Short-Term Conflict Resolution for Unmanned Aircraft Traffic Management

Hao Yi Ong and Mykel J. Kochenderfer

Stanford University

September 22, 2015

Outline

Unmanned aircraft traffic management

- automating conflict avoidance is critical to integrating small unmanned aerial systems (UAS) to civil airspace
- automation to augment controllers

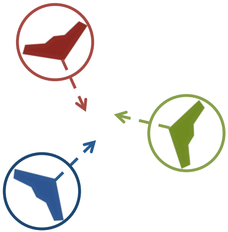


Conflict avoidance

given potential conflicts between drones, find the best set of advisories

- focus on horizontal or co-altitude conflict resolution
  - NASA's proposed airspace for UAS is under 150 m (500 ft)
  - flight altitudes need to avoid disruption and buildings

- conflict defined as loss of minimum separation between aircraft



Complications

- multiagent problem
  - system must coordinate between many aircraft
  - large search space for solution
- uncertainty in system
  - imperfect sensor measurements of current state
  - variable pilot response, vehicle performance, etc. affect future path
- appropriate trade-off between safety and efficiency not obvious

Goals

robust and efficient method

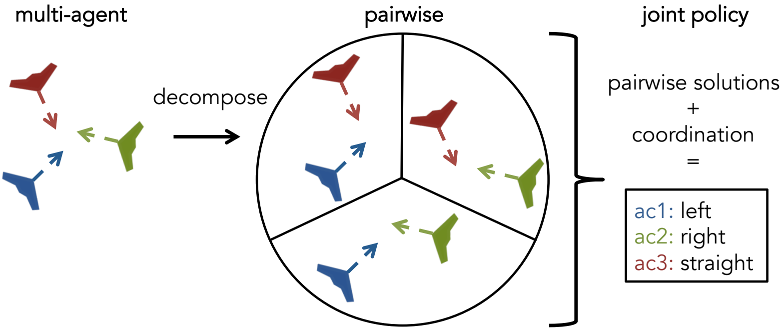
- tractable approach to complex stochastic problem
  - account for uncertainty in large problem with reasonable compute resources
- balances between airspace safety and efficiency
  - ensure safety and provide timely conflict alerts to aircraft
- arbitrary-scale optimization
  - real-time conflict resolution for large airspace

Previous approaches

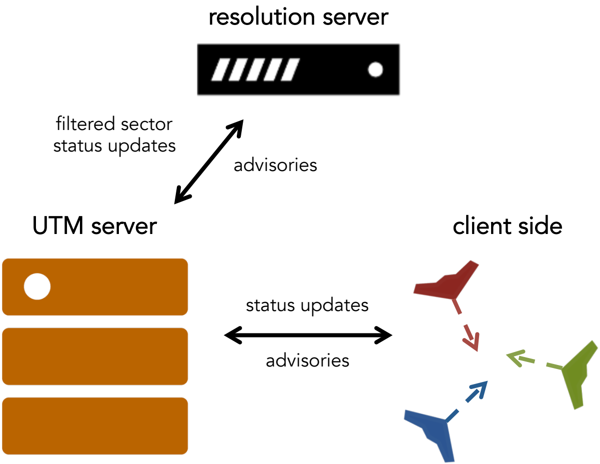
- mathematical programming (MIP, SCP) [SVFH05, ASD12]
  - can work well for simple vehicle networks
- distributed convex optimization [OG15]
  - hard to incorporate stochastic objectives/constraints
- Markov decision process formulation (ACAS X) [KC11, CK12]
  - assumes "white noise" accelerations for intruder aircraft
- and many more... [KY00]

Overview

idea: decomposition + coordination



Conflict Resolution as a UTM service



Conflict Resolution as a Standalone service

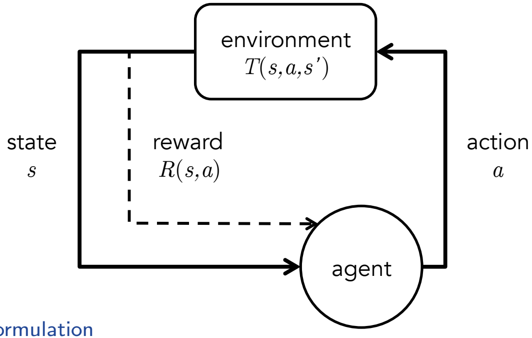
- derivative UTM client service
  - subscribes to UTM server to track aircraft and deconflict routes
  - pub-sub system that operators subscribe to to receive advisories
- current implementation uses standalone model
  - still unclear about end architecture for resolution service
  - clean approach to get started with prototype system
  - implemented with scalability and modularity in mind

