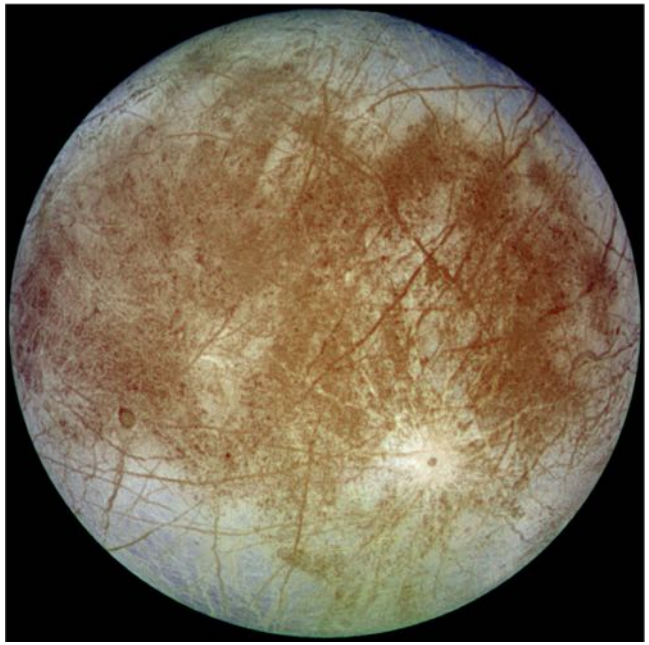
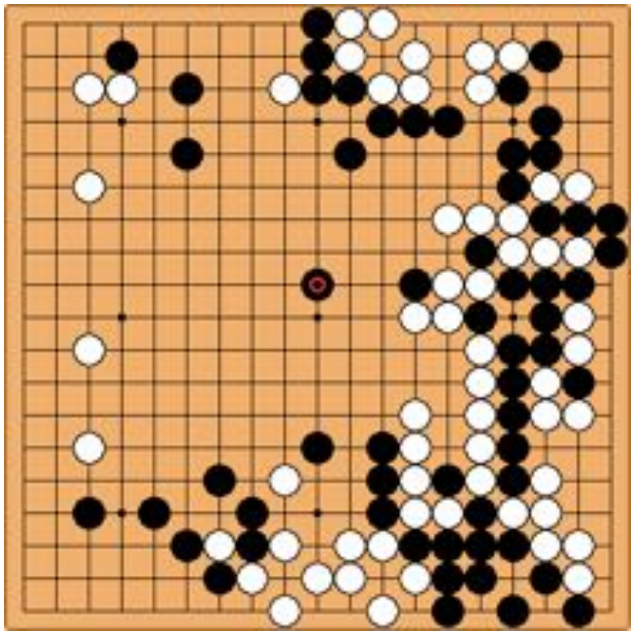
Risk-bounded Task and Motion Planning in a Real World



Prof. Brian Williams, MIT

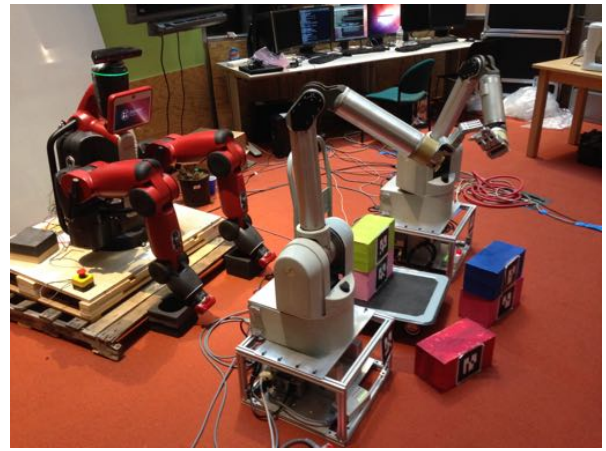
ICAPS Summer School
June 23rd, 2018
Delft, Netherlands

https://events.technologyreview.com/video/watch/innovators-under-35-lars-blackmore



Planning Pervades the Real World

Travel Advisors
and Personal Air Vehicles



Human-Robot Teams
For Manufacturing

Bottom Up Grids for Sustainable Homes

Information-gathering Scouts

Involve:

- motions
- time
- inertia
- hidden state


The real-world world is uncertain

Some levels of risk are unacceptable.

Need to manage uncertainty and risk at every level

"How long does this activity run?"

Temporal uncertainty

"Where is the robot?"

State uncertainty

"Did she turn left or right"?

Enumeration uncertainty

"Where did this Gaussian come from?"

Model uncertainty

"How much charge is left in the battery?"

Resource uncertainty

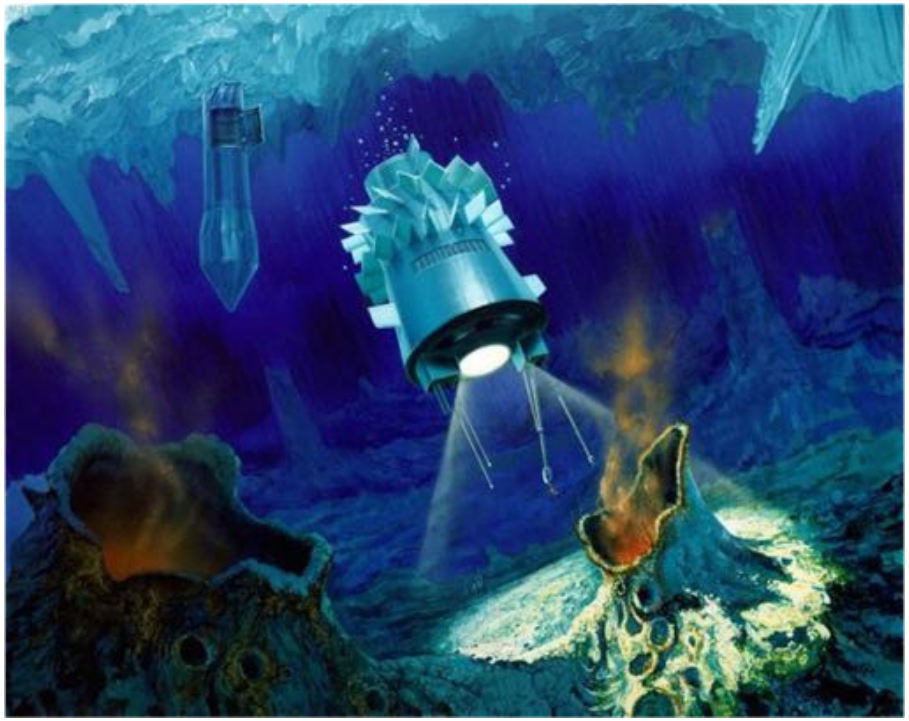
A Decision-Making Support Cast

1. Stochastic Optimization and Stochastic Constraint Programming
2. Probabilistic Scheduling (pSTNs)
3. Risk-bounded And-Or Search (RAO*)

Enables risk-bounded:

- Model-Predictive Control
- Temporal Plan Network Planning
- Conditional Hybrid Planning
- Generative Hybrid Planning

Today: Some Risky Business



Navigating the Oceans of Europa

Joint with WHOI, ACFR, U of M
SOI, Exxon & NASA

Navigating the Oceans of Boston Traffic

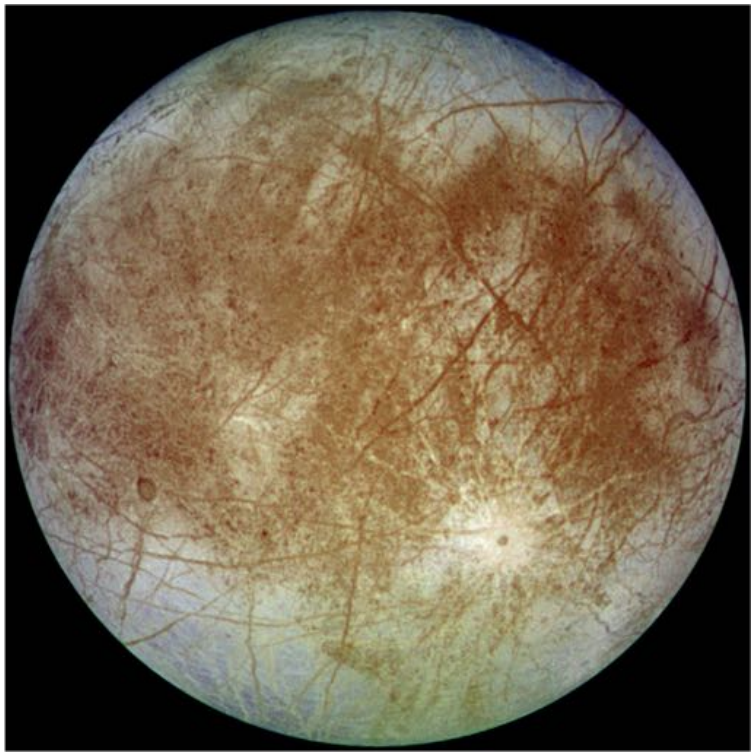
Joint with Toyota, Boeing & Masdar Inst.

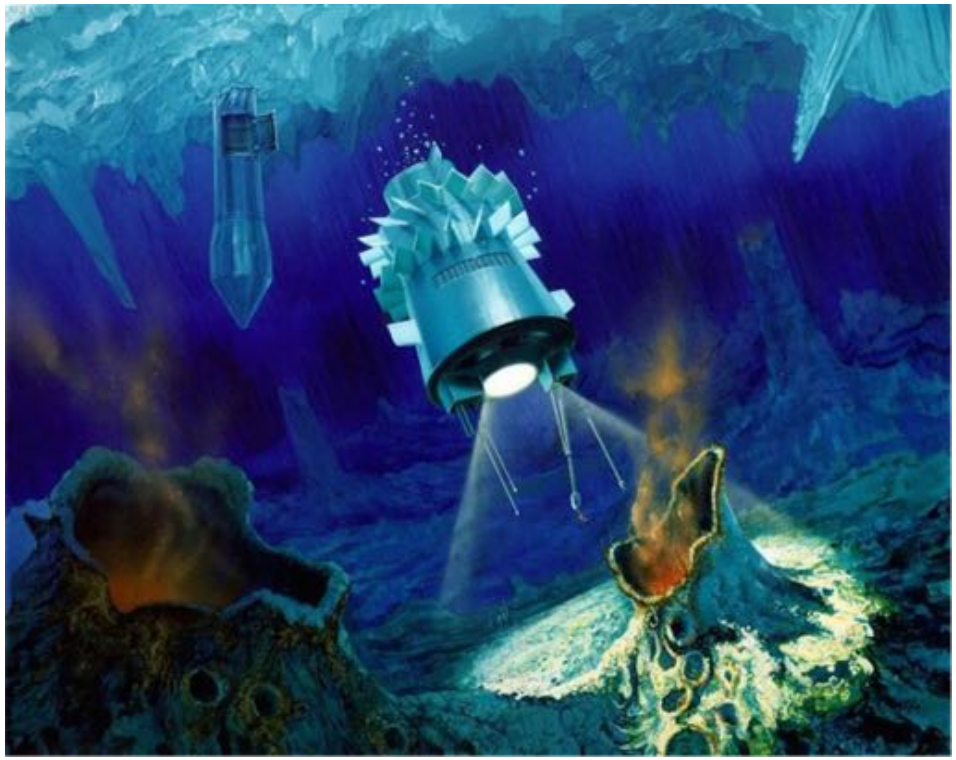
Outline

- Diving into the Real World
- Risk-bounded Planning as Optimization and Constraint Programming
- Generative Task and Motion Planning with Heuristic Forward Search
- Risk-bounded Planning by adding And / OR (AO*) Search


Individuals
WANTED
for hazardous journey, small wages, bitter cold, long months of complete darkness, constant danger, safe return doubtful, honor and recognition in case of success.
Ernest Shackleton 4 Burlington st.
How do we manage risks?

Navigating the Oceans of Europa





NASA's Ocean Worlds Program is introducing a new era of exploration

Europa

Ganymede

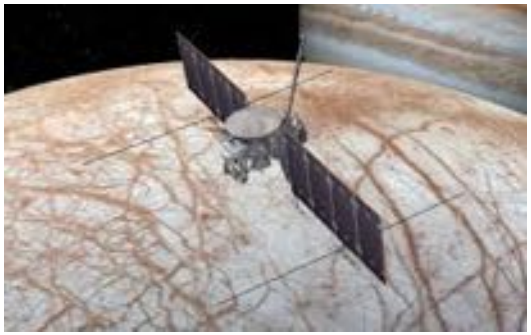
Callisto

Enceladus

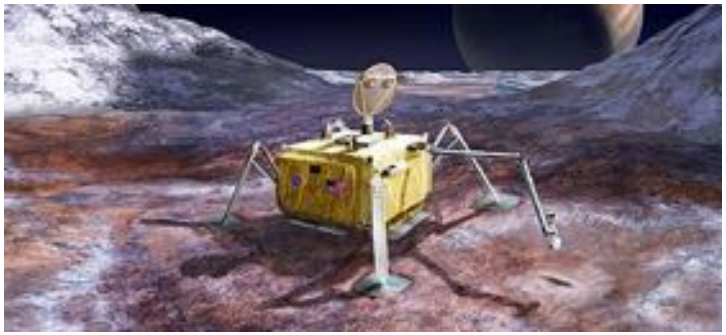
+ 4 others

Which worlds of our solar system have oceans of their own?

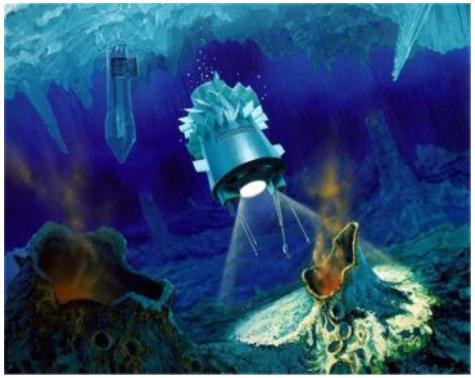
NASA's Ocean Worlds Program is introducing a new era of exploration



