IOWA STATE UNIVERSITY

Scalable FastMDP for Pre-departure Airspace Reservation and Strategic De-conflict

Josh Bertram bertram1@iastate.edu
Joseph Zambreno zambreno@iastate.edu
Peng Wei pwei@gwu.edu

AIAA SciTech 2021

Introduction

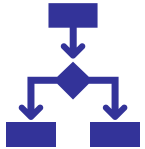
- Large scale air traffic management
  - Air taxi / package delivery
  - Hundreds, thousands of aircraft
  - Unpredictable demand, flight paths
- Need efficient pre-departure flight planning algorithms

Image Credit: NASA

Desired Outcome

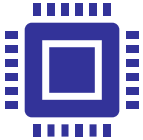
- Pre-Departure Flight Planner (PDFP)
  - Maintains list of approved flight plans
  - User submits source, destination, departure time
  - Returns flyable, conflict-free flight plan(s) with respect to approved flight plans, terrain, obstacles, etc
  - Suggests alternate departure times to minimize fuel / flight time / etc
- Centralized or Decentralized
- Quick planning, replanning
  - Single requests, batched requests
- Scale to thousands of aircraft, room to grow to higher scalability

Our Approach



Formulate problem as Markov Decision Process (MDP)

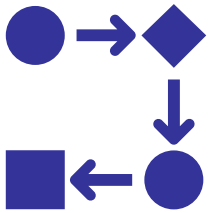
Solve MDP with FastMDP algorithm



Extend FastMDP algorithm to GPU to take advantage of parallelism

Show scalability to thousands of aircraft

Markov Decision Process



Framework for sequential decision making

Discrete time control system with feedback

Solution is a control law known as optimal policy

From Artificial Intelligence (AI) community

Used by ACAS-X (Partially Observable MDP)

Powerful, general approach

But, difficult to make tractable / efficient

Typical approaches are slow to solve MDP

But once solved, fast run-time execution

ACAS-X generates lookup table ~500MB - 1GB, for example

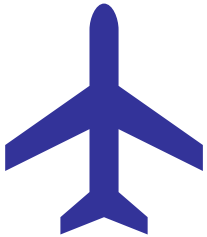
FastMDP Algorithm

Solves a "useful subclass" of MDPs in milliseconds

Applicable to aerospace and robotics problems

Adapts to rapidly changing environments

No training phase; solves on-line in real-time



Needs aircraft dynamics model and a Reward Function

Reward function controls behavior of agent

Provides positive and negative rewards

Built from "reward primitives"

FastMDP Control Flow

Fast MDP CPU Computational Performance

FastMDP CPU algorithm is O(|R| x |A|)

- |R| - Number of rewards
- |A| - Number of actions

For a given problem when |A| is fixed, variation in runtime performance reduces to O(|R|)

- Very good performance, but on CPU beyond ~75 aircraft, algorithm falls behind real time (wall clock time)
- Need method to scale beyond 70 aircraft

Number of aircraft	Timing (ms)
3	34.9
15	43.9
30	58.1
45	67.4
60	77.1
75	95.2
90	116.4
105	123.6
120	137.7
135	149.8
150	162.6

FastMDP Parallelism

Rewards function can be slightly parallelized

Value function computation is highly parallel

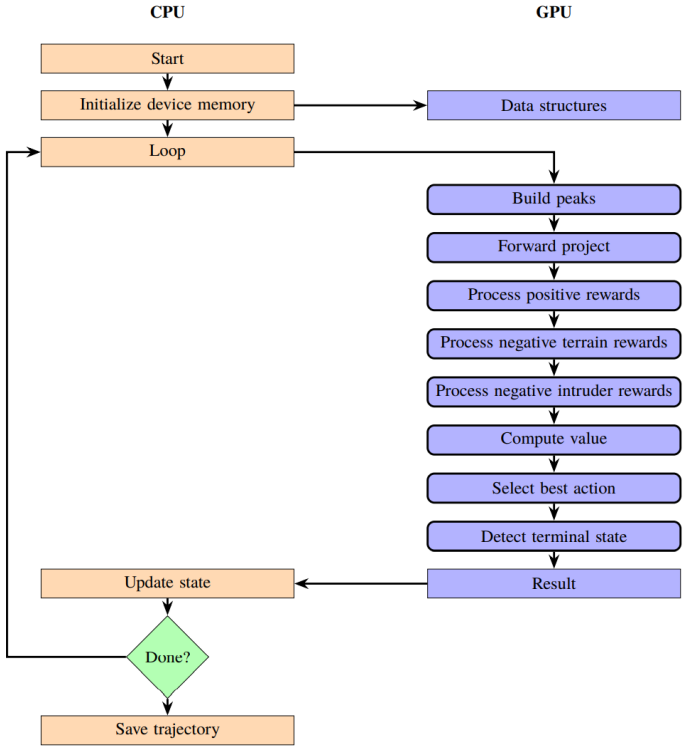
How can we take advantage of this parallism inherent in the algorithm?