Outline

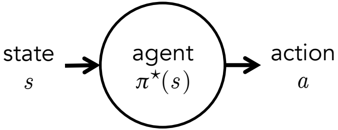
Markov decision process (MDP)

defined by the tuple $(\mathcal{S}, \mathcal{A}, T, R)$

- $\mathcal{S}$ and $\mathcal{A}$ are the sets of all possible states and actions, respectively
- $T\,(s, a, s')$ gives the probability of transitioning into state s' by taking action a at the current state s
- $R\,(s, a)$ gives the reward for taking action a at the current state s



Value iteration



- want to find the optimal policy $\pi^\star(s)$
  - gives action that maximizes the utility $Q^\star(s,a)$ from any given state

$$\pi^\star(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^\star(s,a)$$

- value iteration updates value function guess $\hat{Q}$ until convergence
  - expensive one-off compute but cheap policy extraction ($\hat{Q}$ lookup)

$$\hat{Q}(s,a) := R(s,a) + \sum_{s' \in \mathcal{S}} T(s,a,s') \max_{a' \in \mathcal{A}} \hat{Q}(s',a')$$

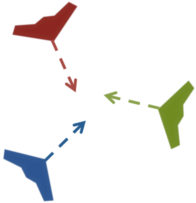
Multiagent MDP

extension of MDP to cooperative multiagent setting

- similar to case where centralized planner has access to system state
- also defined by the tuple $(\mathcal{S}, \mathcal{A}, T, R)$, except
  - $\mathcal{S}$ and $\mathcal{A}$ are all possible joint states and actions
  - T and R operate on elements of $\mathcal{S}$ and $\mathcal{A}$

Short-term conflict resolution

- horizontal conflict resolution between n aircraft
  - conflict defined as loss of minimum separation distance, 500 m
  - aircraft at risk if it could experience a conflict within two minutes
- alert aircraft that need corrective maneuvers
  - but not to the extent that alerts become a nuisance



Resolution advisories

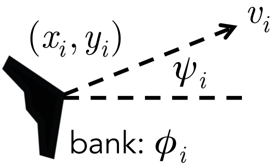
- at each time step $T = 5\,\text{s}$, system issues a joint advisory
  - joint advisory ϕ chosen out of a finite set of corrective bank angles for each aircraft
- action set $\mathcal{A} = \{-20^\circ, -10^\circ, 0^\circ, 10^\circ, 20^\circ, \text{COC}\}^n$
  - positive and negative angles correspond to left and right banks
  - COC is a clear-of-conflict status advisory—nothing needs to be done
- higher resolution comes at the cost of computational complexity

Dynamics

- ith aircraft state $s_i = (x_i, y_i, \psi_i, v_i, p_i)$
  - latitude and longitude x_i and y_i
  - heading ψ_i
  - groundspeed v_i
  - responsiveness indicator p_i

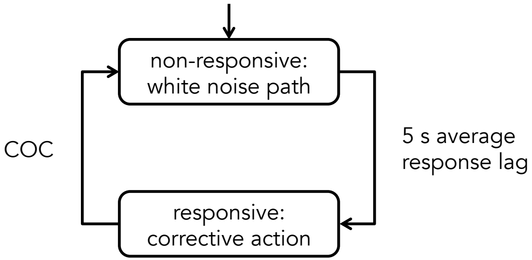
- aircraft follow Dubin's kinematic model
  - simplicity avoids risk of overfitting complicated models

$$\dot{x}_i = v_i \cos \psi_i \qquad \dot{y}_i = v_i \sin \psi_i \qquad \dot{\psi}_i = \frac{g \tan \phi_i}{v_i}$$



Pilot model

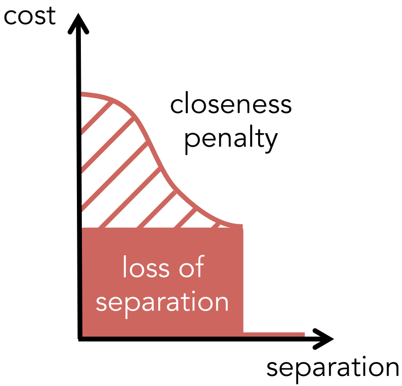
- advisory response determined stochastically by Bernoulli process
- 5 s average response delay to first corrective advisory as recommended by ICAO [ICA07]
  - when responding, the pilot executes the advisory for 5 s time step



