pre-defined sensor footprint sizes



Break down larger search areas
into task-able waypoints

Access, Feasibility, and Utility Computation

- Feasibility:
  - Definition: it is feasible for sensor *i* to scan for waypoint *w* in polygon *p* in the next time horizon
  - Feasible if waypoint *w* is currently within sensor *i's* maximum range
    $\rightarrow f_i^w = -1 \; or \; 1$
- Access:
  - $\mathrm{Confidence_p}$ = confidence of tip in polygon *p*
  - $\mathrm{Access_i^w} = (f_i^w \times \mathrm{Confidence_p})$
- Utility:
  - Definition: the quantified benefit of a specific sensor scanning a specific waypoint [1][2]
  - $u_{i,w,n}^l$ = utility of sensor *i* viewing waypoint *w* exactly *n* times for Commander ℓ
  - $u_{i,w,n}^l = \mathrm{Access_i^w} \times \log_{10}(n+1)$

Constraint Summary

- All tasks must be scheduled within the scheduling/time window
- A single sensor can only perform one task at a given timestep [5]
- Sensors can only be scheduled for a specific timestep if they are available
- Only 1 collection ID per tip can be scheduled within a time window
- Ensure that the number of sensor type looks correspond to the selected collection ID option
- Ensure the collections map to the waypoints in the corresponding polygons
- An optional rule: ensure at least one request per mission commander is executed in each schedule

Objective Function

- Variables:

$u_{i,w,n}^{l}$ = utility of sensor *i* viewing waypoint *w* exactly *n* times for Commander ℓ

$\varphi_{i,w,n}^{l}$ = binary variable that expresses sensor *i* views waypoint *w* exactly *n* times for Commander ℓ

r_l = rank of Commander ℓ

$\beta_{i,w,n}^{c}$ = binary variable that expresses sensor *i* executes waypoint *w* exactly *n* times for collection ID *c*

v_c = collection value associated with collection ID *c*

- Maximize overall utility through maximizing objective function, J :

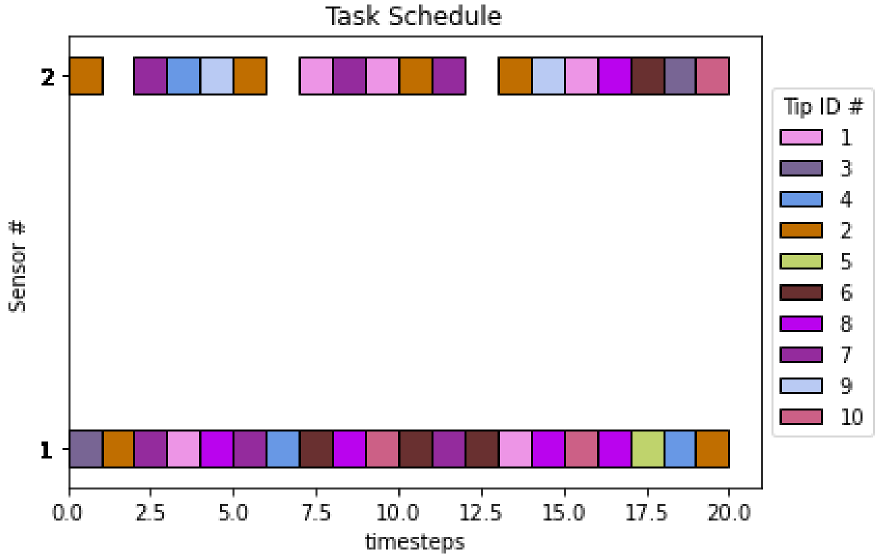
$$J = \sum_{i=1}^{S} \sum_{w=0}^{W} \sum_{n=0}^{N} \sum_{l=1}^{L} \varphi_{i,w,n}^{l} u_{i,w,n}^{l} r_l + \sum_{i=1}^{S} \sum_{w=0}^{W} \sum_{n=0}^{N} \sum_{c=1}^{C} \beta_{i,w,n}^{c} u_{i,w,n}^{l} v_c$$