Europa Clipper



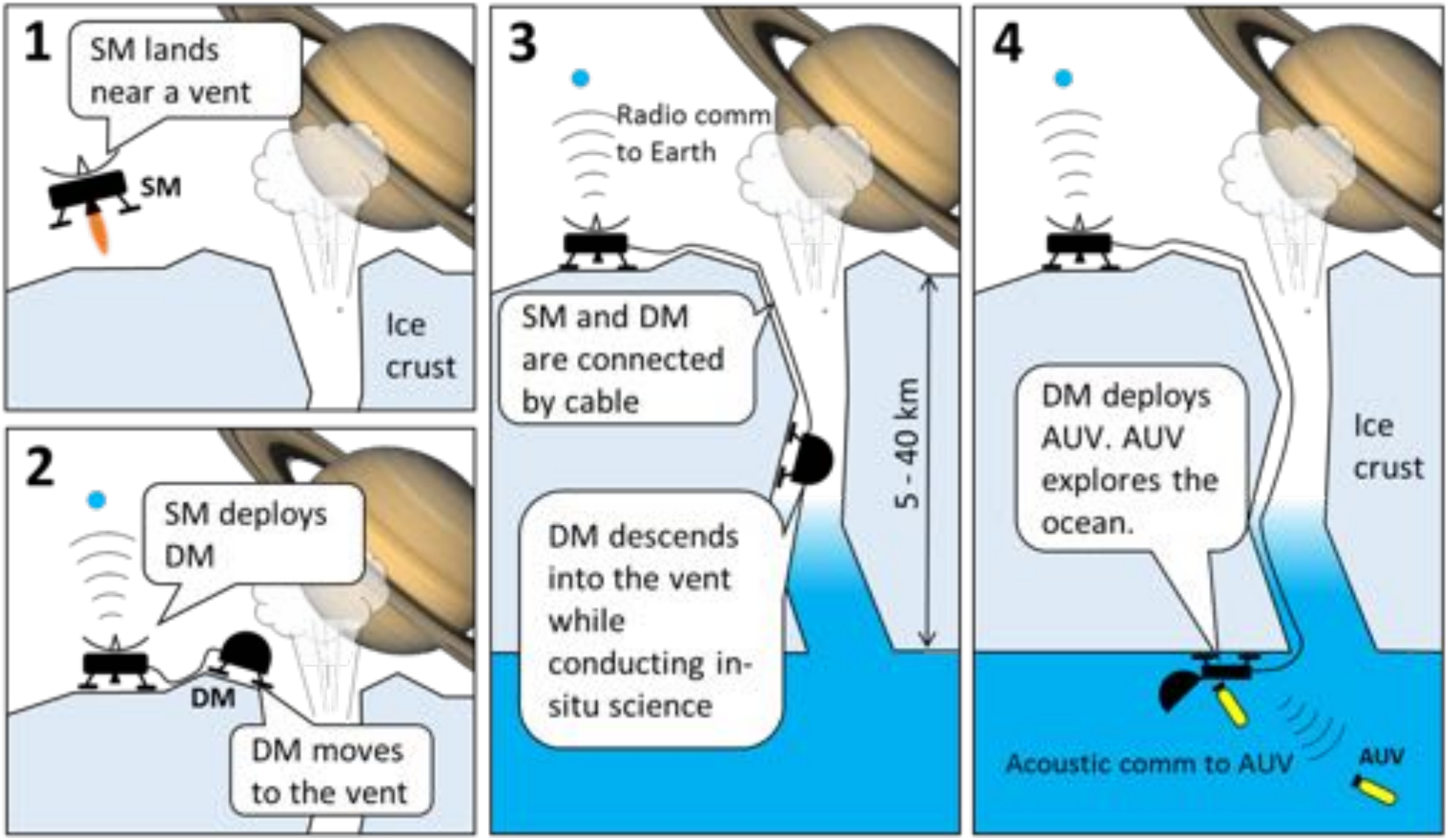
Europa Lander



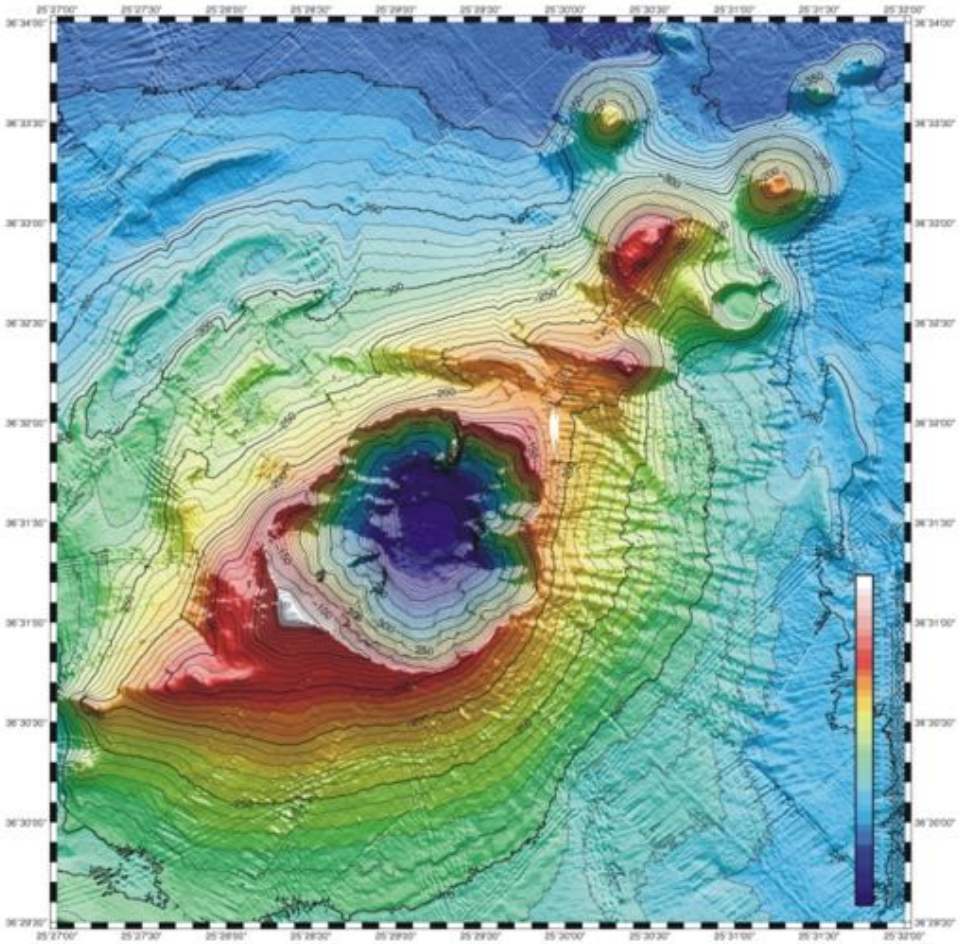
Europa Hydrobot Concept

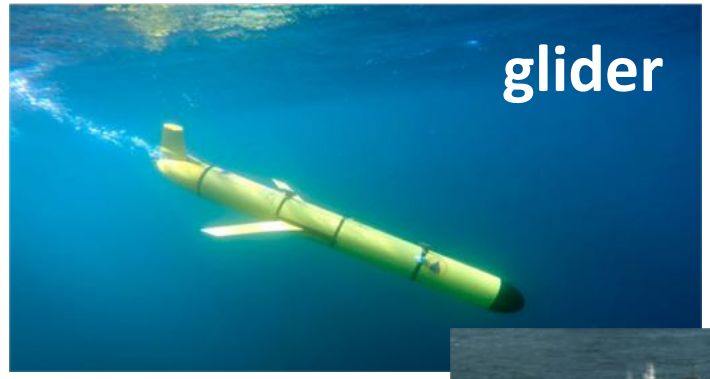
NIAC: Journey to the Center of Icy Moons

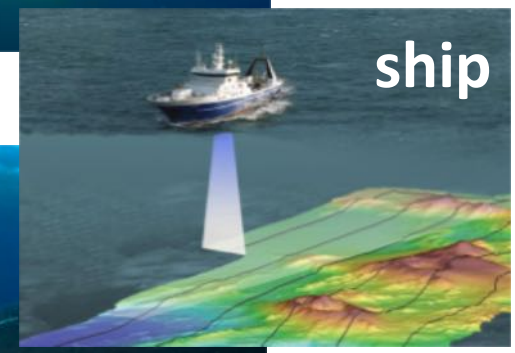
Masahiro Ono (PI)

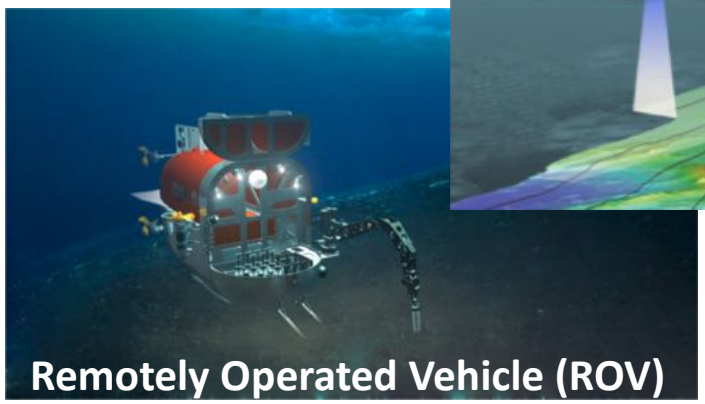


Spring 2019: Mission to the Kolumbo Volcano







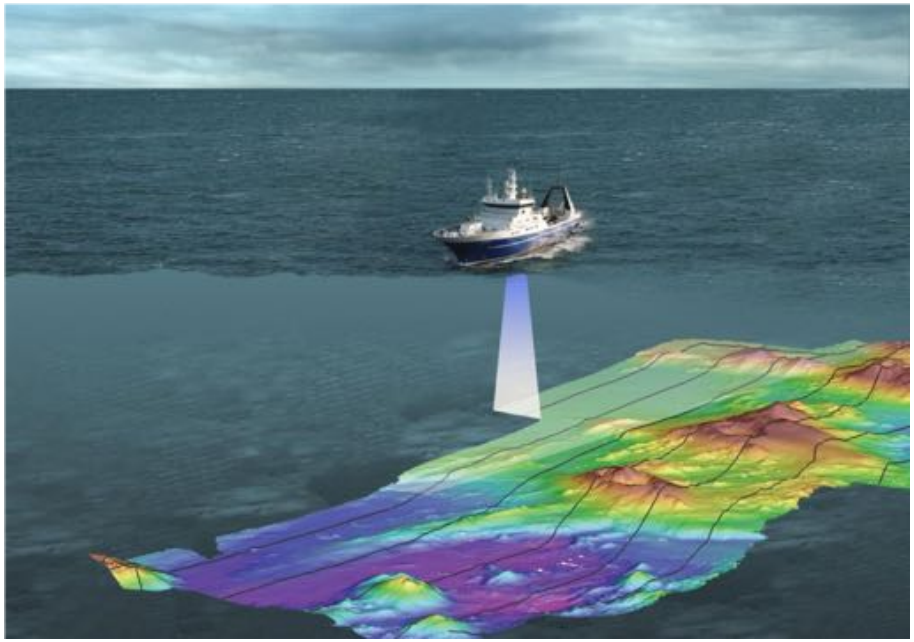


Kolumbo Deep-Sea Volcano near Santorini, Greece

Team: WHOI, MIT, AFRC, U. Michigan

Stage 1: Ship Survey

- Map seafloor of volcano.
- Analyze multi-beam data.

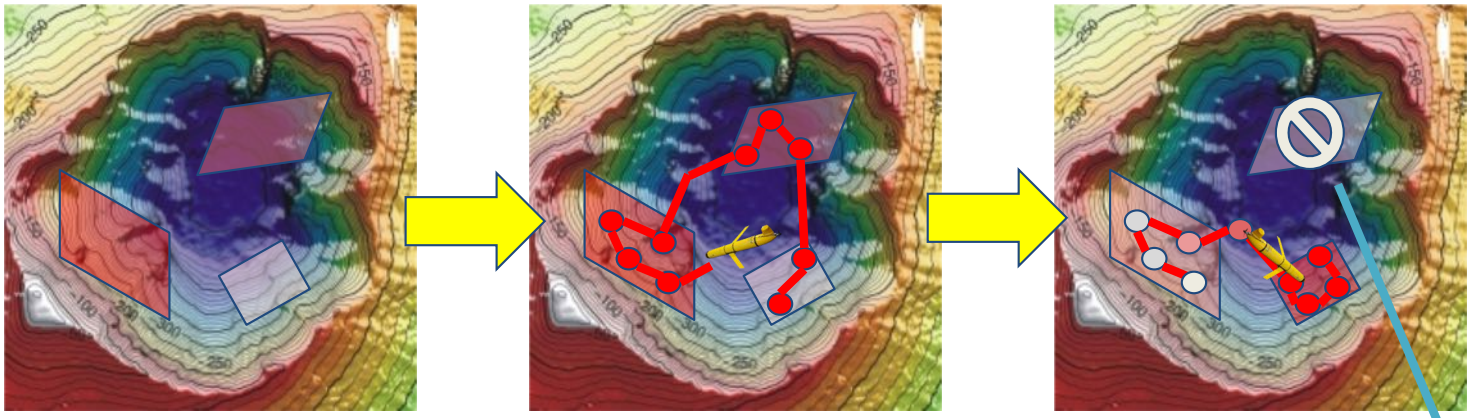


Output:

- Regions of interest based on multi-beam.

Stage 2: Adaptive Search with Glider

Glider adaptively plans sequences of visits that maximize expected information gain, while meeting crew constraints and while bounding risk.



"Prior" for interesting regions as decided by the scientists based on surface vessel data

Initial offline plan

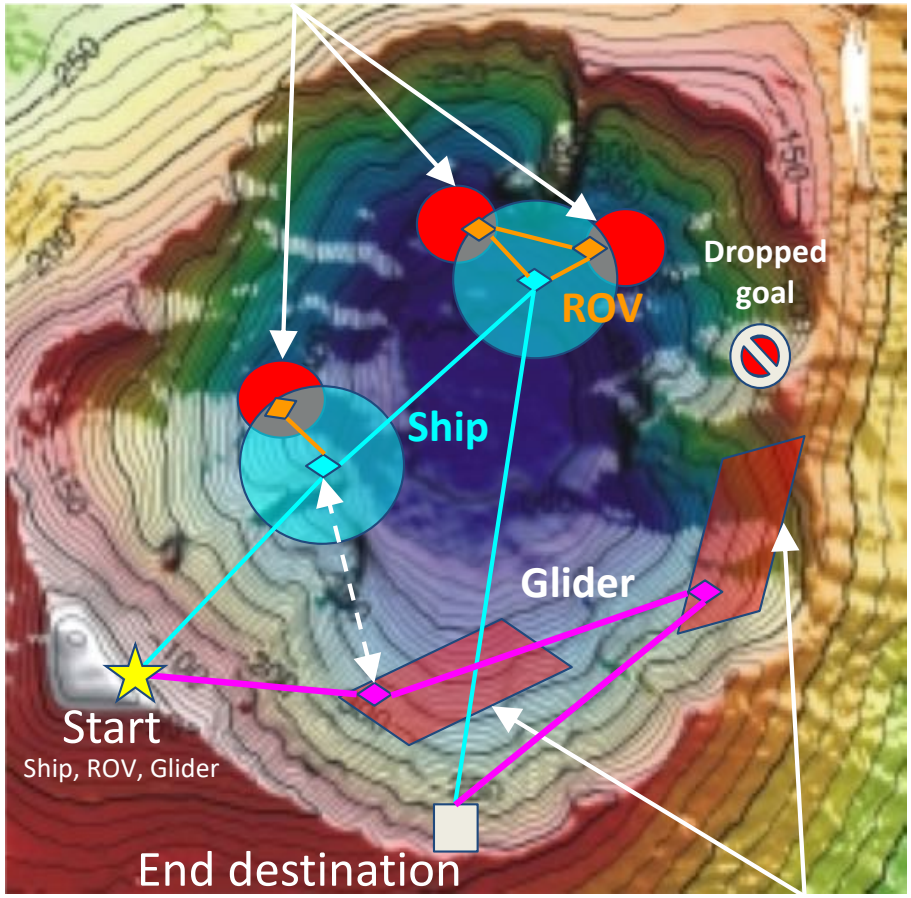
Online adaptation

Belief updated .
no time left to explore
less promising region

Stage 3: ROV Sample Return + Glider Adaptive Search

ROV inspects regions and collects samples, while the Glider simultaneously explores.

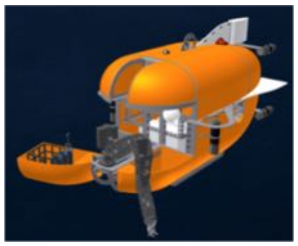
Desired ROV sampling locations



Areas of most interest for glider

Maximize science reward while:

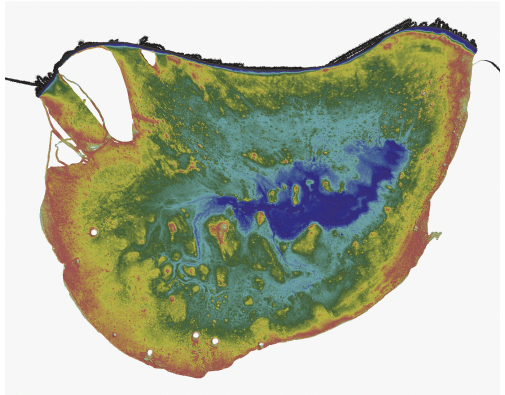
- adapting science from latest observations,
- ensuring safety,
- respecting range constraints, and
  - Ship-ROV < 100-200m
  - Ship-Glider comm range: 2-3km
- respecting resource and time constraints.
  - ROV battery (~6-8 hours).

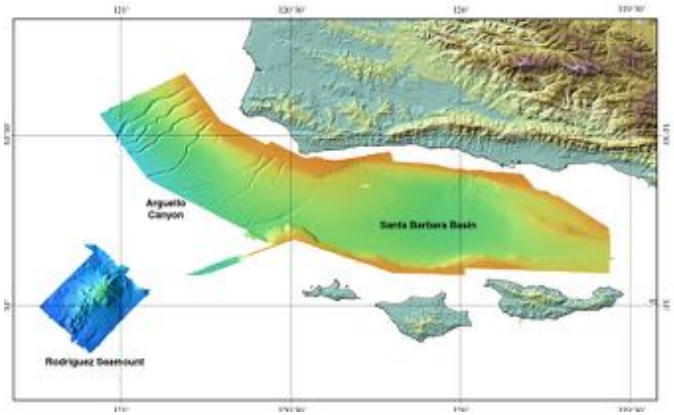


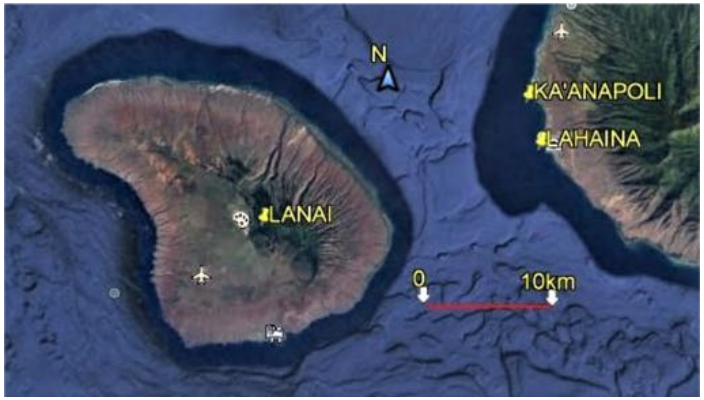
Who needs help with planning?