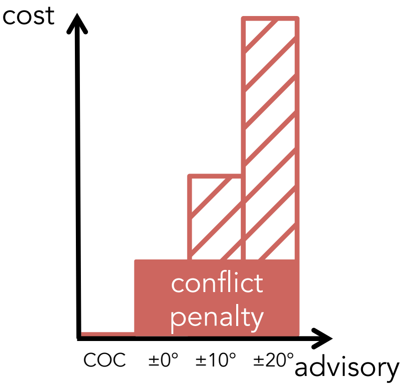
Reward function

balance competing objectives

- maximize safety



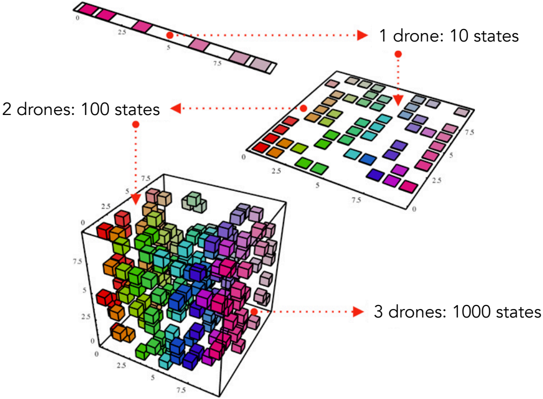
- minimize disruption



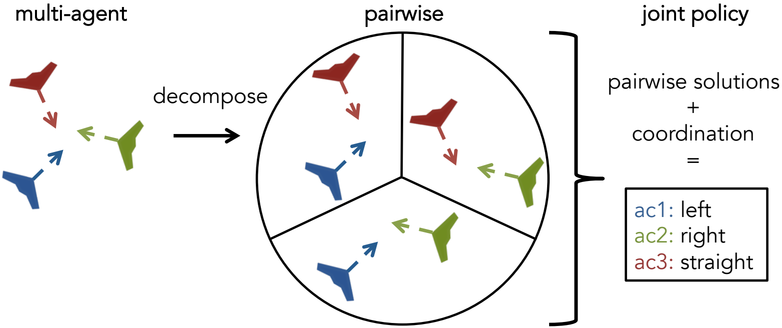
Outline

Curse of dimensionality

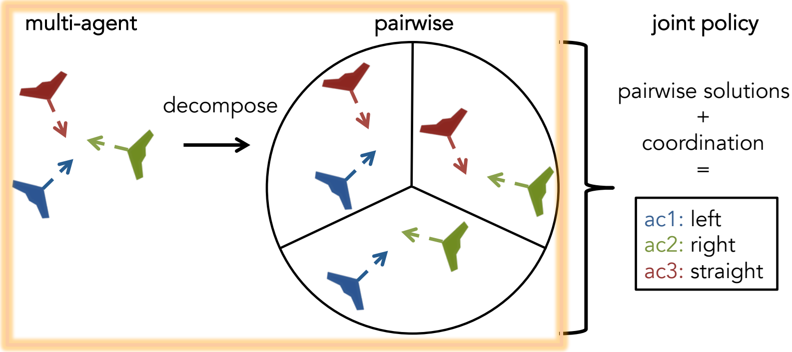
- in an MMDP, $\mathcal{S}$ and $\mathcal{A}$ are all possible joint states and actions
- **problem:** $\mathcal{S}$ and $\mathcal{A}$ scale exponentially with number of agents



Decompose and coordinate



MMDP decomposition

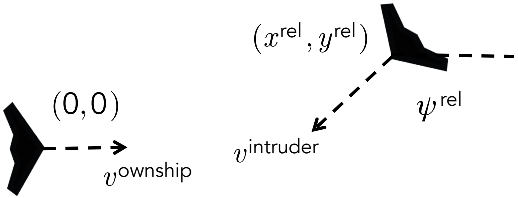


Pairwise conflict resolution

decompose full MMDP into $O\left(n^2\right)$ pairwise encounters

- action set, pilot response model, and reward function unchanged
- use relative positions and bearing to reduce state space
  - arbitrarily set one aircraft as ownship, and the other as intruder
  - speeds remain in global reference frame

$$s = \left(x^{\text{rel}}, y^{\text{rel}}, \psi^{\text{rel}}, v^{\text{ownship}}, v^{\text{intruder}}, p^{\text{ownship}}, p^{\text{intruder}}\right)$$



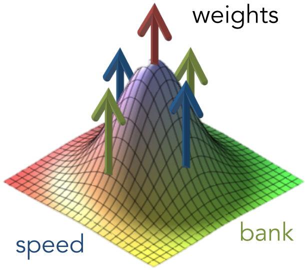
Discretization

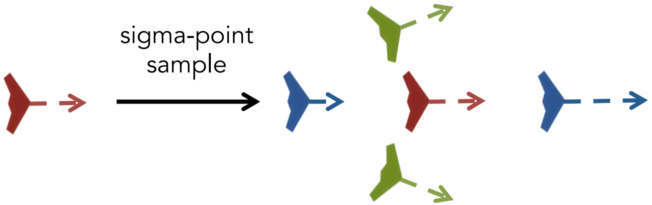
- in general, no analytical solution to problem with continuous state space
- approximate value function $\hat{Q}$ over discretized state space
  - discretize state space via multilinear interpolation
  - iteratively update $\hat{Q}$ until convergence

$$\hat{Q}\left(s,a\right) := R\left(s,a\right) + \sum_{s' \in \mathcal{S}} T\left(s,a,s'\right) \max_{a' \in \mathcal{A}} \hat{Q}\left(s',a'\right)$$