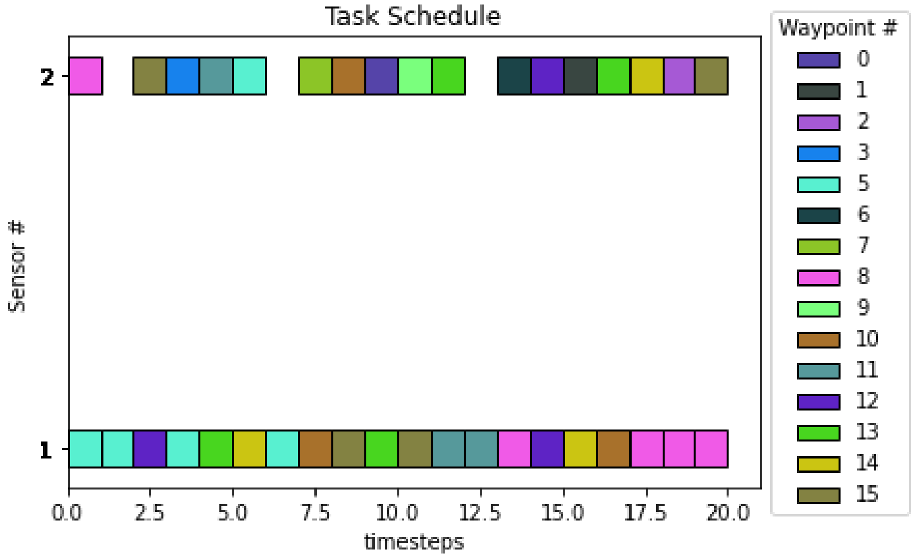
MILP Optimization Setup

- MILP model implemented in Python using Pyomo [6]
  - Optimization modeling package
  - Allows encoding of variables, constraints, and objectives
  - Interfaces directly to various optimization solvers
- Open-source CBC [7] and licensed Gurobi [8] provide numerical optimization of MILP models

Small-Scale Example Results

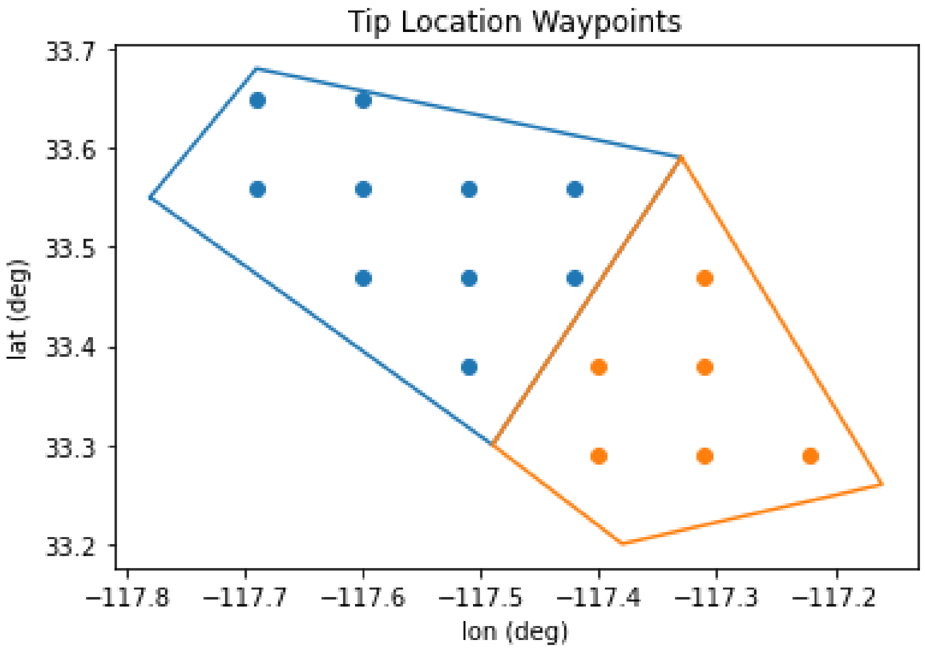
Number of Mission Commanders	1
Number of Tips	10
Number of Polygons	2
Number of Collection Options	82
Number of Waypoints	16
Number of Sensors	2
Number of Timesteps	20