- The Manipulator } Chekov
- The Vehicles (ROV, Glider, Ship) } Sulu
- The Flotilla (ROV, Glider, Ship) } Scotty
- The Scientists } Spock
- The Chief Engineer } Bones
- The Bosan and Deck Hands
- The Chief Scientist,
- The Captain
- The Bridge Officer

} Kirk, Uhura, Scotty

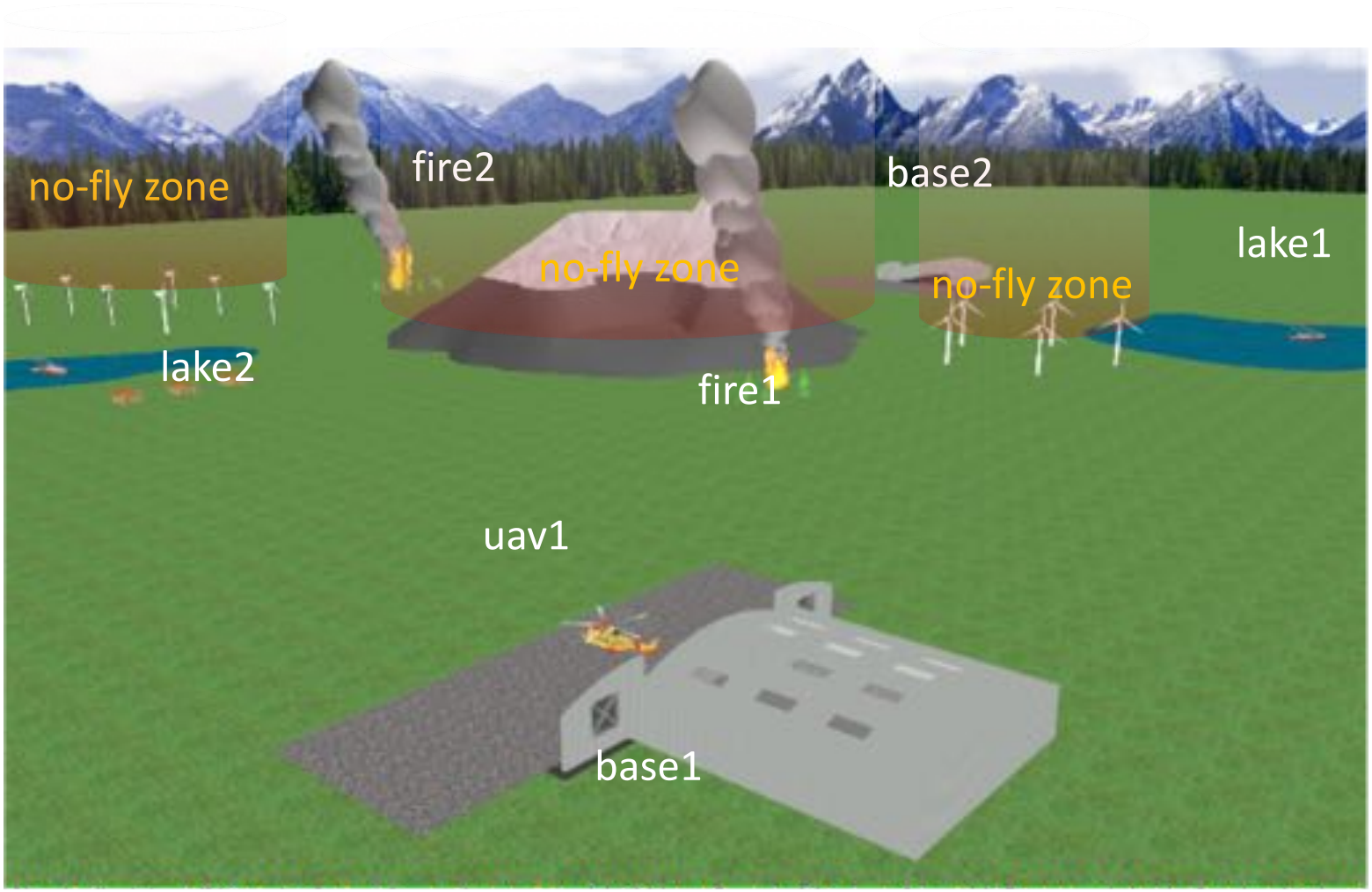






- Mar 2015: Scott's Reef
- Sept 2016: Santa Barbara
- Oct 2016: Cape Cod
- Jan. 2018: Hawaii
- Dec. 2018: Costa Rica
- June 2019: Santorini
- June 2021: Alaska

Programming These Guys



Traditional Imperative Programs:

Specify each action and waypoint (the usual programmatic way)

```
class Main{
  UAV uav1;
  Lake lake1;
  Lake lake2;
  Fire fire1;
  Fire fire2;
  // constructor
  Main (){
      uav1 = new UAV();
      uav1.location= base_1_location;
      uav1.flying = no;
      uav1.loaded = no;

      lake1 = new Lake();
      lake1.location = lake_1_location;

      lake2 = new Lake();
      lake2.location = lake_2_location;

      fire1 = new Fire();
      fire1.location = fire_1_location;
      fire1 = high;

      fire2 = new Fire();
      fire2.location = fire_2_location;
      fire2 = high;
  }
  // "main" method
 method run() {
    sequence{
      uav1.takeoff();
      uav1.fly(base_1_location,lake_2_location);
      uav1.load_water(lake2);
      uav1.fly(lake_2_location,fire_2_location);
      uav1.drop_water_high_altitude(fire2);
      … <13 additional activities> …
      uav1.land();
     }
   }
}
```

These are the actions the UAV can take.

```
class UAV {
  Roadmap location;
  Boolean flying;
  Boolean loaded;

  primitive method takeoff()
    flying == no => flying == yes;

  primitive method land()
    flying == yes => flying == no;

  primitive method load_water(Lake lakespot)
    ((flying == yes) && (loaded == no)
      && (lakespot.location == location)) => loaded == yes;

  primitive method drop_water_high_altiture(Fire firespot)
    ((flying == yes) && (loaded == yes)
     && (firespot.location == location) && (firespot == high))
     => ((loaded == no) && (firespot == medium));

  primitive method drop_water_low_altiture(Fire firespot)
    ((flying == yes) && (loaded == yes)
     && (firespot.location == location) && (firespot == medium))
     => ((loaded == no) && (firespot == out));

  #MOTION_PRIMITIVES(location, fly, flying==yes)
}
```

A program specifies the exact sequence of actions.

State-based Programs:

Specify the goal states, let the computer plan the activities.

```
class Main{
  UAV uav1;
  Lake lake1;
  Lake lake2;
  Fire fire1;
  Fire fire2;
 // constructor
  Main (){
      uav1 = new UAV();
      uav1.location= base_1_location;
      uav1.flying = no;
      uav1.loaded = no;

      lake1 = new Lake();
      lake1.location = lake_1_location;

      lake2 = new Lake();
      lake2.location = lake_2_location;

      fire1 = new Fire();
      fire1.location = fire_1_location;
      fire1 = high;

      fire2 = new Fire();
      fire2.location = fire_2_location;
      fire2 = high;
  }

 // "main" method
 method run() {
    sequence{
       (fire1 == out);
       (fire2 == out);
       (uav1.flying == no &&
       uav1.location == base_1_location);
    }
  }
}
```

A program specifies the desired states.

These are how UAV actions behave.

```
class UAV {
  Roadmap location;
  Boolean flying;
  Boolean loaded;

  primitive method takeoff()
    flying == no => flying == yes;

  primitive method land()
    flying == yes => flying == no;

  primitive method load_water(Lake lakespot)
    ((flying == yes) && (loaded == no)
     && (lakespot.location == location)) => loaded == yes;

  primitive method drop_water_high_altiture(Fire firespot)
    ((flying == yes) && (loaded == yes)
     && (firespot.location == location) && (firespot == high))
     => ((loaded == no) && (firespot == medium));

  primitive method drop_water_low_altiture(Fire firespot)
    ((flying == yes) && (loaded == yes)
     && (firespot.location == location) && (firespot == medium))
     => ((loaded == no) && (firespot == out));

  #MOTION_PRIMITIVES(location, fly, flying==yes)
}
```

State-based Programs:

Specify the goal states, let the computer plan the activities.

```
class Main{
  UAV uav1;
  Lake lake1;
  Lake lake2;
  Fire fire1;
  Fire fire2;
 // constructor
  Main (){
      uav1 = new UAV();
      uav1.location= base_1_location;
      uav1.flying = no;
      uav1.loaded = no;

      lake1 = new Lake();
      lake1.location = lake_1_location;

      lake2 = new Lake();
      lake2.location = lake_2_location;

      fire1 = new Fire();
      fire1.location = fire_1_location;
      fire1 = high;

      fire2 = new Fire();
      fire2.location = fire_2_location;
      fire2 = high;
  }

 // "main" method
 method run() {
    sequence{
       (fire1 == out);
       (fire2 == out);
       (uav1.flying == no &&
       uav1.location == base_1_location);
    }
  }
}
```

A program specifies the desired states.

These are how UAV actions behave.

```
class UAV {
  Roadmap location;
  Boolean flying;
  Boolean loaded;

  primitive method takeoff()
    flying == no => flying == yes;

  primitive method land()
    flying == yes => flying == no;

  primitive method load_water(Lake lakespot)
    ((flying == yes) && (loaded == no)
     && (lakespot.location == location)) => loaded == yes;

  primitive method drop_water_high_altiture(Fire firespot)
    ((flying == yes) && (loaded == yes)
     && (firespot.location == location) && (firespot == high))
     => ((loaded == no) && (firespot == medium));

  primitive method drop_water_low_altiture(Fire firespot)
    ((flying == yes) && (loaded == yes)
     && (firespot.location == location) && (firespot == medium))
     => ((loaded == no) && (firespot == out));

  #MOTION_PRIMITIVES(location, fly, flying==yes)
}
```

State-based Programs:

Specify the goal states, let the computer plan the activities.

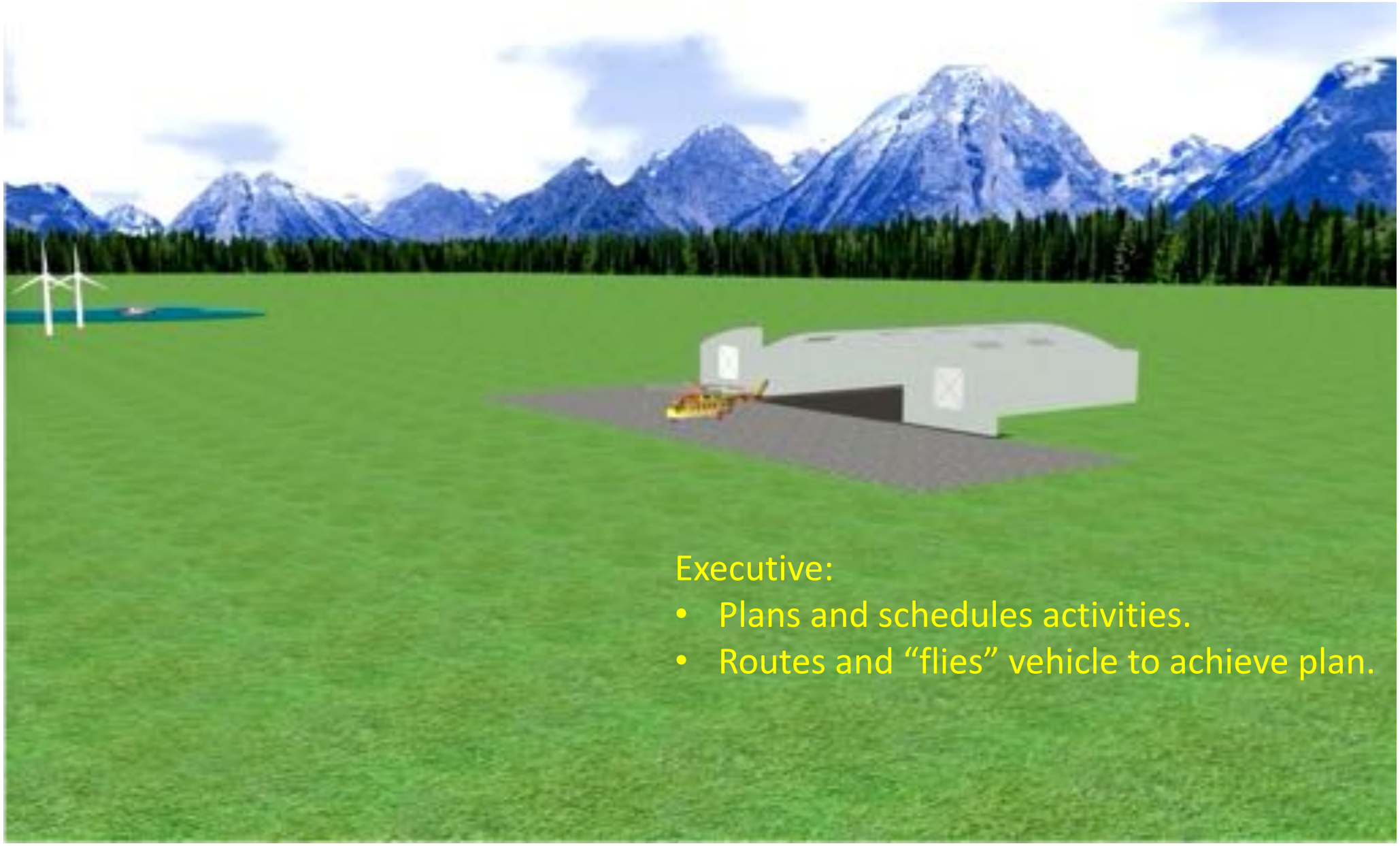
```
class Main{
  UAV uav1;
  Lake lake1;
  Lake lake2;
  Fire fire1;
  Fire fire2;
 // constructor
  Main (){
      uav1 = new UAV();
      uav1.location= base_1_location;
      uav1.flying = no;
      uav1.loaded = no;

      lake1 = new Lake();
      lake1.location = lake_1_location;

      lake2 = new Lake();
      lake2.location = lake_2_location;

      fire1 = new Fire();
      fire1.location = fire_1_location;
      fire1 = high;

      fire2 = new Fire();
      fire2.location = fire_2_location;
      fire2 = high;
  }

 // "main" method
 method run() {
    sequence{
       (fire1 == out);
       (fire2 == out);
       (uav1.flying == no &&
       uav1.location == base_1_location);
    }
  }
}
```

A program specifies the desired states.

These are how UAV actions behave.

```
class UAV {
  Roadmap location;
  Boolean flying;
  Boolean loaded;

  primitive method takeoff()
    flying == no => flying == yes;

  primitive method land()
    flying == yes => flying == no;

  primitive method load_water(Lake lakespot)
    ((flying == yes) && (loaded == no)
      && (lakespot.location == location)) => loaded == yes;

  primitive method drop_water_high_altiture(Fire firespot)
    ((flying == yes) && (loaded == yes)
     && (firespot.location == location) && (firespot == high))
     => ((loaded == no) && (firespot == medium));

  primitive method drop_water_low_altiture(Fire firespot)
    ((flying == yes) && (loaded == yes)
     && (firespot.location == location) && (firespot == medium))
     => ((loaded == no) && (firespot == out));

  #MOTION_PRIMITIVES(location, fly, flying==yes)
}
```

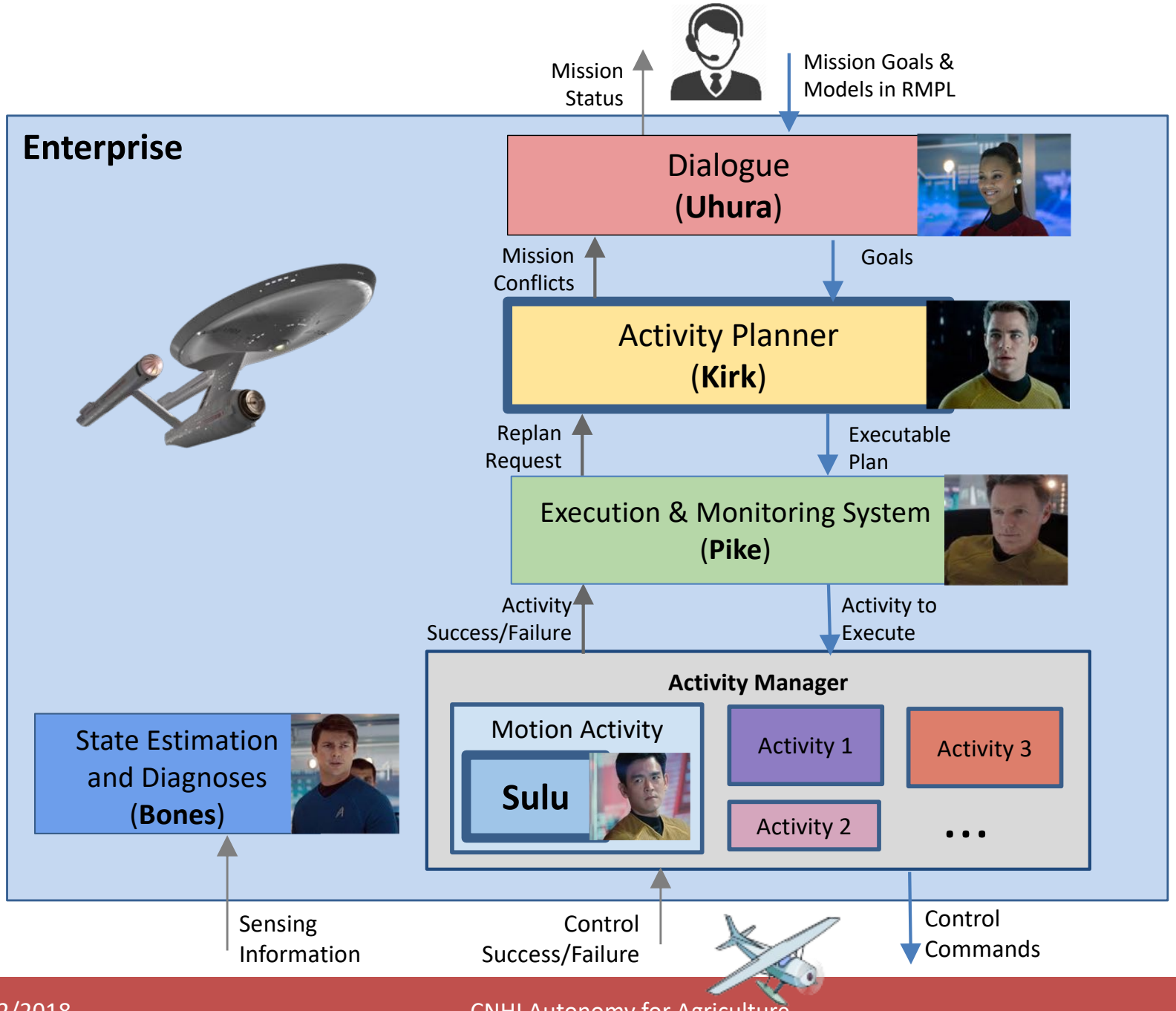
In Simulation



Executive:

- Plans and schedules activities.
- Routes and “flies” vehicle to achieve plan.

Enterprise: A Goal-directed Executive

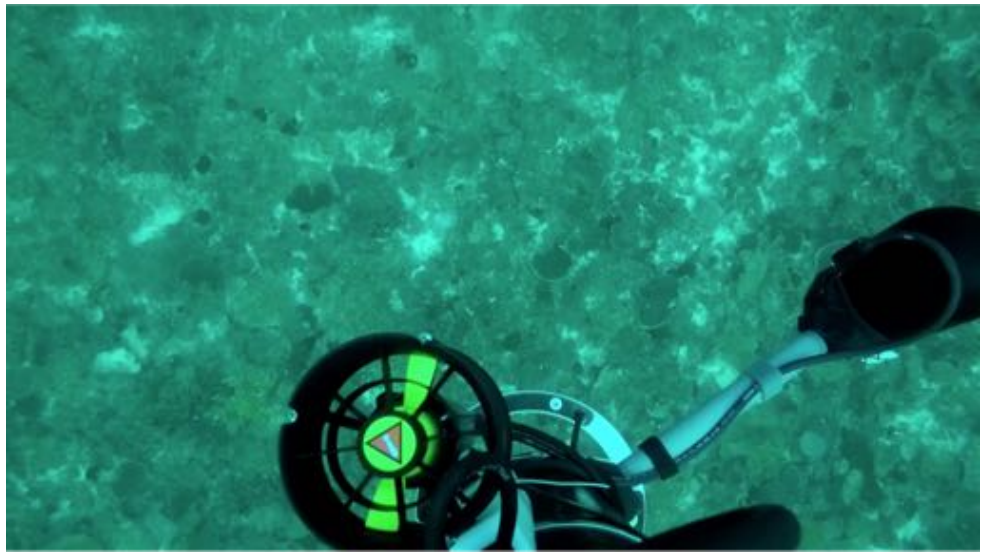


Uhura helps negotiate . . .