Approximating environment noise

sigma-point sampling





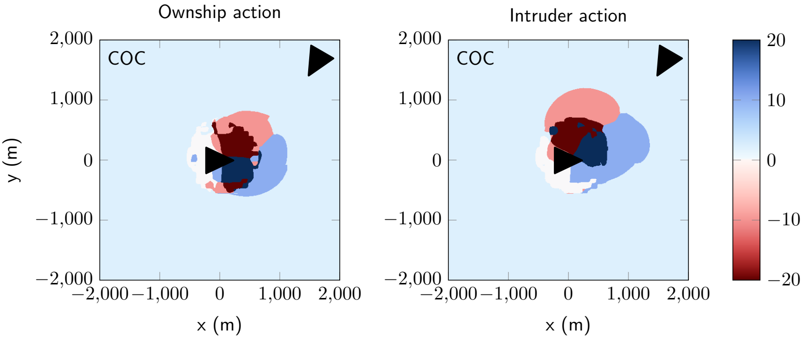
Implementation

- discretize continuous state space into 9.6 million states
  - expensive but one-off computation
- $\hat{Q}$ converges in 7 hours (40 iterations over $\mathcal{S} \times \mathcal{A}$)
  - solver: parallel implementation in Julia
  - machine: 20 2.3 GHz Intel Xeon cores with 125 GB RAM

Variable	Minimum	Maximum	Number of values
$x^{\mathrm{rel}}, y^{\mathrm{rel}}$	$-3000\,\mathrm{m}$	$3000\,\mathrm{m}$	51
ψ^{rel}	$0°$	$360°$	37
$v^{\mathrm{ownship}}, v^{\mathrm{intruder}}$	$10\,\mathrm{m\,s^{-1}}$	$20\,\mathrm{m\,s^{-1}}$	5
$p^{\mathrm{ownship}}, p^{\mathrm{intruder}}$	-	-	2

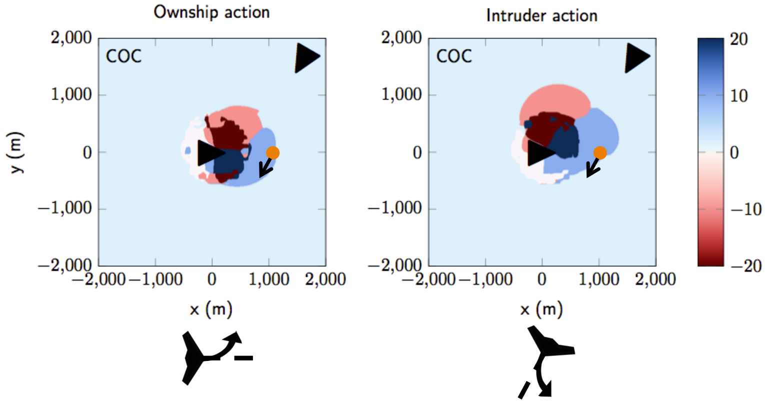
Pairwise policy visualization

passing intrusion (240° bearing)