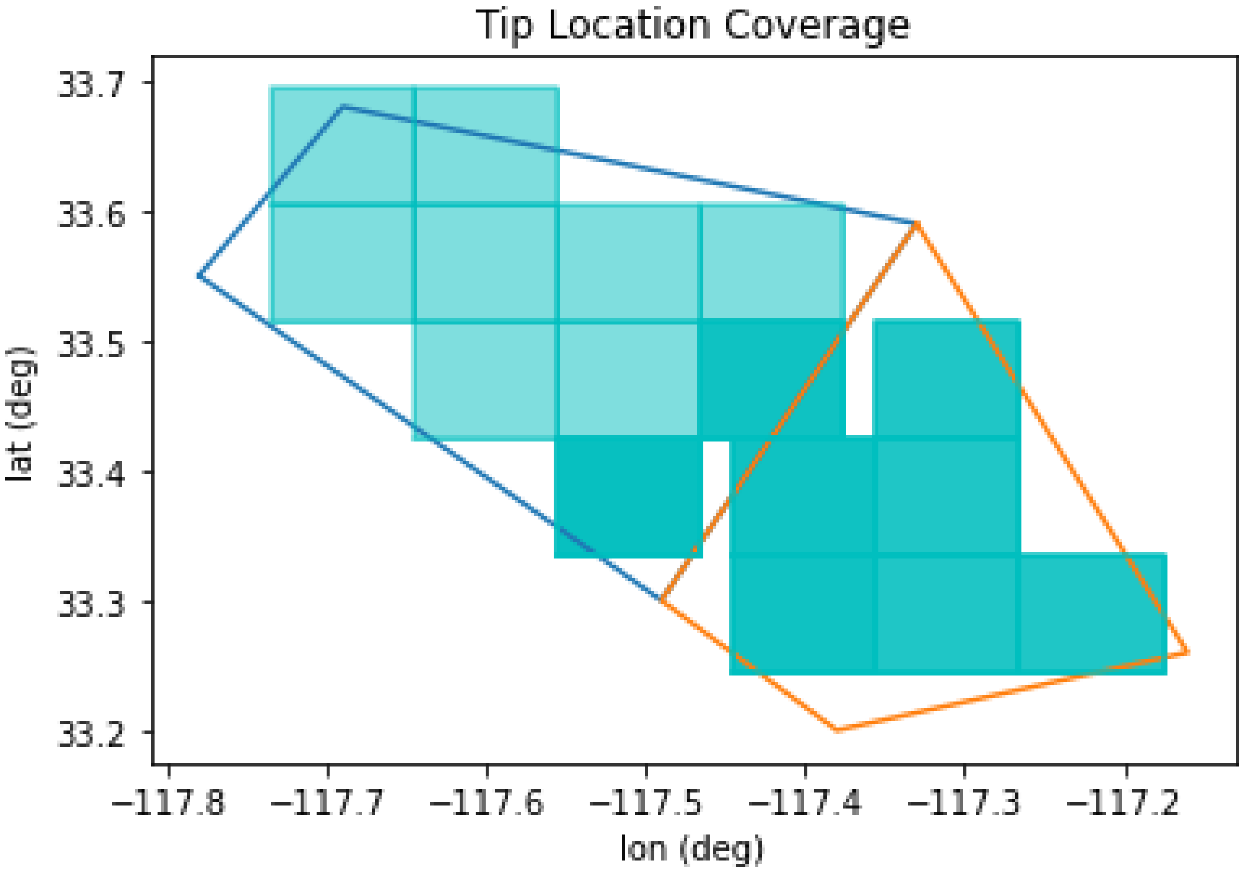




Note: Time gaps in schedule are due to unavailable sensors

Small-Scale Example: Polygon Coverage



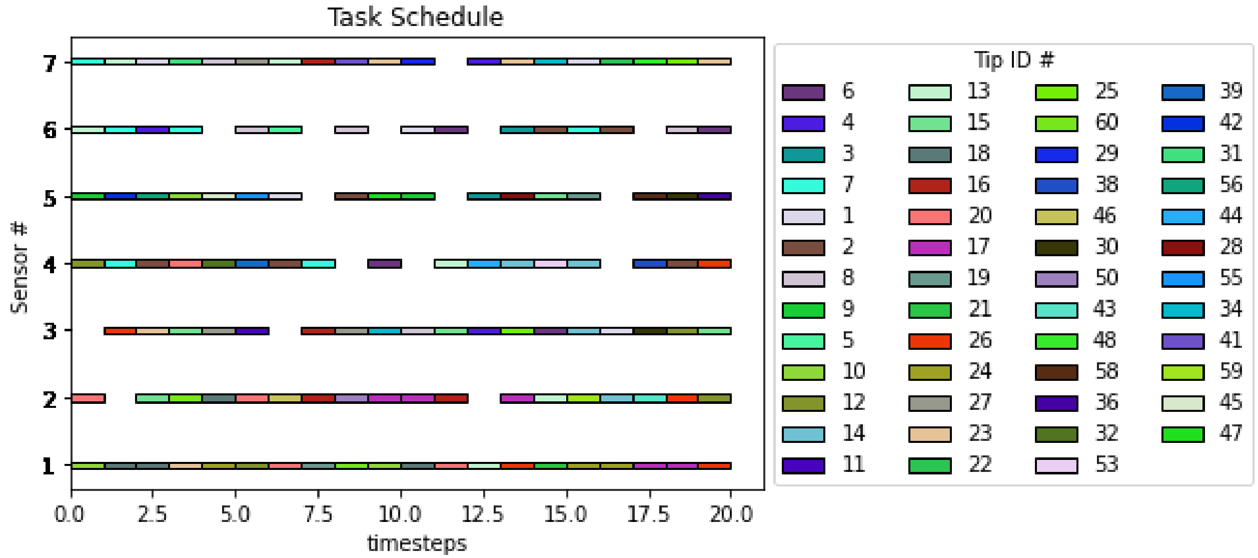


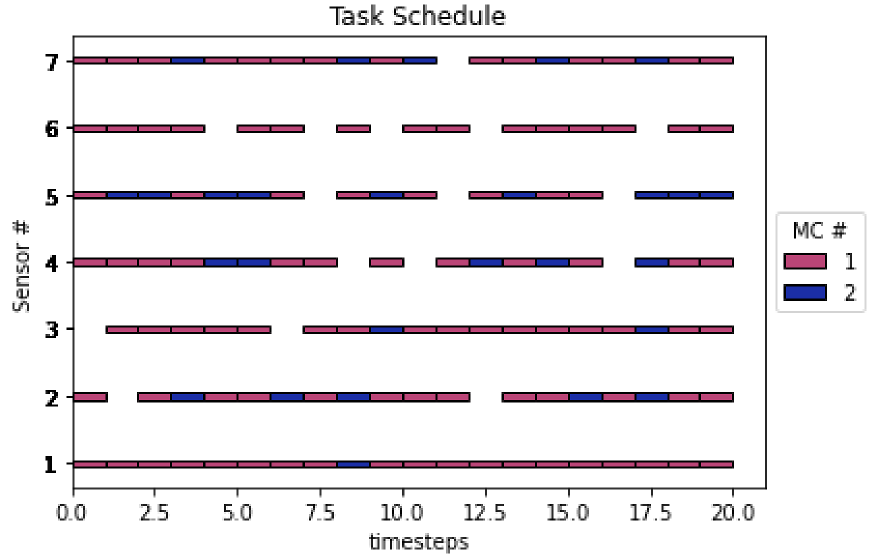
Opacity of footprints determined by number of looks to a given waypoint

Larger-Scale Example Results

Number of Mission Commanders	2
Number of Tips	60
Number of Polygons	5
Number of Collection Options	346
Number of Waypoints	69
Number of Sensors	7
Number of Timesteps	

Note: Time gaps in schedule are due to unavailable sensors





Computation Time Comparison

- Small-Scale: Gurobi solves faster than open-source CBC
- Larger-Scale: Exceeds bounds of CBC solver, solved in minutes with Gurobi

	Small-Scale		Larger-Scale	
	CBC	**Gurobi**	**CBC**	**Gurobi**
Build Time (s)	4.65	4.51	N/A	105.49
Solve Time (s)	16.32	5.93	N/A	169.53
Total Time (s)	20.97	10.44	N/A	275.02

Conclusion and Future Work

- Formulation provides flexibility to update computations of utility, access, feasibility, and objectives as this work evolves
- Algorithm is scalable to handle varying model sizes and scenario complexities
- Ongoing development:
  - Incorporation of sensor dynamics to account for real-time sensor locations throughout the time window for access and feasibility constraints
  - Overlapping area requests and the completion of simultaneous collections thereby allowing more waypoints to be scanned in a schedule window
  - Non-myopic time planning to account for time windows in the future