Outline

- Diving into the Real World
- Risk-bounded Planning as Optimization and Constraint Programming
- Generative Task and Motion Planning with Heuristic Forward Search
- Risk-bounded Planning by adding And / OR (AO*) Search

Observing Systems Often Confront Risk



Operator: Specifies goals and acceptable risk.
Executive: Decides how to use risk effectively.

"Map the boundary of the algal bloom, and return to the base in an hour. Avoid collision. Here is a map."

Qualitative State Plan

Risk that bloom not mapped < 5%

Risk of collision < .1%

Model-based Executive

Observations

Commands

Plant

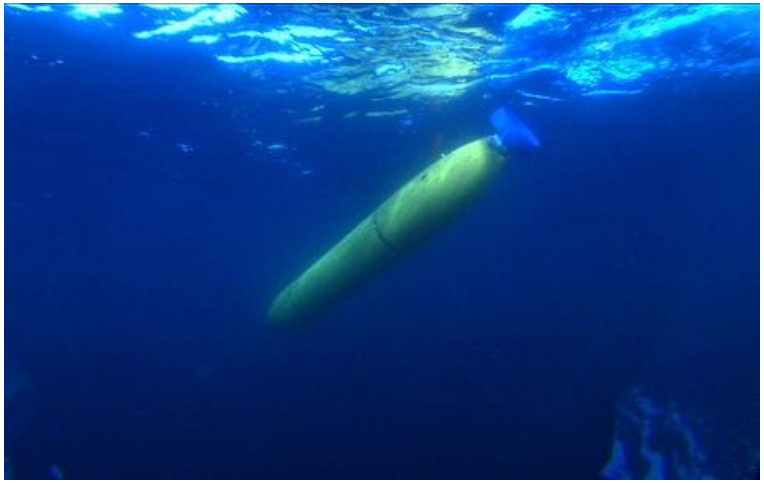
[Blackmore PhD 06]

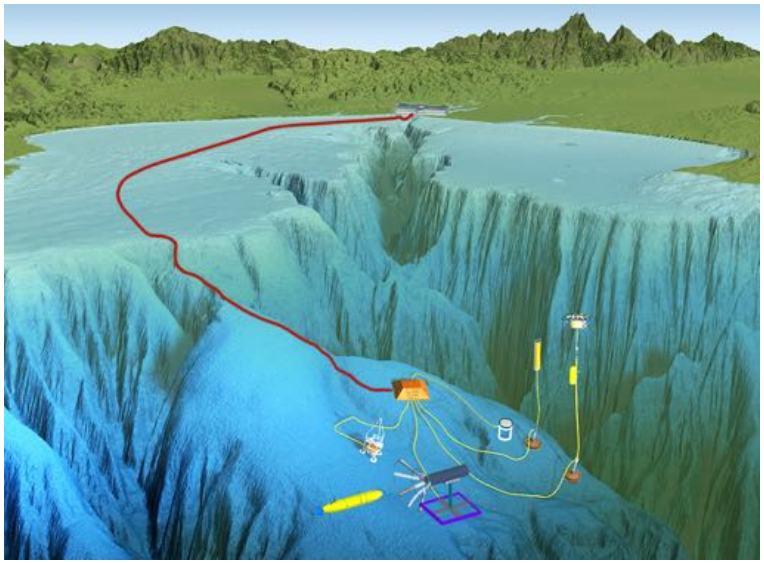
[Ono & Williams, AAAI 08]

Outline

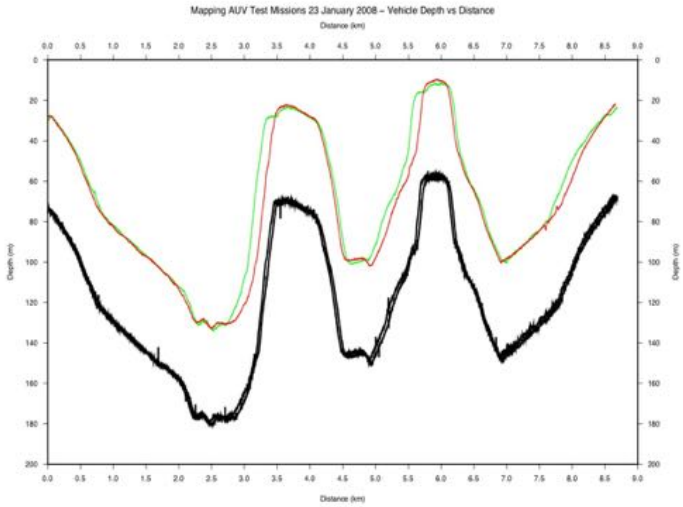
- Diving into the Real World
- Risk-bounded Planning as Optimization and Constraint Programming
  - Risk-bounded Motion Planning
  - Risk-bounded Task Planning
- Generative Task and Motion Planning with Heuristic Forward Search
- Risk-bounded Planning by adding And / OR (AO*) Search

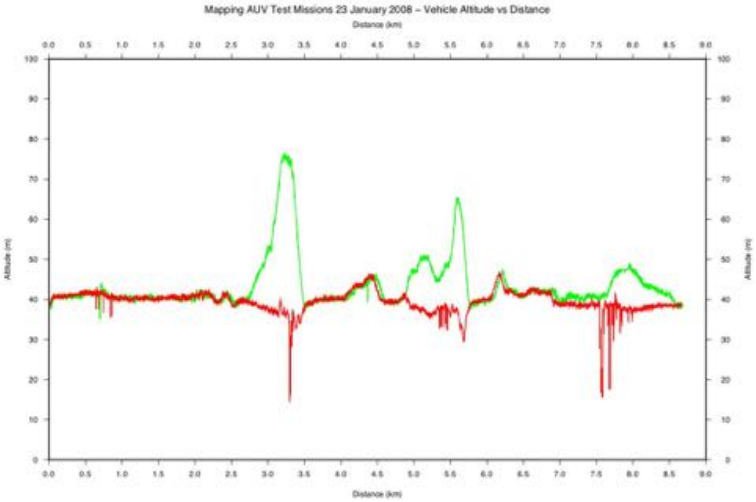
Jan. 23rd, 2008: Depth Navigation for Bathymetric Mapping





Courtesy MBARI

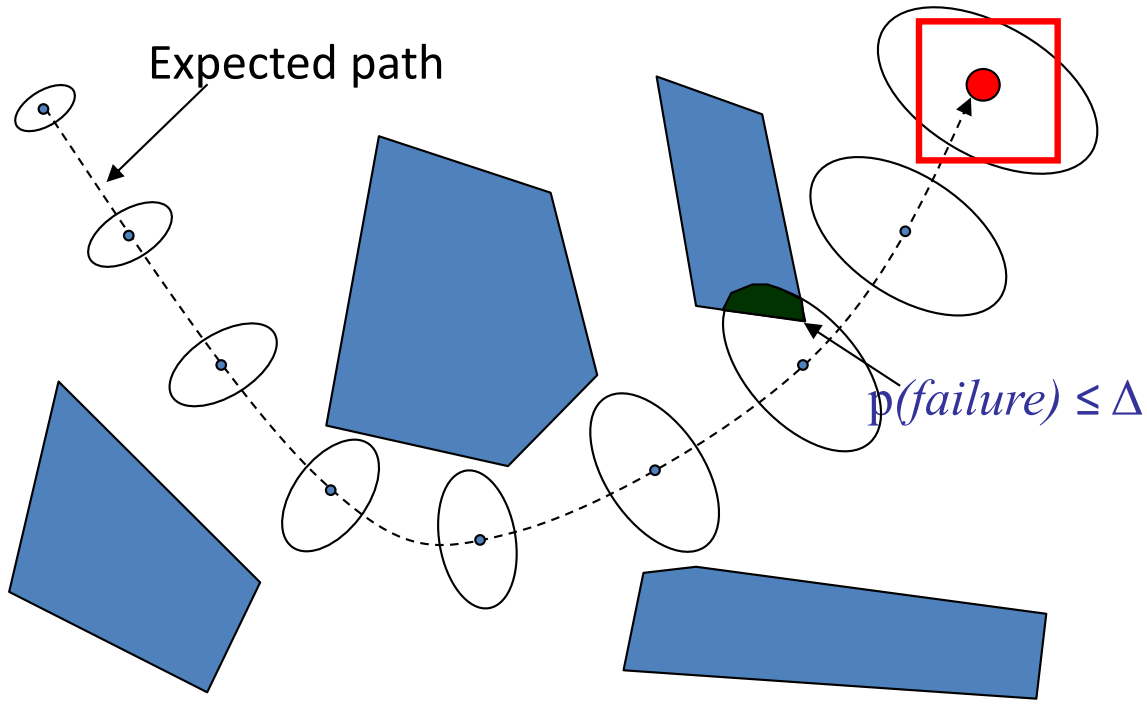




pSulu: A "Risk-bounded" Motion Planner

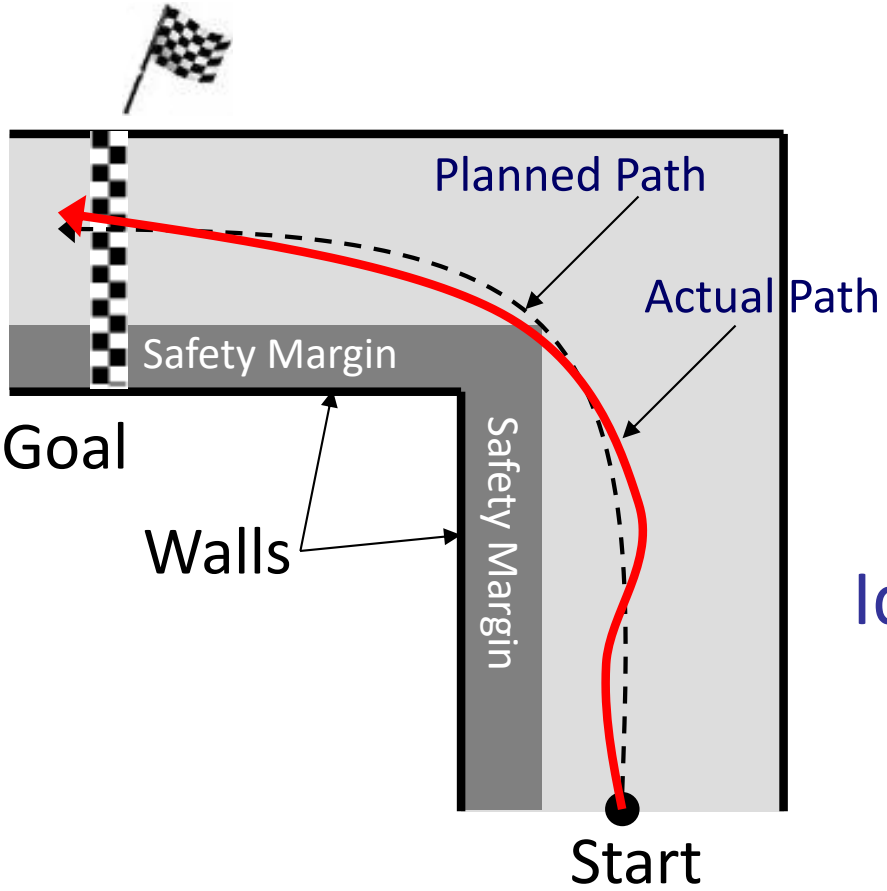
– "Plan optimal path to goal such that

p(failure) ≤ Δ."



– Called a *Chance Constraint (CC).*

How do We Perform Risk-bounded Planning Quickly?



Problem

Find fastest path to the goal, while limiting the probability of crash to 0.1%

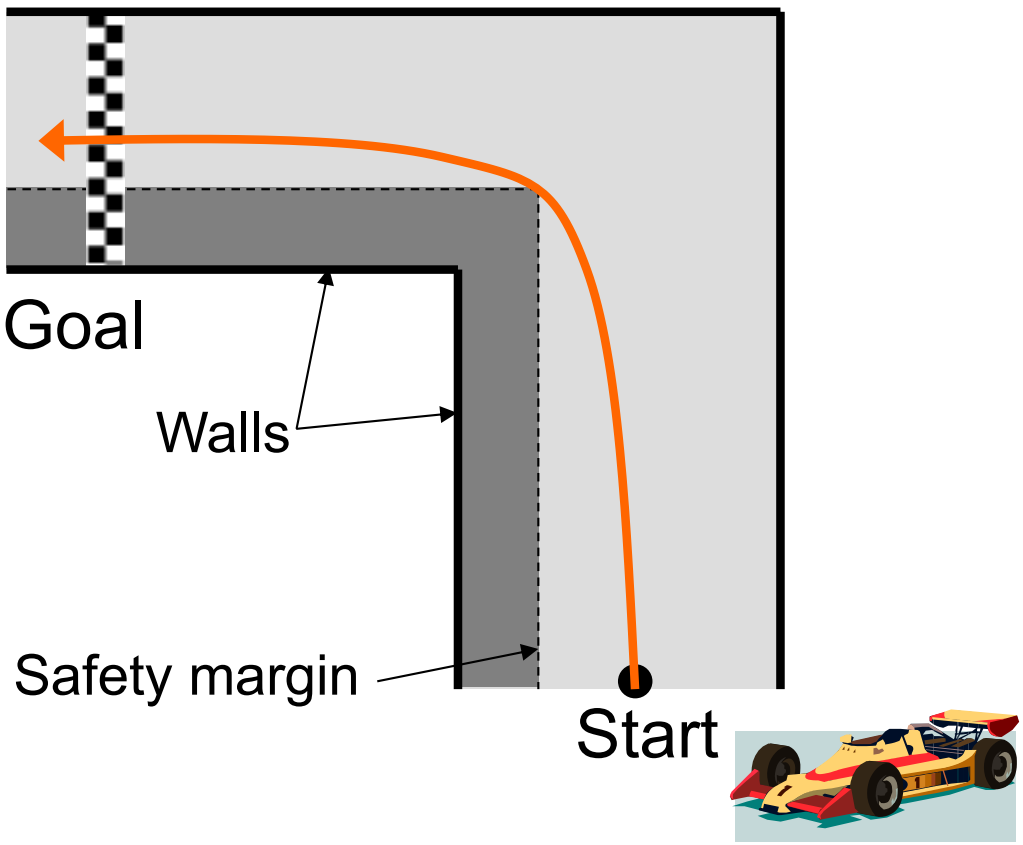
Risk bound

Idea:

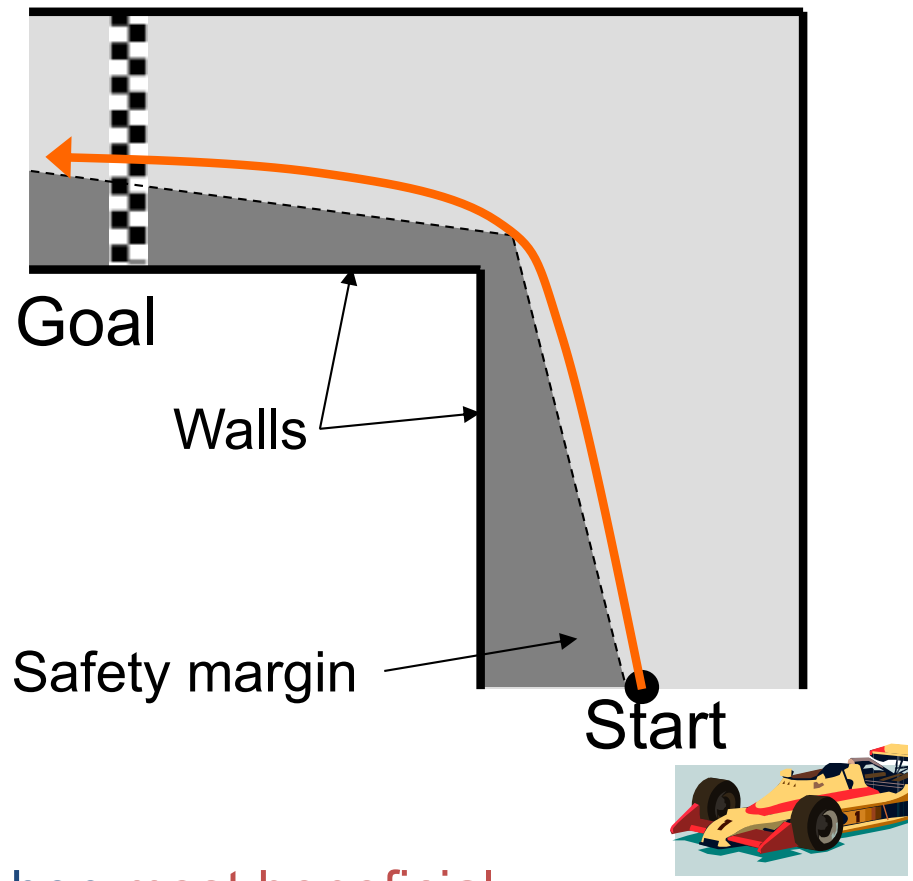
1. Create safety margin that satisfies the risk bound from start to goal.
2. Reduce to simpler, deterministic optimization problem.

What is the Best Safety Margin?