Example advisory execution

passing intrusion at $(1000\,\mathrm{m}, 0\,\mathrm{m})$

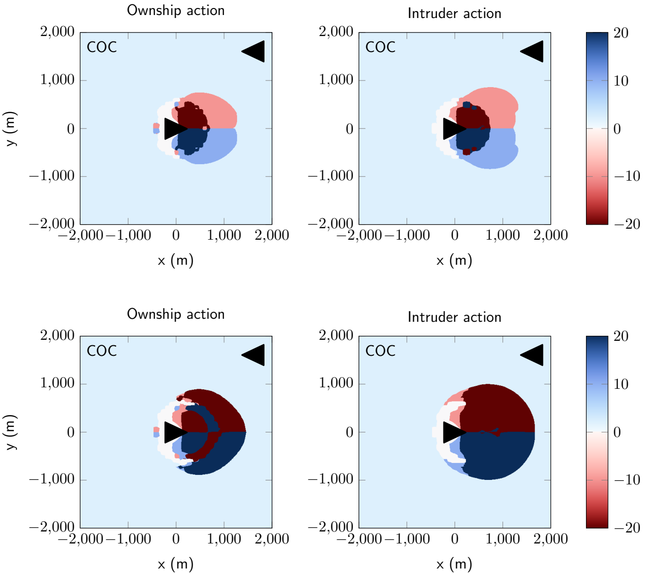


ownship: 10° left bank

intruder: 10° left bank

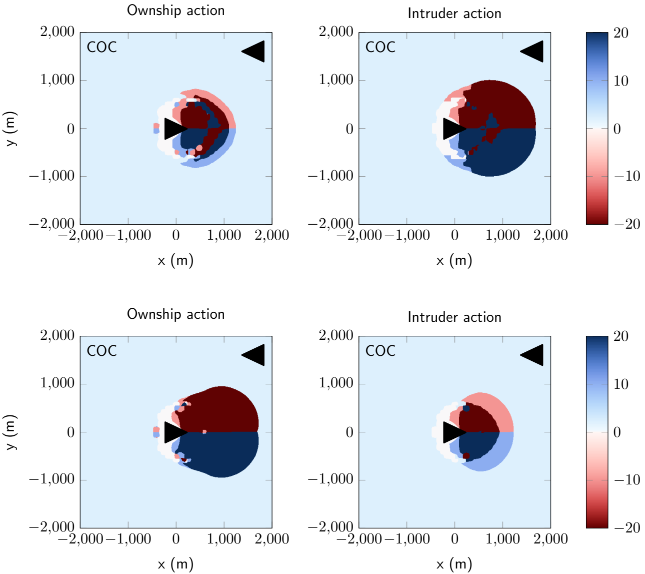
Accounting for pilot response

pilots responsive (top) and non-responsive (bottom)

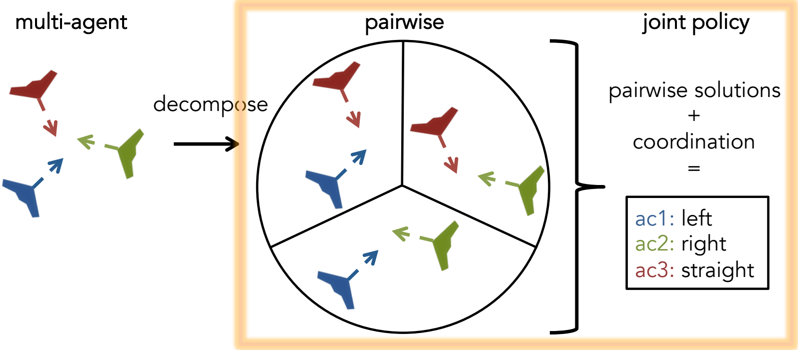


Accounting for pilot response

ownship responsive (top) and intruder responsive (bottom)



Multithreat coordination



Utility fusion

idea: combine pairwise utilities Q_{ij} to form proxy for global utility Q

- Q_{ij} operates on the pairwise MDP state-action pair (s_{ij}, a_{ij}) for aircraft i and j
- fusion strategies $\pi(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(s, a)$
  - max-min: "worst-case"

$$Q^{\text{min}}(s, a) = \min_{i<j} \{U_{ij}(s_{ij}, a_{ij})\}$$

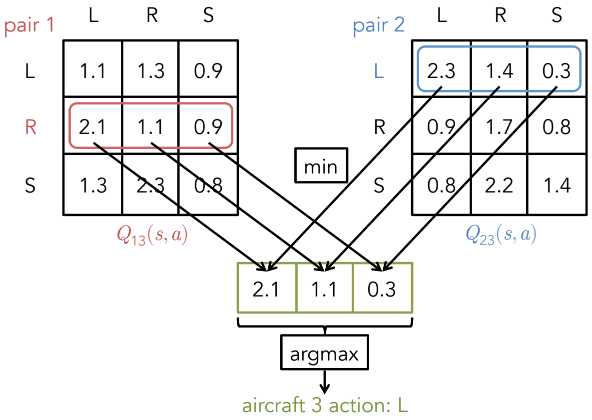
  - max-sum: "average"

$$Q^{\text{sum}}(s, a) = \sum_{i<j} U_{ij}(s_{ij}, a_{ij})$$

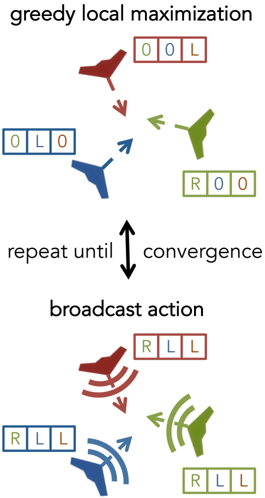
problem: action space $\mathcal{A}$ scales exponentially with number of aircraft