Forward projection is extremely parallel

Fast MDP GPU Approach

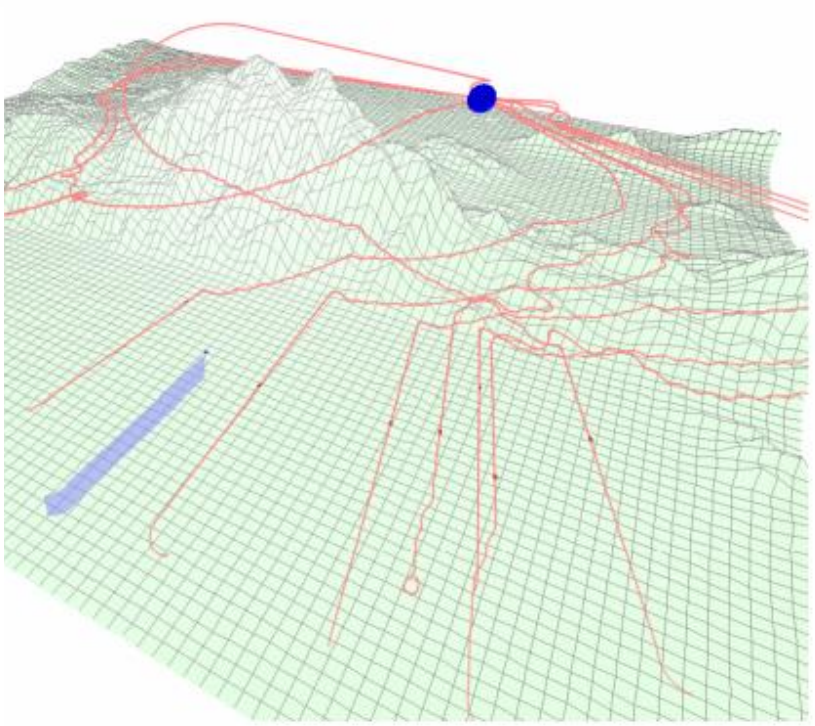
Build pipeline of kernels

- Each kernel solves part of the problem
- Each kernel runs many threads in parallel
- Takes advantage of parallelism available at each stage of the problem



Experimental Setup

- UTM-like simulation environment
- System-maintained approved flight plans (red tracks)
- User submits source, destination (blue sphere)
- System returns viable, conflict-free flight path (blue track)



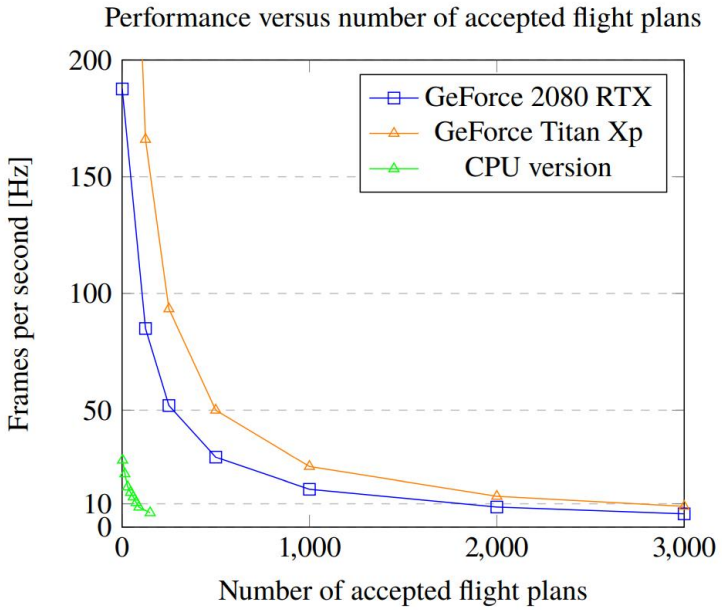
Computational Performance

Measures average time for each agent to perform one "cycle" of FastMDP:

- Formulate an MDP
- Solve MDP
- Forward project actions
- Select best action

Target performance: 100 ms or less (> 10 Hz)

Good performance even up to 2000 – 3000 other aircraft (10000 – 15000 rewards)



Questions?