(a) Uniform width safety margin



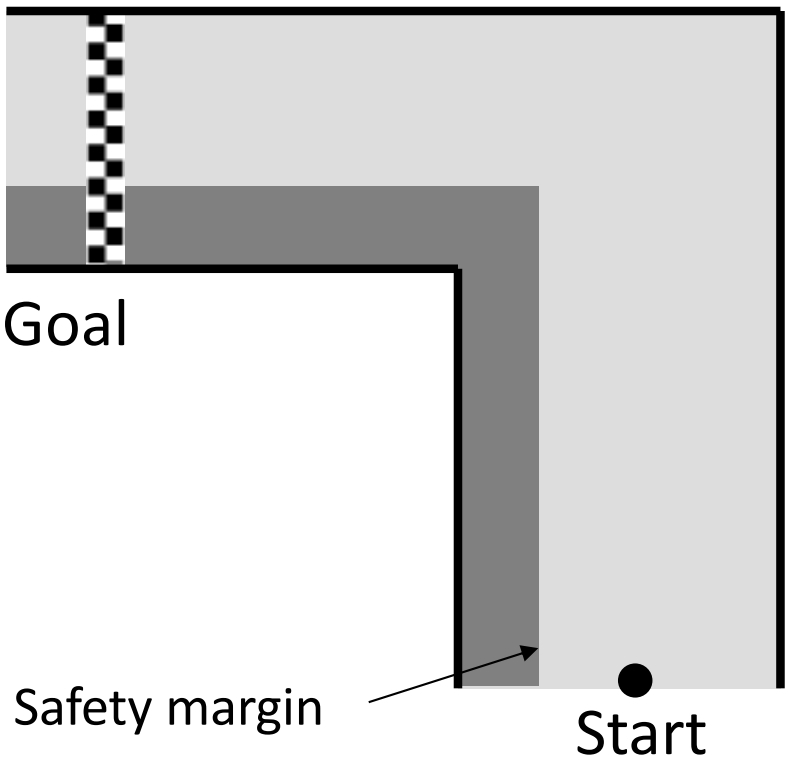
(b) Uneven width safety margin



(b) Produce a better path → by taking risk when most beneficial.

To find best safety margin: Use **Algorithmic Risk Allocation** [Ono & Williams, AAAI 08]

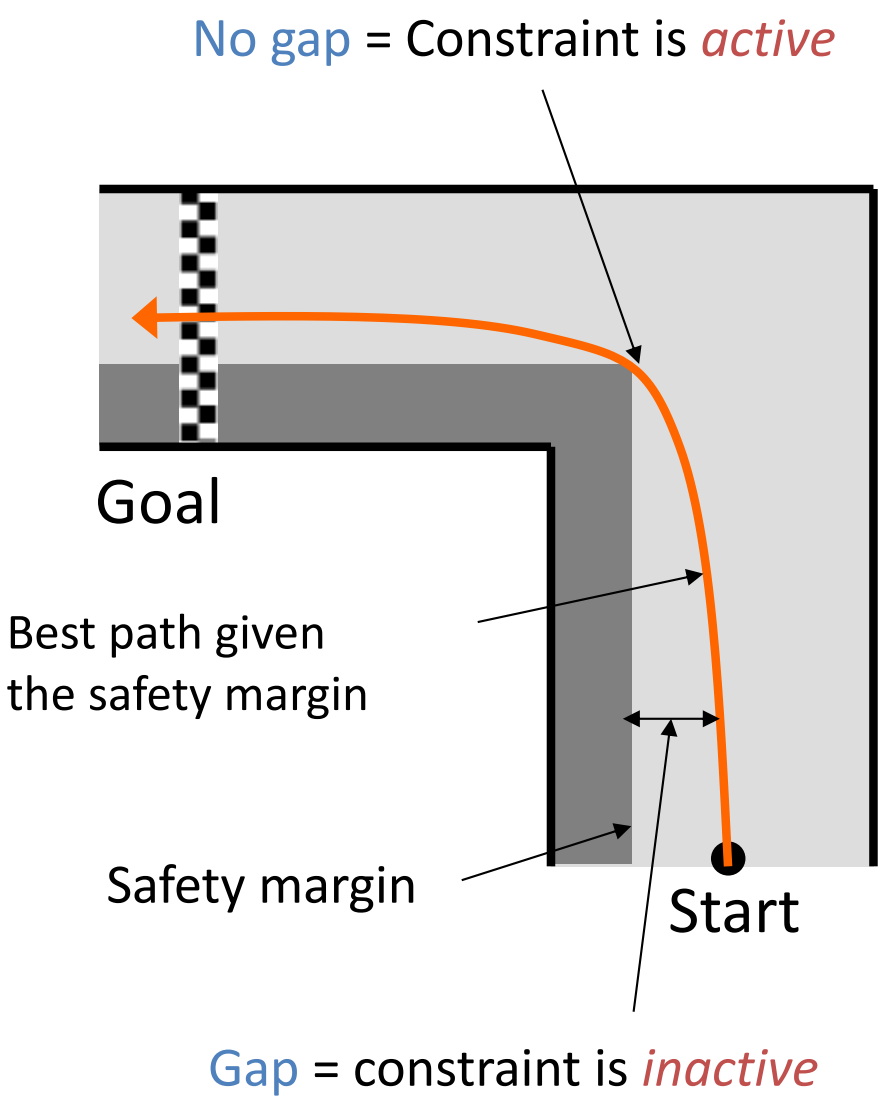
Iterative Risk Allocation Algorithm



Algorithm IRA

1 Initialize with arbitrary risk allocation
2 Loop
3 Compute the best available path given the current risk allocation
4 Decrease the risk where the constraint is inactive
5 Increase the risk where the constraint is active
6 End loop

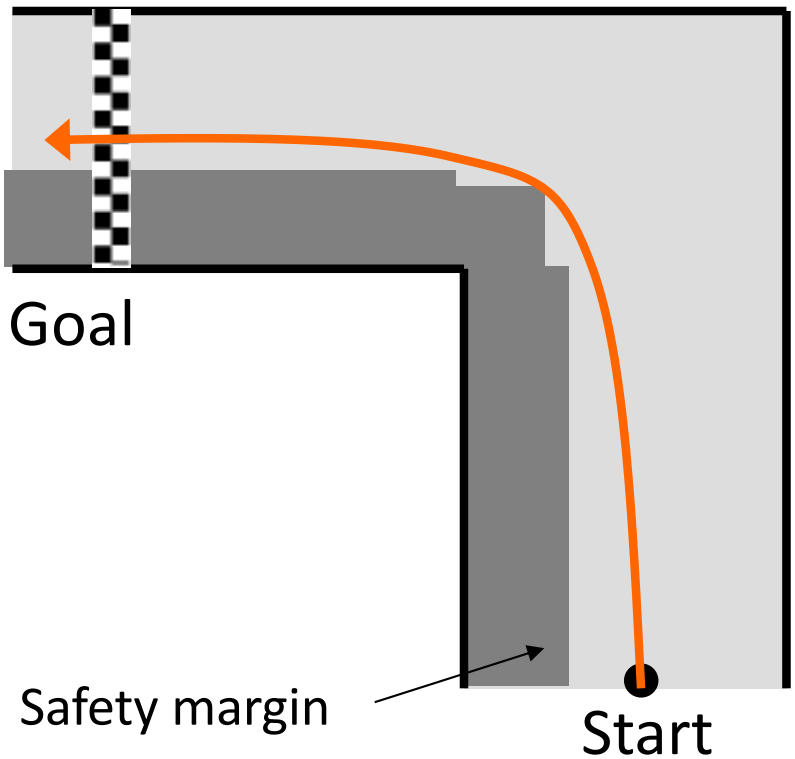
Iterative Risk Allocation Algorithm



Algorithm IRA

1 Initialize with arbitrary risk allocation
2 Loop
3 Compute the best available path given the current risk allocation
4 Decrease the risk where the constraint is inactive
5 Increase the risk where the constraint is active
6 End loop

Iterative Risk Allocation Algorithm

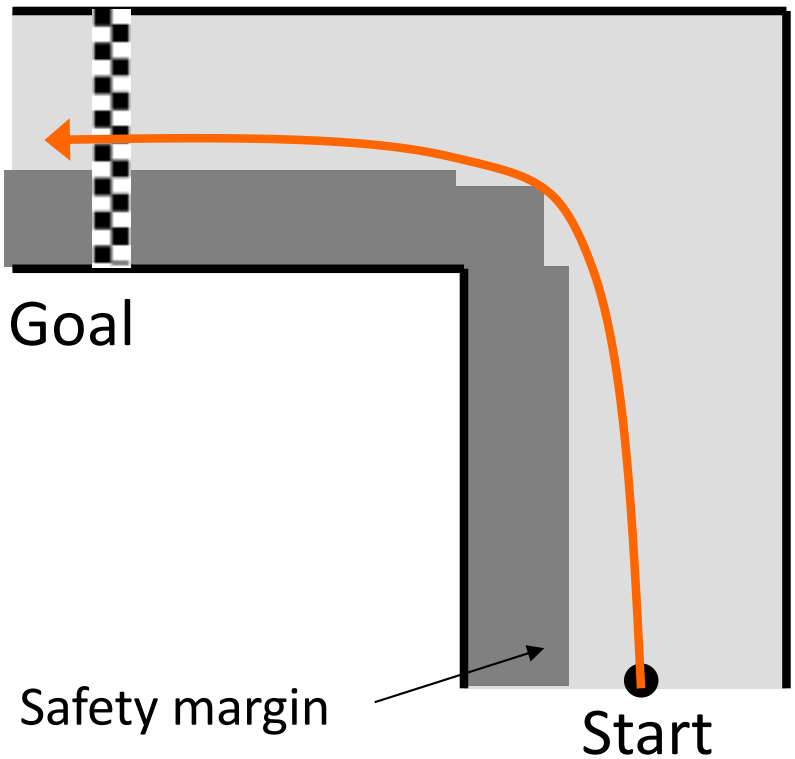


Reallocate risk

Algorithm IRA

1 Initialize with arbitrary risk allocation
2 Loop
3 Compute the best available path given the current risk allocation
4 Decrease the risk where the constraint is inactive
5 Increase the risk where the constraint is active
6 End loop

Iterative Risk Allocation Algorithm

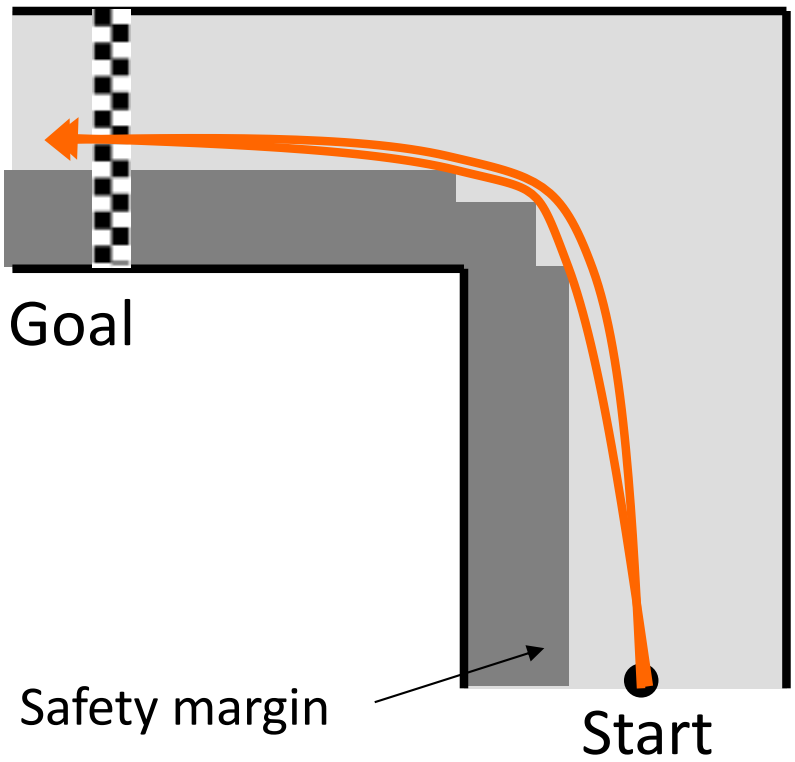


Reallocate risk

Algorithm IRA

1 Initialize with arbitrary risk allocation
2 Loop
3 Compute the best available path given the current risk allocation
4 Decrease the risk where the constraint is inactive
5 Increase the risk where the constraint is active
6 End loop

Iterative Risk Allocation Algorithm



Repeat

Algorithm IRA

1 Initialize with arbitrary risk allocation

2 Loop

3 Compute the best available path given the current risk allocation

4 Decrease the risk where the constraint is inactive

5 Increase the risk where the constraint is active

6 End loop

. . . or do the Math to Create a Convex Problem with Guaranteed Convergence

$$\min_{u_{1:T} \in \mathbf{U}^T} J(u_{1:T})$$

$$s.t. \quad \bigwedge_{t=0}^{T-1} x_{t+1} = Ax_t + Bu_t + w_t$$

Stochastic dynamics

$$w_t \sim N(0, \Sigma_t)$$

$$x_0 \sim N(\bar{x}_0, \Sigma_{x,0})$$

$$\Pr\left[\bigwedge_{t=1}^{T} \bigwedge_{i=1}^{N} h_t^{iT} x_t \leq g_t^i \right] \geq 1 - \Delta$$

Chance constraint

. . . or do the Math to Create a Convex Problem with Guaranteed Convergence

$$\min_{\boldsymbol{\delta}} \min_{u_{1:T} \in \mathbf{U}^T} J(u_{1:T})$$

$$s.t. \quad \bigwedge_{t=0}^{T-1} x_{t+1} = A\bar{x}_t + Bu_t$$

Convex Non-linear Program

$$\bigwedge_{t=1}^{T} \bigwedge_{i=1}^{I} h_t^{iT} \bar{x}_t \leq g_t^i - m_t^i\left(\delta_t^i\right)$$

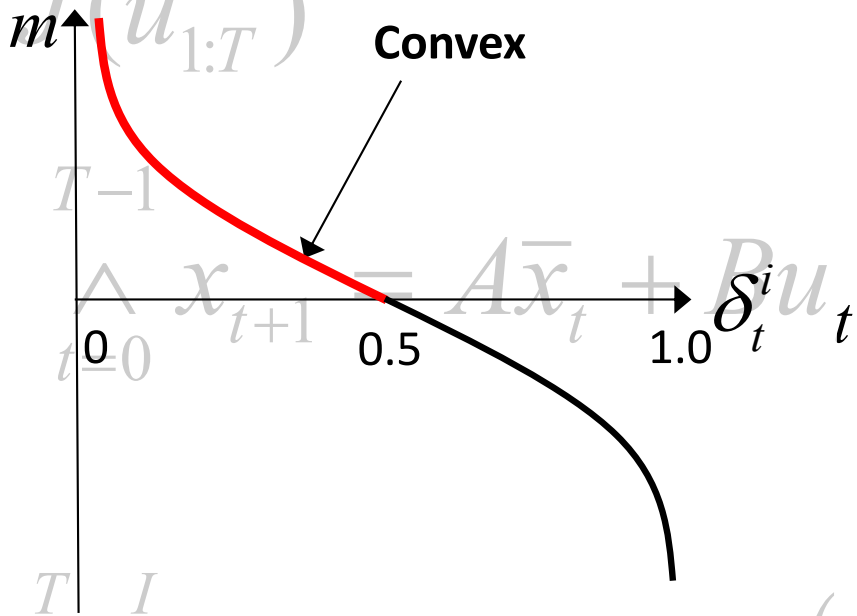
Convex if $\delta < 0.5$

$$\sum_{t,i} \delta_t^i \leq \Delta$$

. . . or do the Math to Create a Convex Problem with Guaranteed Convergence

$$\min_{\boldsymbol{\delta}} \min_{u_{1:T} \in \mathbf{U}^T} I(u_{1:T})$$

$$s.t.$$



Convex
Non-linear
Program

$$\bigwedge_{t=0}^{T-1} x_{t+1} = A\bar{x}_t + Bu_t$$

$$\bigwedge_{t=1}^{T} \bigwedge_{i=1}^{I} h_t^{iT} \bar{x}_t \le g_t^i - m_t^i\left(\delta_t^i\right)$$