Search heuristic: Alternating maximization

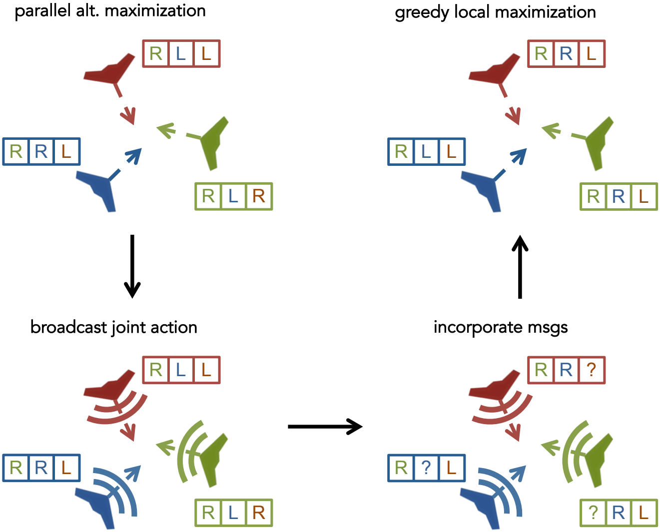
- maximize global utility by varying one action and fixing the rest
- e.g., fix drone 1 and 2's actions (R and L) and vary drone 3's action



Distributed variant I: Parallel search

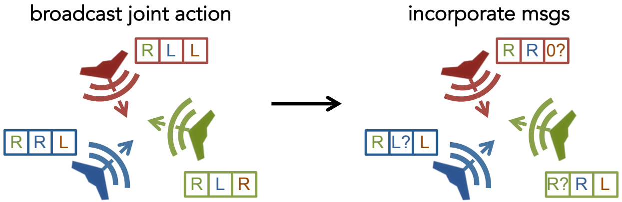


Distributed variant II: Consensus search



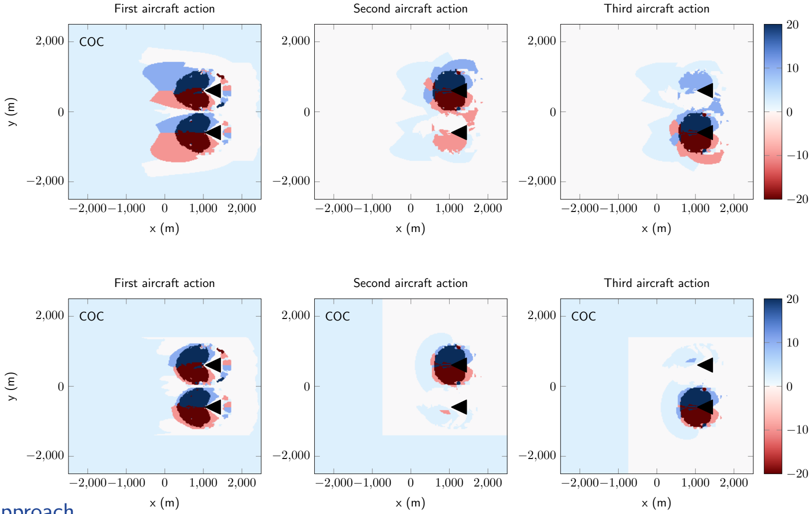
Consensus search message

- parallel search could incur large communication cost
- messages reduce communication between compute nodes, but risk yielding unsynchronized joint action across aircraft
- incorporate messages by "averaging" what intruders think ownship should do to approximate consensus



Joint policy visualization

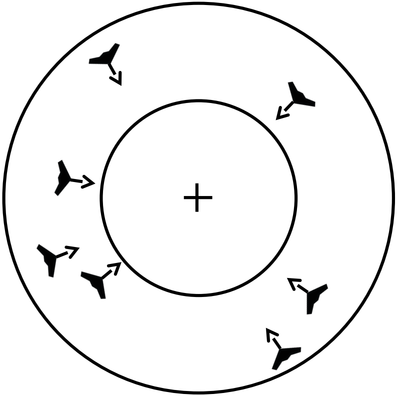
triple aircraft encounter max-min (top) and max-sum policies (bottom)



Outline

Encounter model

- generate uniformly random starting aircraft states in annulus
  - speeds uniformly random
  - headings initialized to point to annulus center
  - resample if loss of separation



Aircraft model

- ODE solver for dynamics
  - continuous Dubin's kinematic model
  - Gaussian noise in acceleration and banking
- corrective bank angles mapped to PID control policies

Baseline methods

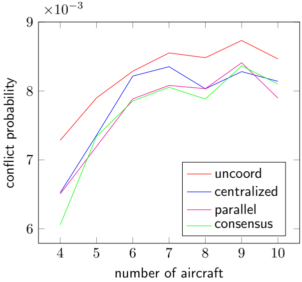
- closest threat
  - multithreat scenario decomposed into pairwise encounters
  - aircraft execute solution to encounter with closest intruder
- uncoordinated
  - aircraft assumes all other aircraft are white-noise intruders
  - executes greedy local maximization to find own action

Experiment

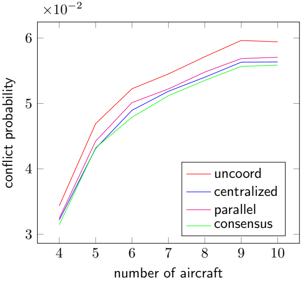
- $>$1 million simulations from 2 to 10 aircraft
  - code: Julia implementation
  - machine: 3.4 GHz Intel i7 processor with 32 GB RAM
- $\sim$10 ms decision times (vs. 5 s decision period)
  - serial solve times of $<$10 ms for up to 10 aircraft sufficient for server-based resolution system
  - unoptimized code that doesn't account for communication costs for distributed variants