  - Extend the deployment of this algorithm to real-world environments

References

[1] Richards, J., Patel, A., Thorpe, A., & Schlossman, R. (2019). Autonomous multi-platform sensor scheduling for intelligence, surveillance, and reconnaissance. In *2019 National Symposium on Sensor and Data Fusion*. MSS.

[2] Richards, J., Patel, A., Thorpe, A., & Schlossman, R. (2019). *Autonomous Multi-Platform Sensor Scheduling for Intelligence Surveillance and Reconnaissance* (No. SAND2019-11381C). Sandia National Lab.(SNL-NM), Albuquerque, NM (United States).

[3] Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). *Integer programming* (Vol. 271, pp. 67-70). Berlin: Springer.

[4] Vielma, J. P. (2015). Mixed integer linear programming formulation techniques. *Siam Review*, *57*(1), 3-57.

[5] Valicka, C. G., Garcia, D., Staid, A., Watson, J. P., Hackebeil, G., Rathinam, S., & Ntaimo, L. (2019). Mixed-integer programming models for optimal constellation scheduling given cloud cover uncertainty. *European Journal of Operational Research*, *275*(2), 431-445.

[6] Hart, W. E., Laird, C. D., Watson, J. P., Woodruff, D. L., Hackebeil, G. A., Nicholson, B. L., & Siirola, J. D. (2017). *Pyomo-optimization modeling in python* (Vol. 67, p. 277). Berlin: Springer.

[7] COIN-OR. (2014). *COIN-OR Branch-and-Cut solver*. GitHub. Retrieved September 21, 2022, from https://github.com/coin-or/Cbc

[8] *Gurobi Optimizer*, Gurobi. (2021, November 19). Retrieved September 2022, from http://www.gurobi.com