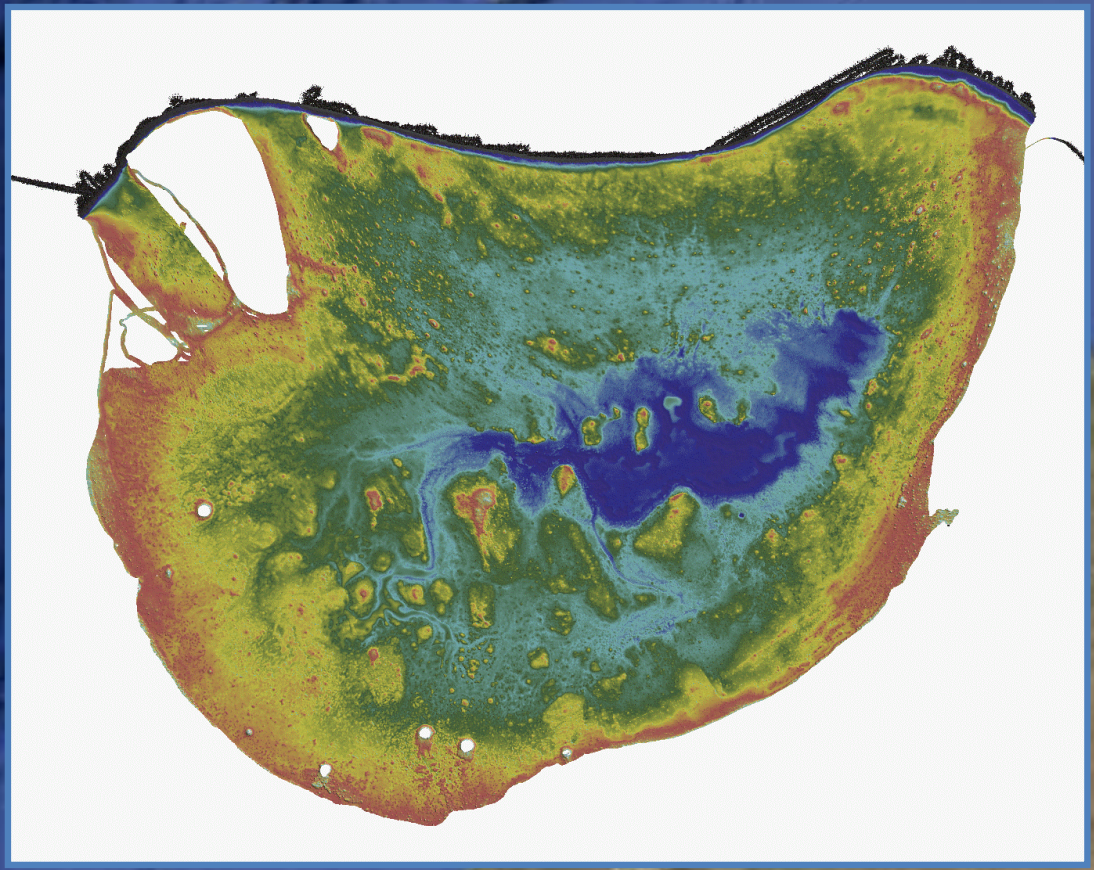
Convex if $\delta < 0.5$

$$\sum_{t,i} \delta_t^i \le \Delta$$

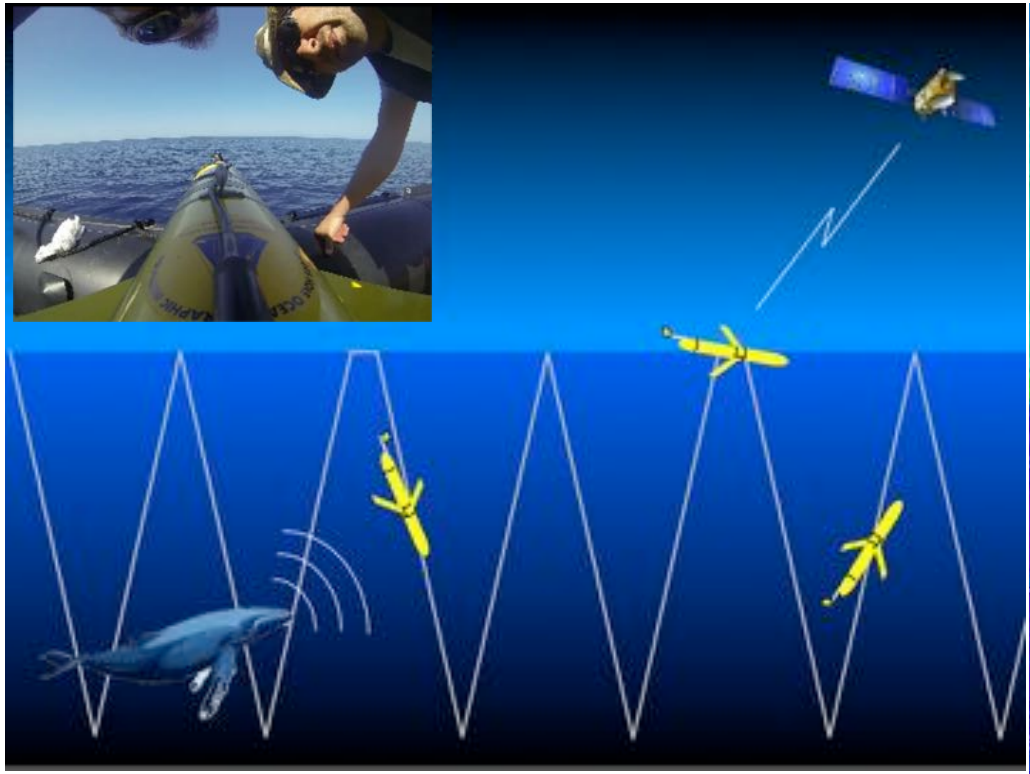
Outline

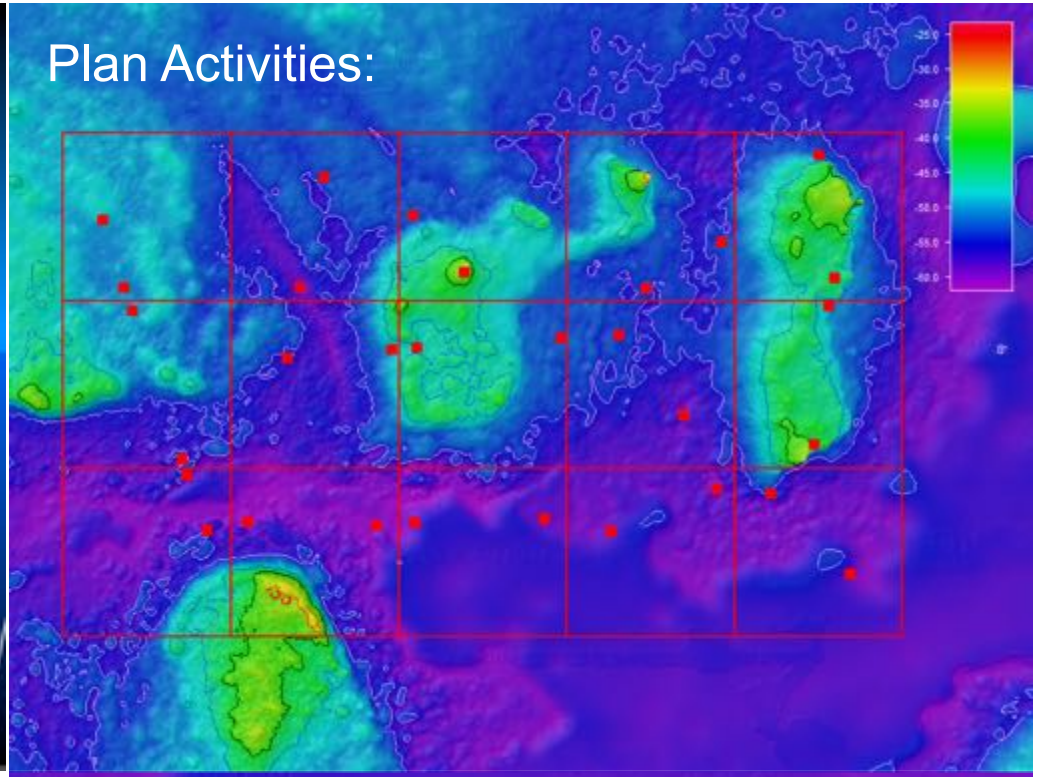
- Diving into the Real World
- Risk-bounded Planning as Optimization and Constraint Programming
  - Risk-bounded Motion Planning
  - Risk-bounded Task Planning
- Generative Task and Motion Planning with Heuristic Forward Search
- Risk-bounded Planning by adding And / OR (AO*) Search

March, 2015: Scott's Reef



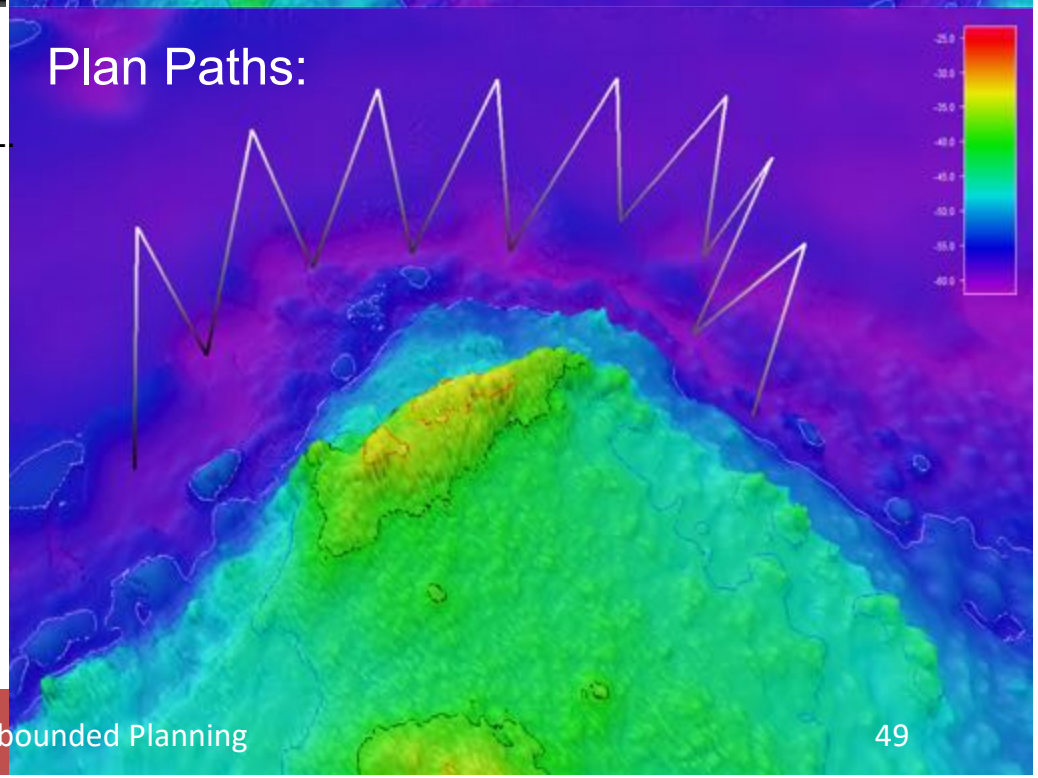
Slocum Glider





Combined Activity and Path Planning Rules:

- TRANSIT THROUGH BOTH GOAL POINTS IN EACH CELL.
- AVOID FIXED AND MOVING HAZARDS.
- MINIMIZE ENERGY EXPENDITURE BY OPTIMIZING DEPTH BAND, LINEAR DISTANCE, AND TIDAL CURRENTS.
- STAY WITHIN 2KM COMMS RANGE OF MOTHER SHIP.



Choices to be made:

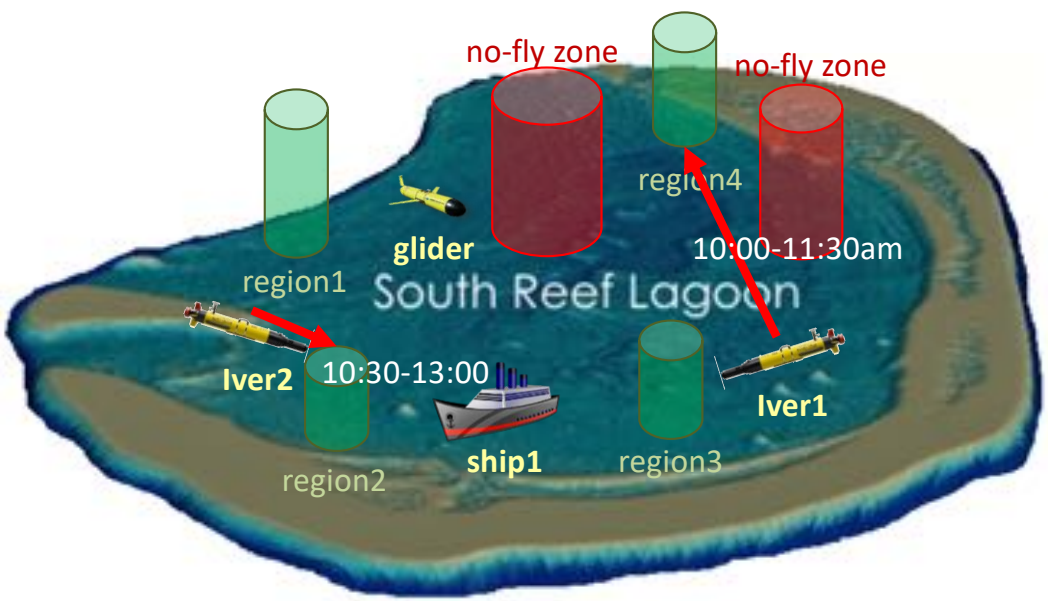
- Which observations to make,
- Order of visits,
- Schedule of arrivals and departures,
- What paths between waypoints,
- Which Inflection points along path.

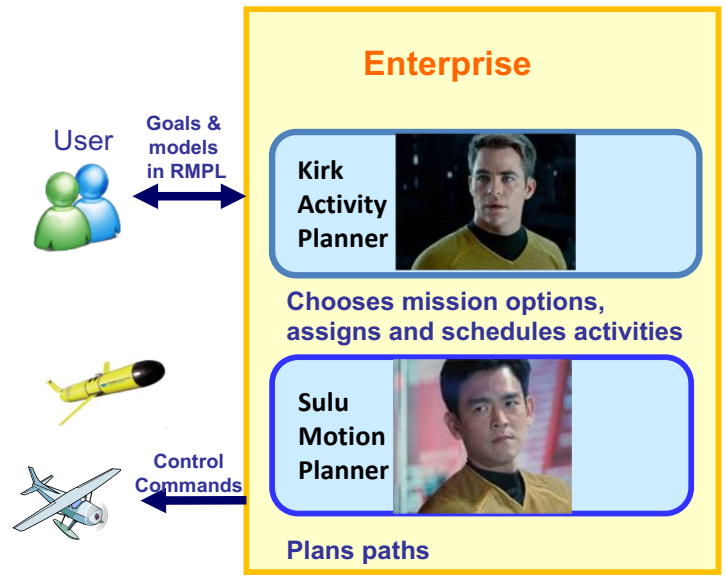
Describe as a Risk-bounded Decision-theoretic Program

Program A ::=
prim_action(args) |
remain_in(R) | start_in(R) | end_in(R) |
[lb, ub] A |
Sequence {A1; A2; ... } |
Parallel {A1; A2; ... } |
Choose { [with reward R1] A1;
[with reward] R2 A2; . . .} |

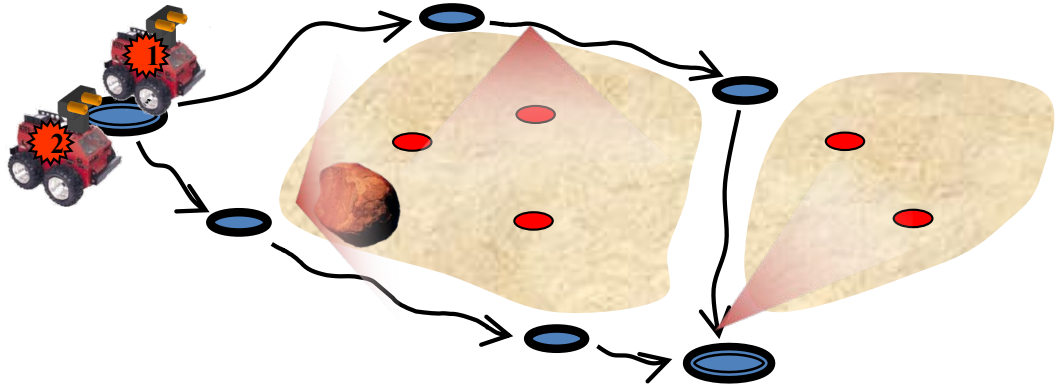
Model:
- Durative PDDL
- $X_{t+1} = Ax_t + Bu_t$

- Topographical Map
- Target Regions
- Currents

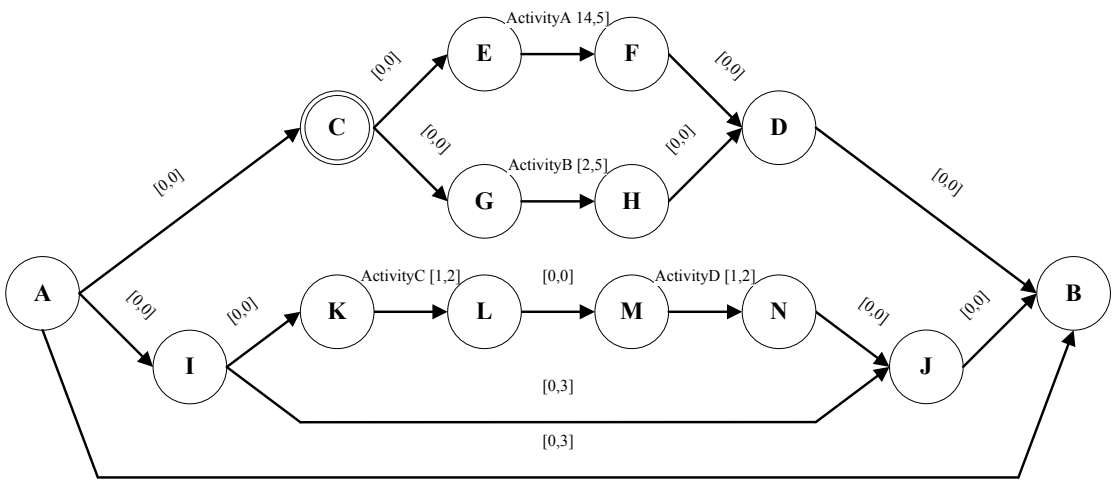




Translate RMPL to a Probabilistic Temporal Plan Network (pTPN)



[Kim, Williams, Abramson, IJCAI01]



Stochastic Constraint Program:

- Uncontrolled durations
- Uncontrolled choices
- . . .