Safety performance

- max-min utility fusion



- max-sum utility fusion



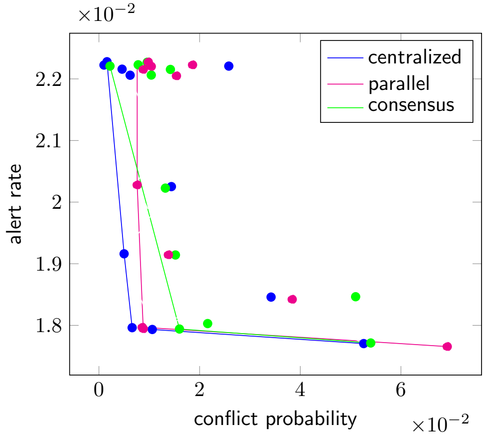
- distributed variants perform as well as serial version
- coordinated methods >10 times better than closest threat heuristic
- coordinated methods ∼10% better than uncoordinated method

Safety performance takeaways

- no near mid-air collisions (NMACs) in simulations
  - NMAC threshold distance defined at 30 m (100 ft)
  - due to penalty on smaller separation distances even in conflict
- simulations validate all algorithmic variants
  - distributed variants perform as well as serial version
  - coordinated methods significantly better than baselines

Trade-off: Safety vs. alert rate

- vary relative penalty between loss of separation and disruption
- best performance at "knee" of Pareto frontier

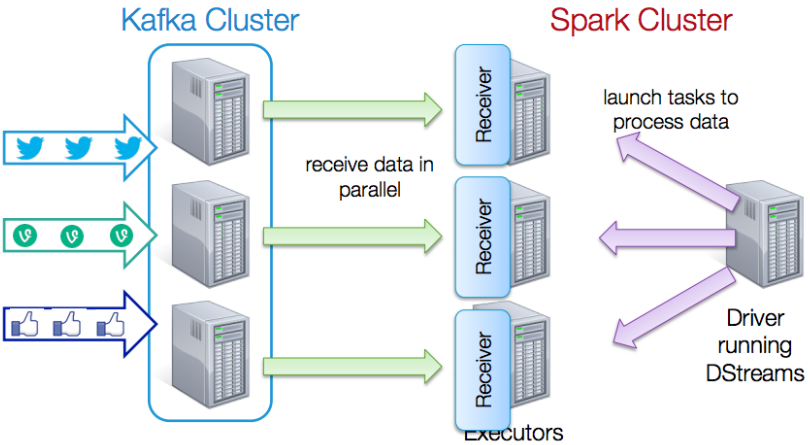


Outline

Practical conflict avoidance

- recall: pub-sub system that UTM clients subscribe to for advisories
  - standalone system that subscribes to UTM server for flight tracking
- **goals:** robustness, scalability, and modularity
  - cluster computing framework for large-scale data processing
  - streaming analytics for real-time conflict resolution
  - built-in system fault-tolerance

System design



System design

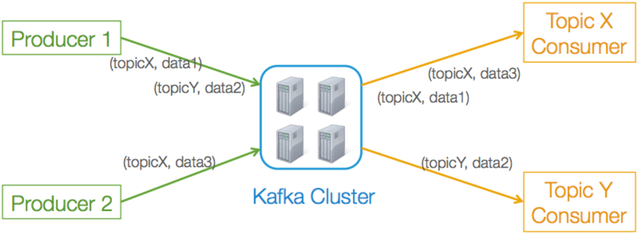
- why Kafka?
  - designed as unified, high-throughput, low-latency platform for handling real-time data feeds
  - open source with active industry use (originally developed by LinkedIn)
- why Spark?
  - easy, high-level API
  - ease of operations with minimal fuss over fault-tolerance
  - enormous, active community and industry deployments (CISCO, Netflix, Intel,. . .)

Driver-worker model

- driver-worker model for conflict resolution
  - driver loads up policy object that contains lookup table and algorithm for conflict resolution
  - workers receives policy object via broadcast and subsequently delegated tasks by driver

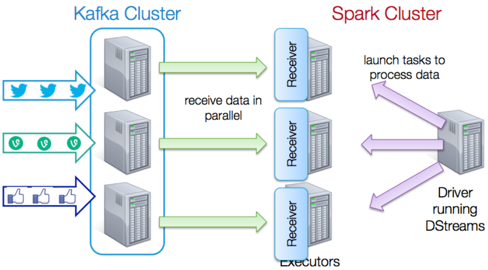
- formatted conflict data stored as an “infinite” stream of resilient distributed datasets (RDDs)
  - distributed, partitioned collection of conflict objects (JSON)
  - automatically rebuilt on failure

Pub-sub model



- ingestor publishes conflict objects to `conflict` Kafka topic
  - subscribes to and ingests UTM server stream and crudely identifies potential conflicts by proximity
  - formats conflict data as JSON strings
- advisor publishes advisories to `advisory` Kafka topic
  - producers are worker nodes in cluster generating advisories
  - consumers are UTM operator clients flying drones

Working model



- Spark driver launches workers/executors and broadcasts policy object
- workers receive conflict data in parallel from `conflict` Kafka topic as RDD stream
- workers publish advisories to `advisory` Kafka topic for UTM operator clients

Outline

Summary

- three variants of coordination-based conflict resolution algorithm
  - 1 centralized/serial and 2 distributed variants
- millisecond solve times for real-time application for multiple aircraft
- robust distributed system for practical conflict avoidance

Implementation

- policy generation on Julia scientific programming language

  `https://github.com/sisl/ConflictAvoidanceDASC`

- distributed system with Apache Kafka and Spark Streaming

  `https://bitbucket.org/sisl/utm-alpha`

Further research

- tracking aircraft via belief states
  - use the UTM client server to track reported aircraft states
  - generate local set of belief states for all aircraft to better estimate state uncertainty in real-time
- best action for drone in case of communication loss
  - figure out the response state of the aircraft via belief state tracking
  - where to go in case of communication loss, what flight profile to take, and what UTM should assume about the drones flight
- dealing with adversarial flights

Further software development

- standardize resolution advisory and advisory receipt formats with focus on minimizing message size
- quantify delay times for sending packets between advisory server and clients in order to model it into the problem

References

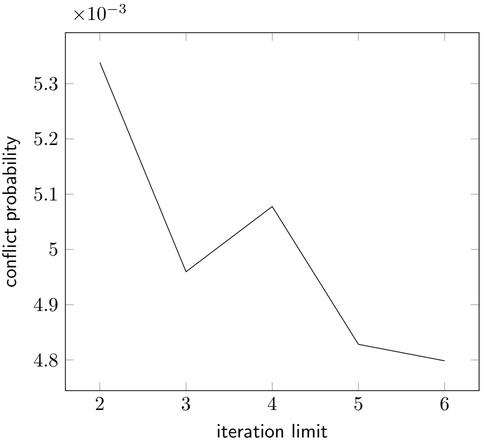
- Federico Augugliaro, Angela P Schoellig, and Raffaello D'Andrea.
  Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach.
  In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1917–1922, October 2012.
- J. P. Chryssanthacopoulos and M. J. Kochenderfer.
  Decomposition methods for optimized collision avoidance with multiple threats.
  Journal of Guidance, Control, and Dynamics, 35(2):398–405, 2012.
- International Civil Aviation Organization ICAO.
  Surveillance, radar and collision avoidance.
  International Standards and Recommended Practices, 4, 2007.

References

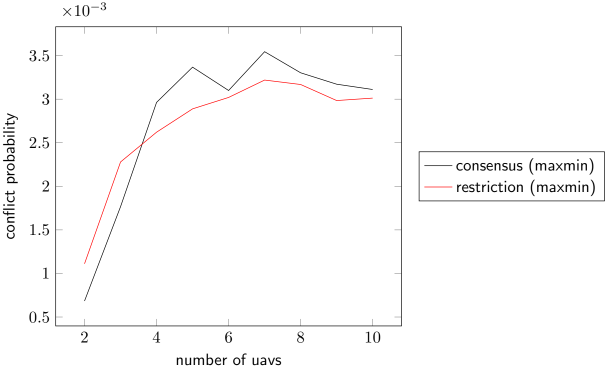
- M.J. Kochenderfer and J.P. Chryssanthacopoulos.
  Robust airborne collision avoidance through dynamic programming.
  Project Report ATC-371, MIT Lincoln Laboratory, 2011.
- J. K. Kuchar and L. C. Yang.
  A review of conflict detection and resolution modeling methods.
  IEEE Trans. Intell. Transp. Syst., 1(4):179–189, 2000.
- H. Y. Ong and J. C. Gerdes.
  Cooperative collision avoidance via proximal message passing.
  In *American Control Conference (ACC)*, July 2015.
- Tom Schouwenaars, Mario Valenti, Eric Feron, and Jonathan How.
  Implementation and flight test results of MILP-based UAV guidance.
  In *IEEE Aerospace Conference*, pages 1–13. IEEE, 2005.

Impact of search timeout

conflict decreases with iteration limit but with diminishing returns



Restriction messages for consensus search



- instead of broadcasting joint action, aircraft broadcast restriction messages that indicate what other aircraft shouldn't do
  - e.g., reward straight path when it receives no left and no right messages at a decision period