OpSat: Stochastic Constraint Programming with Relaxation

Model: C(X,Y)

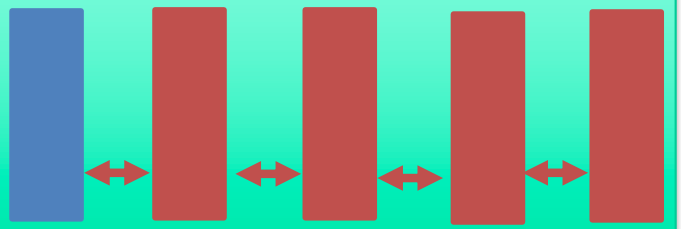
Generate:
Enumerate, best-first or BB, every assignment to X, using tree search

Full assignments to X s.t. C(X,Y) satisfiable

Candidate C_i:
Full assignment to X

Consistent?
(True / False)
Conflicting assignments

Test & Bound:



Stochastic and risk-bounded

Preferred relaxations

Hybrid conflicts

Stochastic constraint Sub-solvers:
- **Temporal**
- Linear
- Finite domain
- Convex
- Global

Risk-aware Scheduling

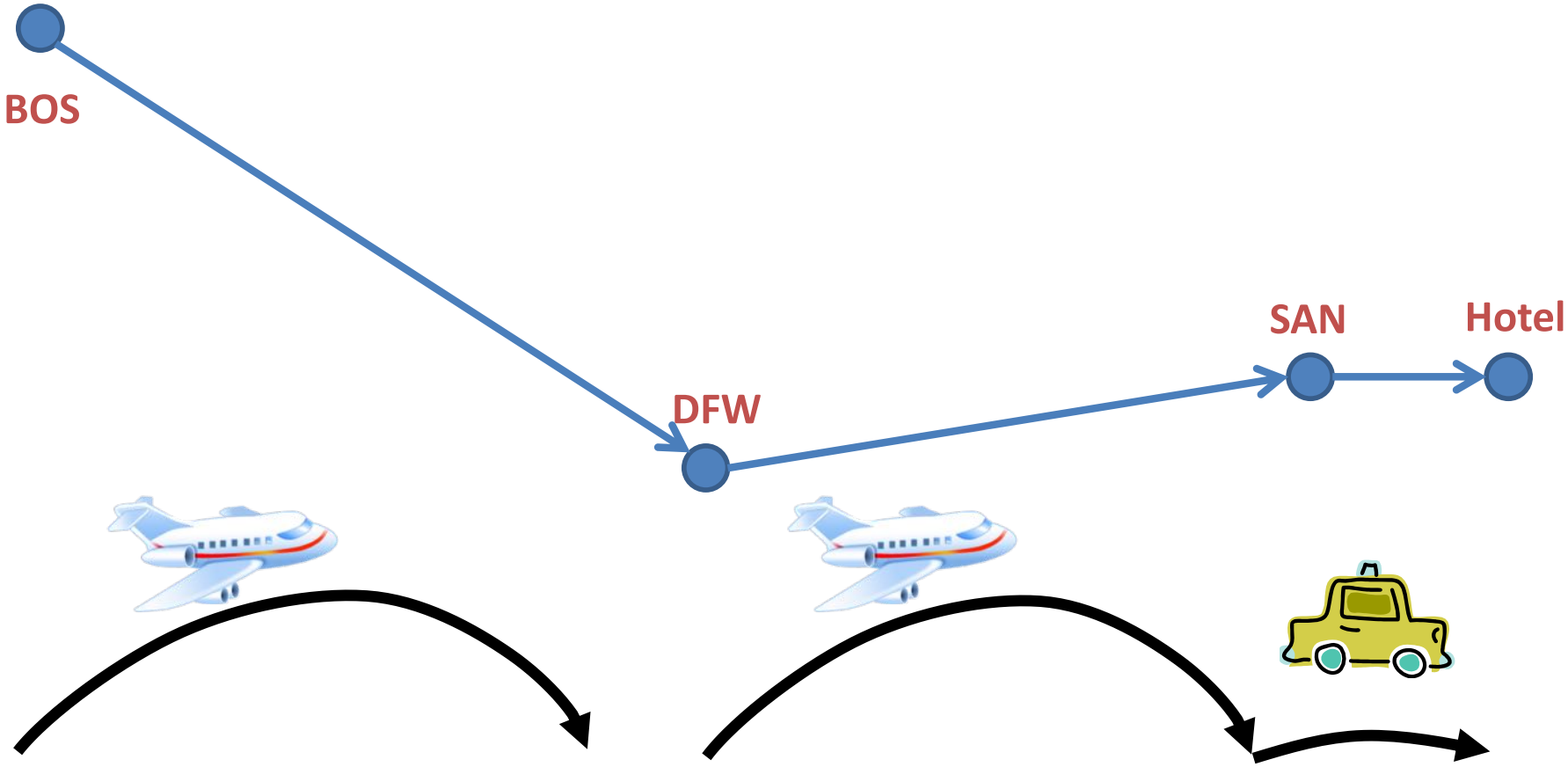
Scenario

Fast checking: [A Wang & Williams, AAAI 15]

Most preferred schedule: [Fang, Yu & Williams, AAAI 14]

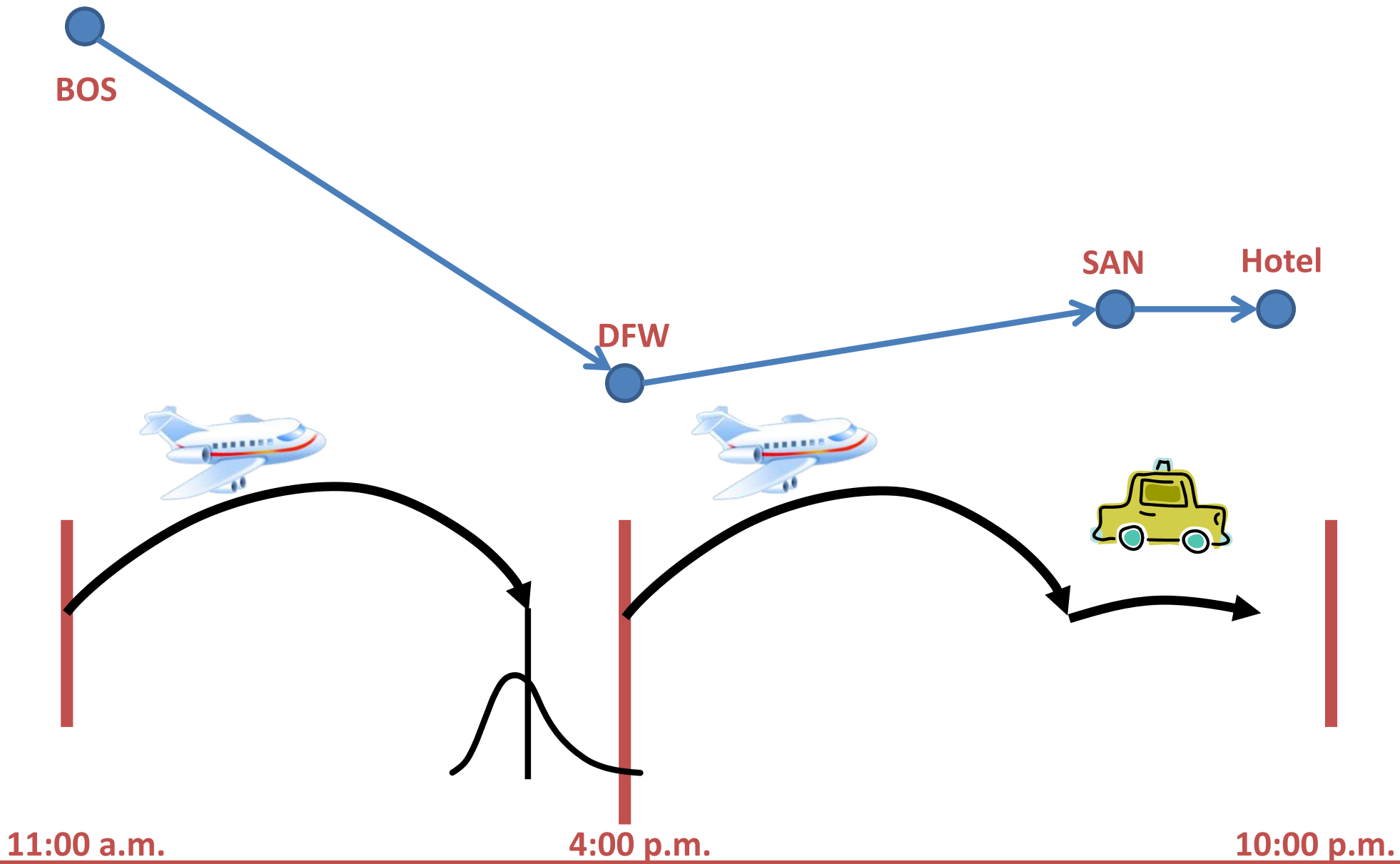
Example

Scenario



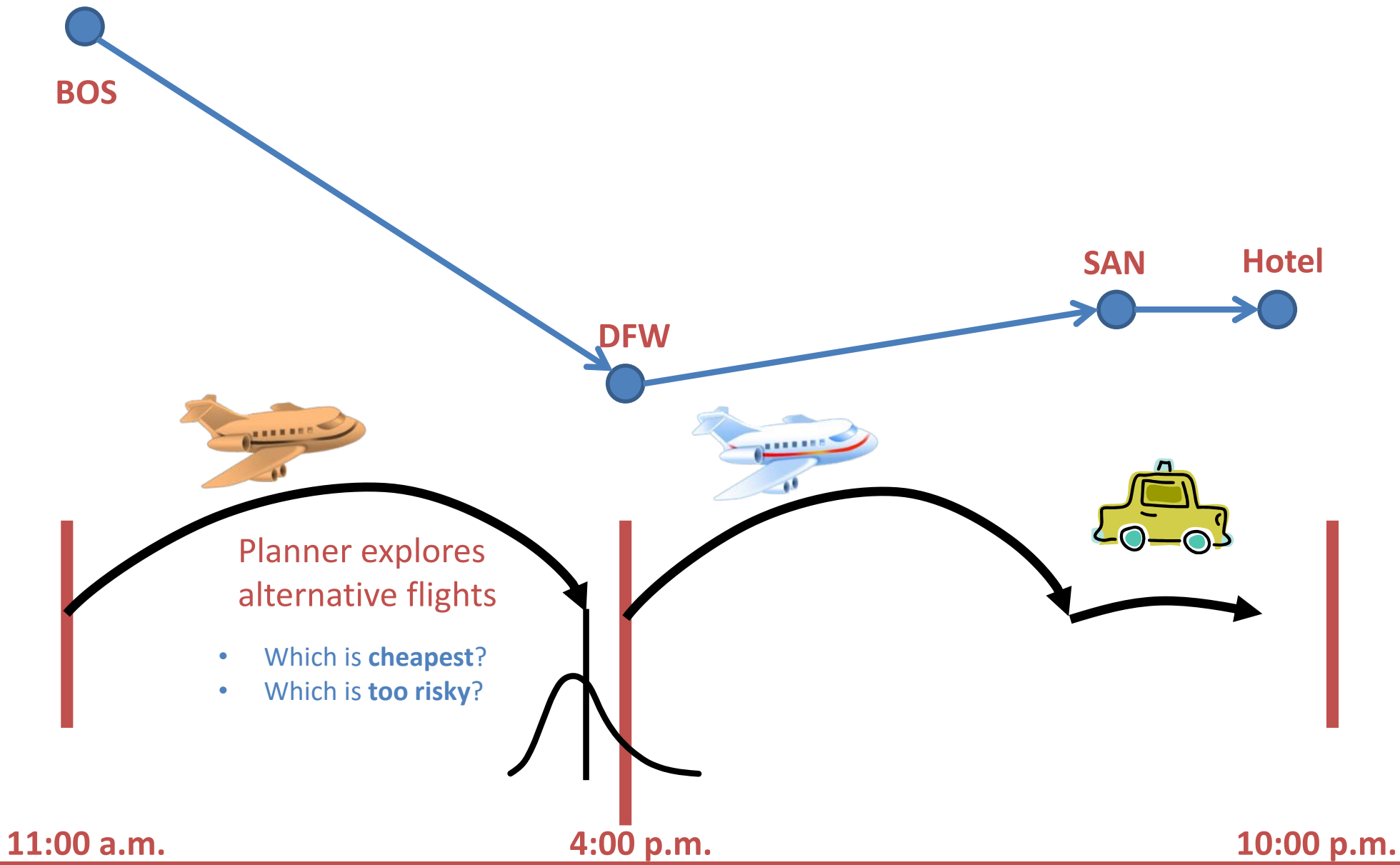
Example

Temporal risk



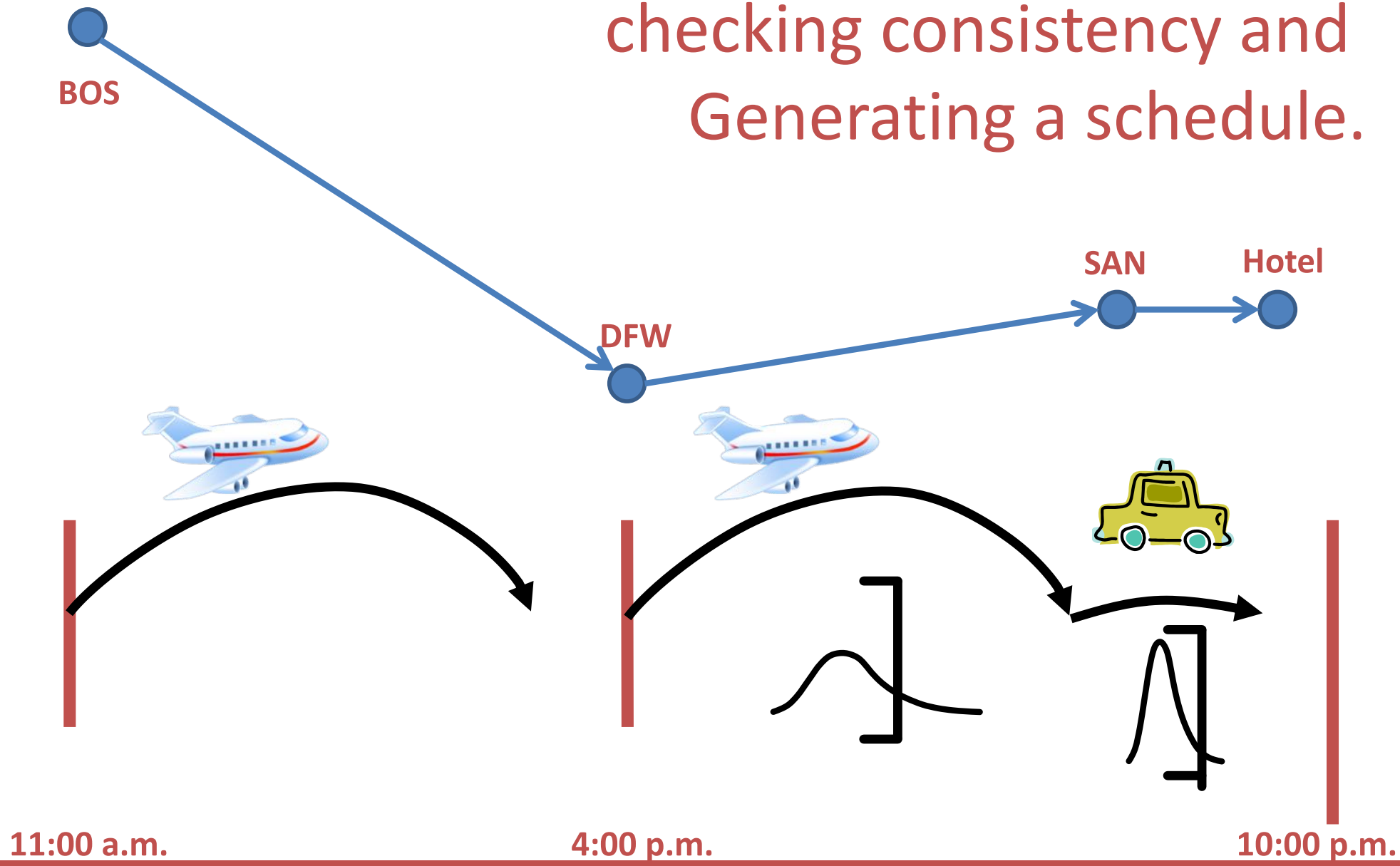
Example

Temporal risk



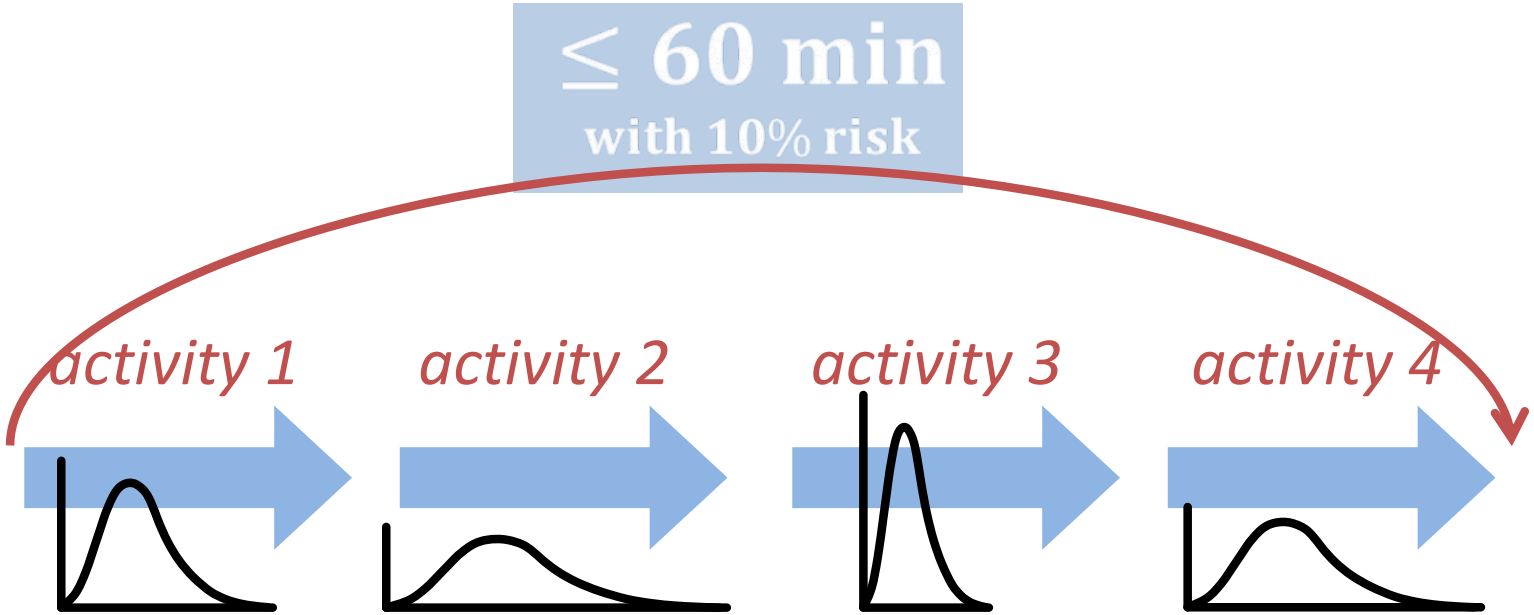
Example

Idea: Use **risk allocation** when checking consistency and Generating a schedule.



Problem

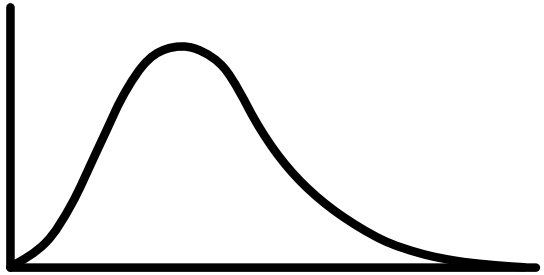
Input: pSTN + CC



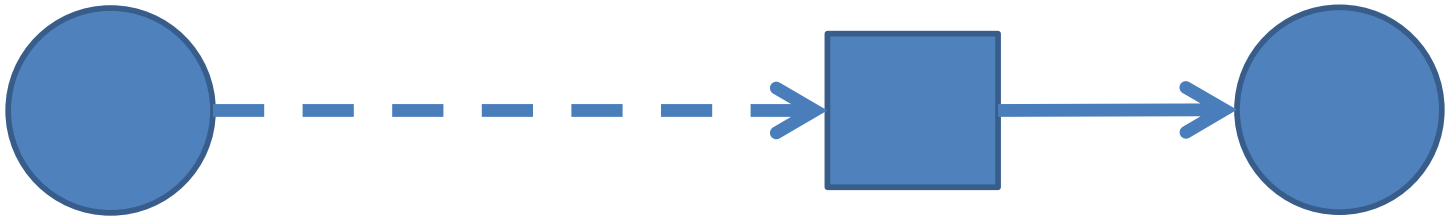
pSTN = probabilistic Simple Temporal Network
CC = risk bound ("chance constraint")

Problem

Input: activity model



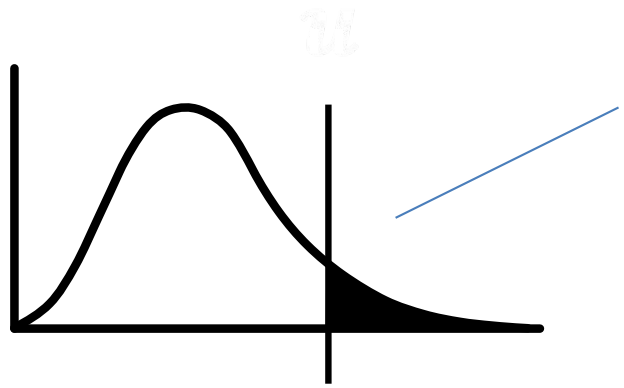
$[0, +\infty)$



Problem

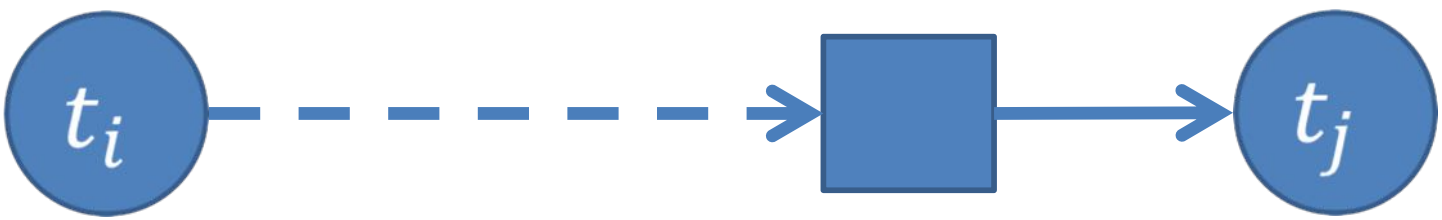
Output: scheduling strategy

Possible schedules + <u>risk allocation</u>



Risk allocation used to decide what to ignore.

$[0, +\infty)$

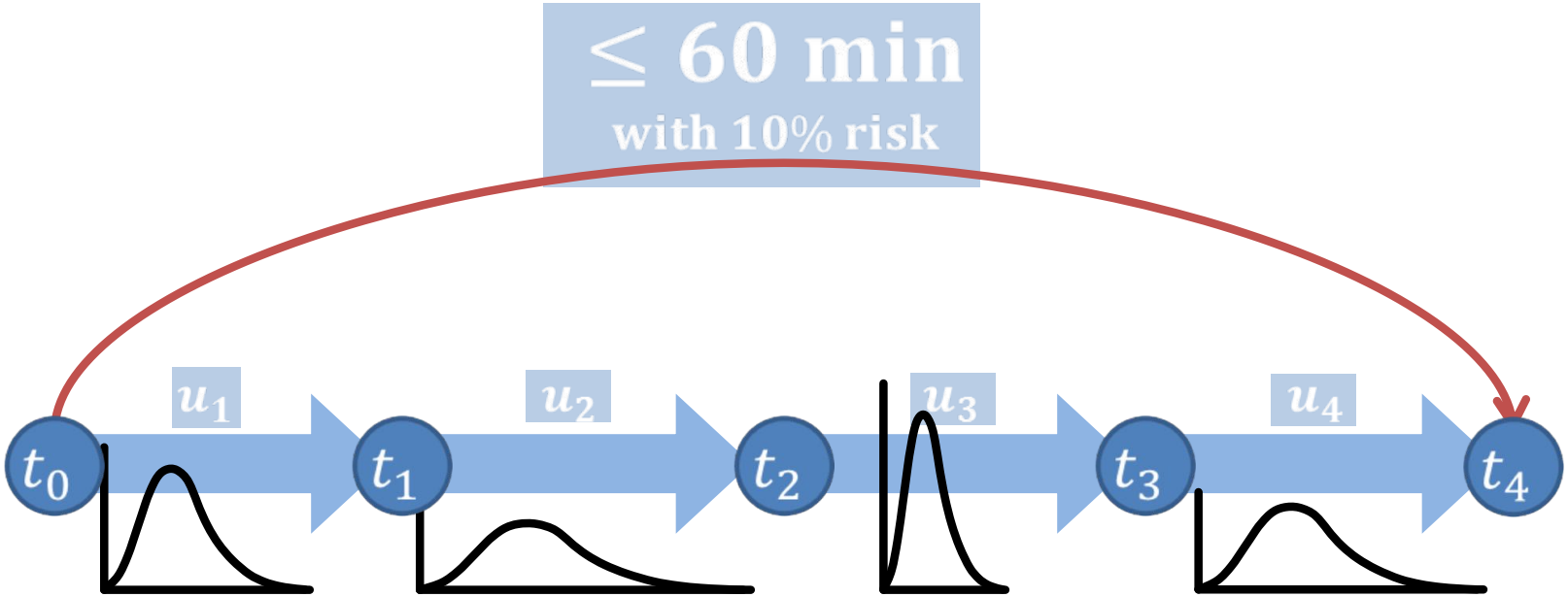


$$\text{risk} = \int_u^{+\infty} f(t)\,\mathrm{d}t$$

Problem

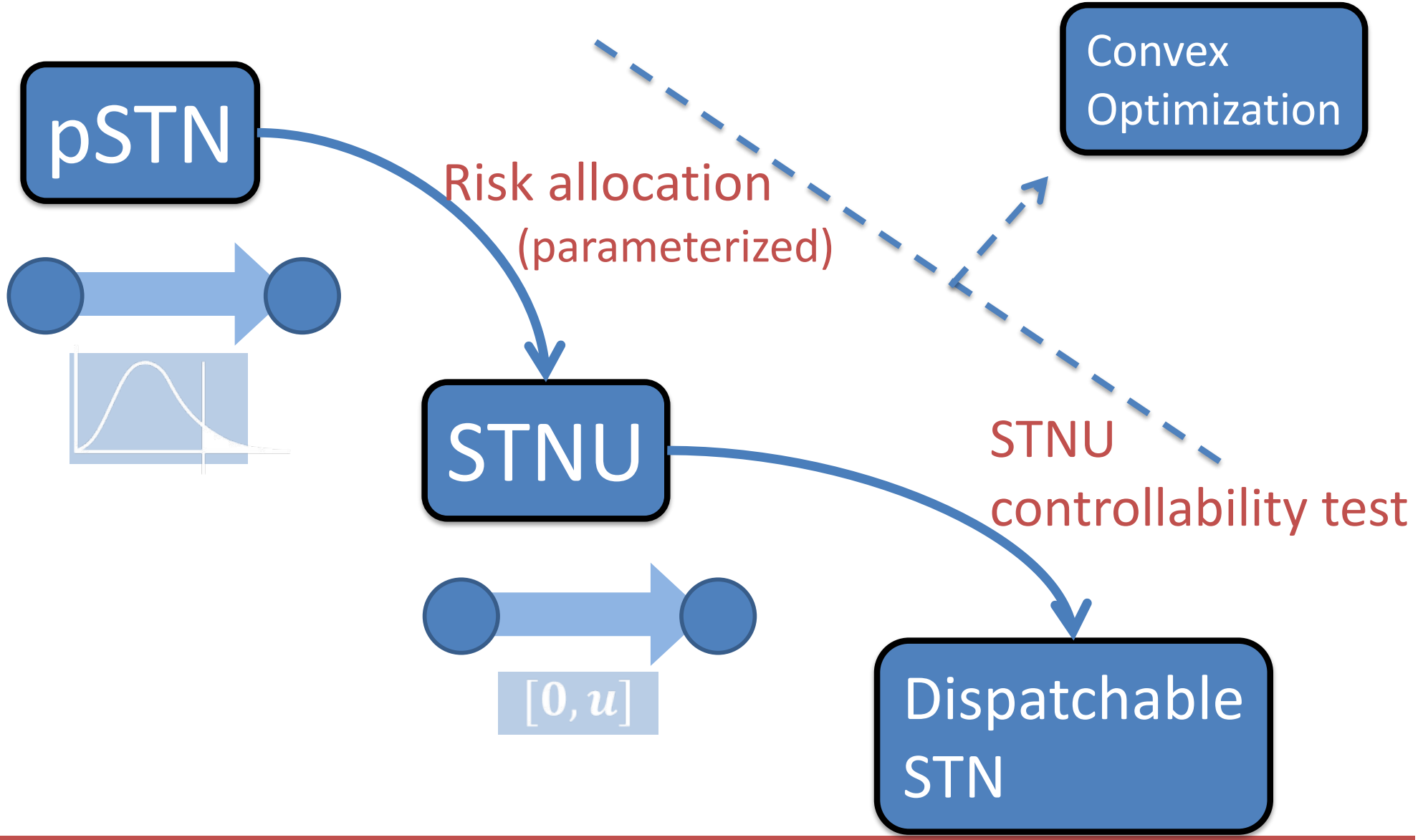
Output: scheduling strategy

Possible schedules + risk allocation



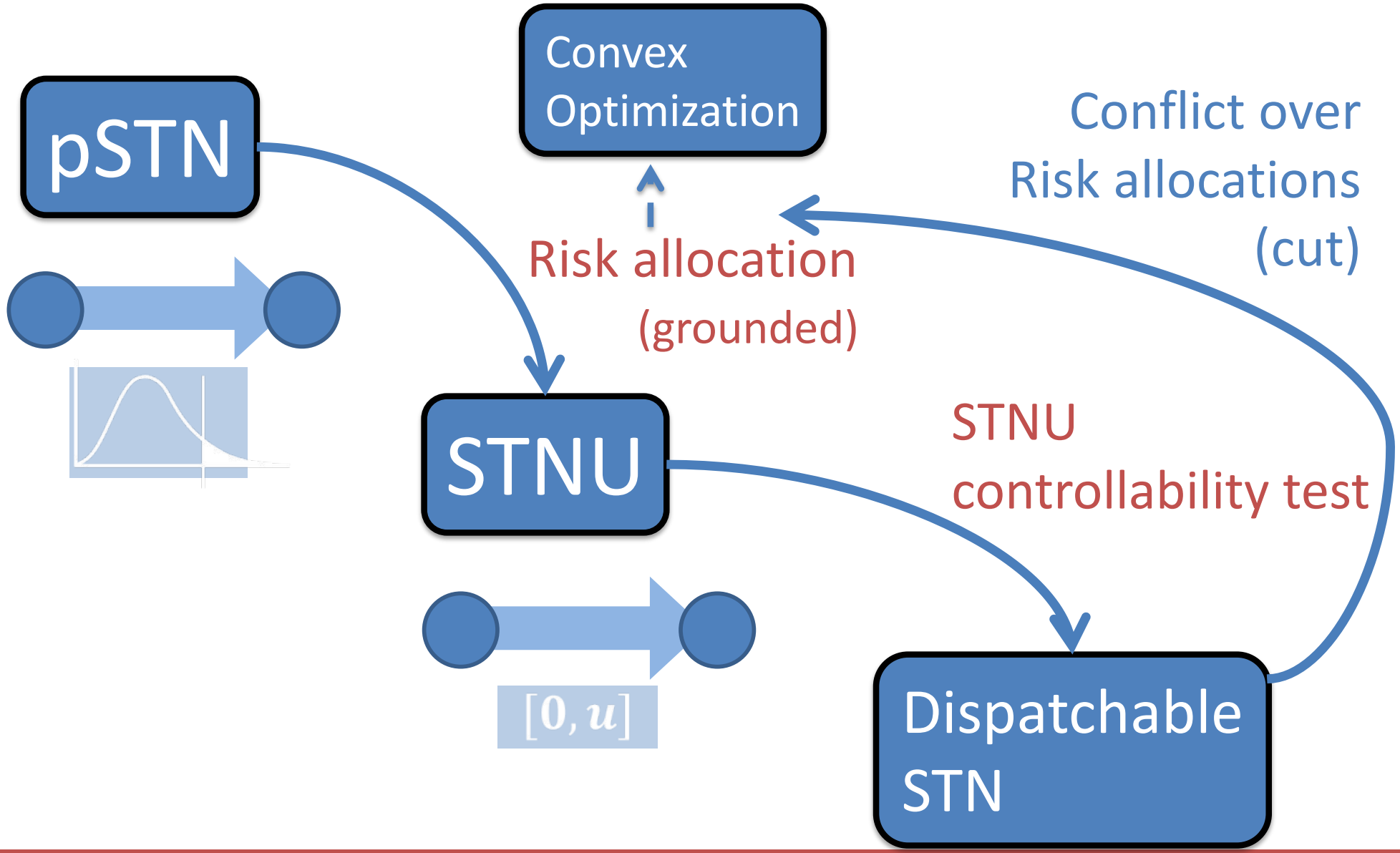
$$\sum \text{risk}_i \leq 10\%$$

Optimal Schedule Picard [Fang, Yu & Williams, AAAI14]



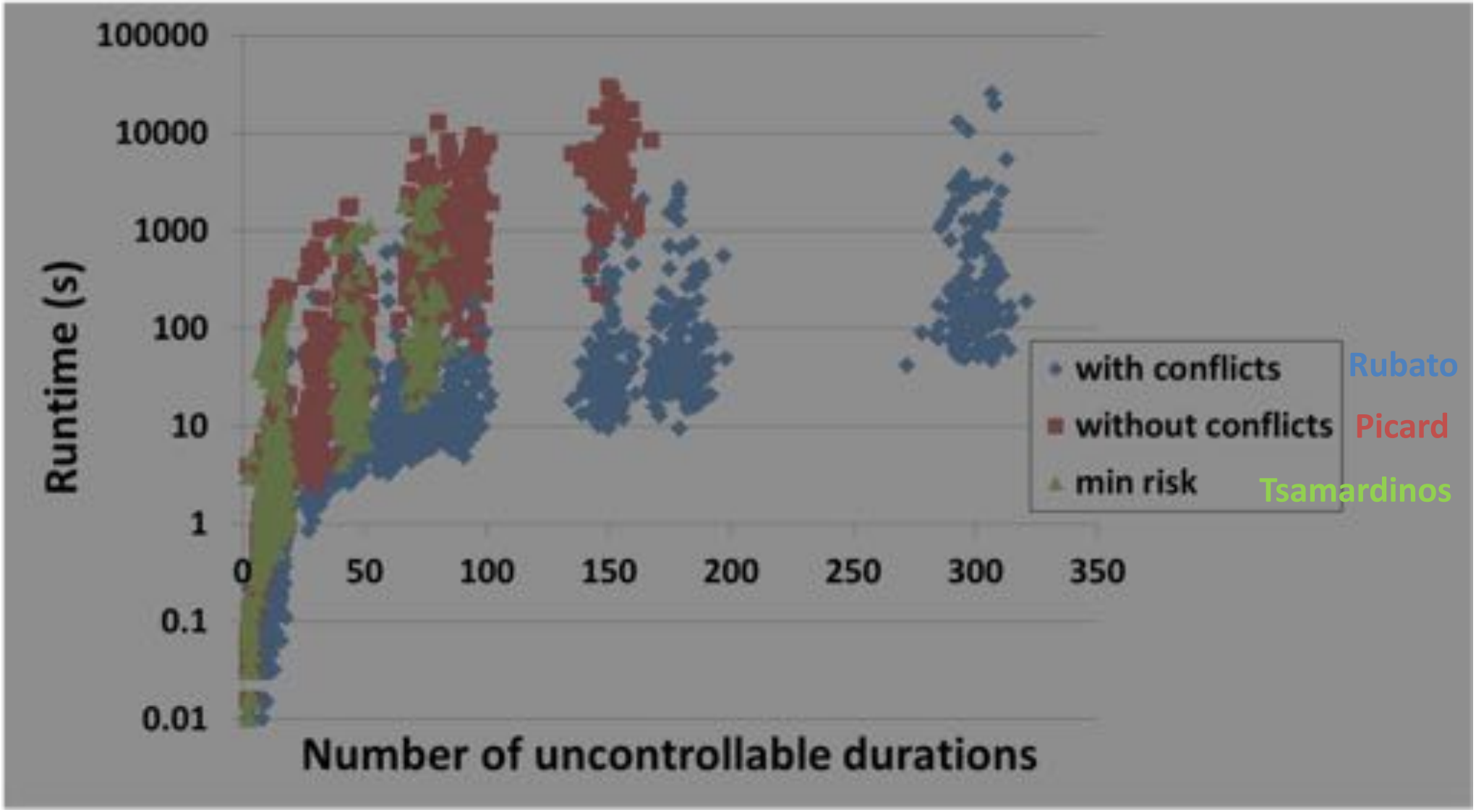
pSTN Feasibility

Rubato [A. Wang & Williams, AAAI15]



Algorithm

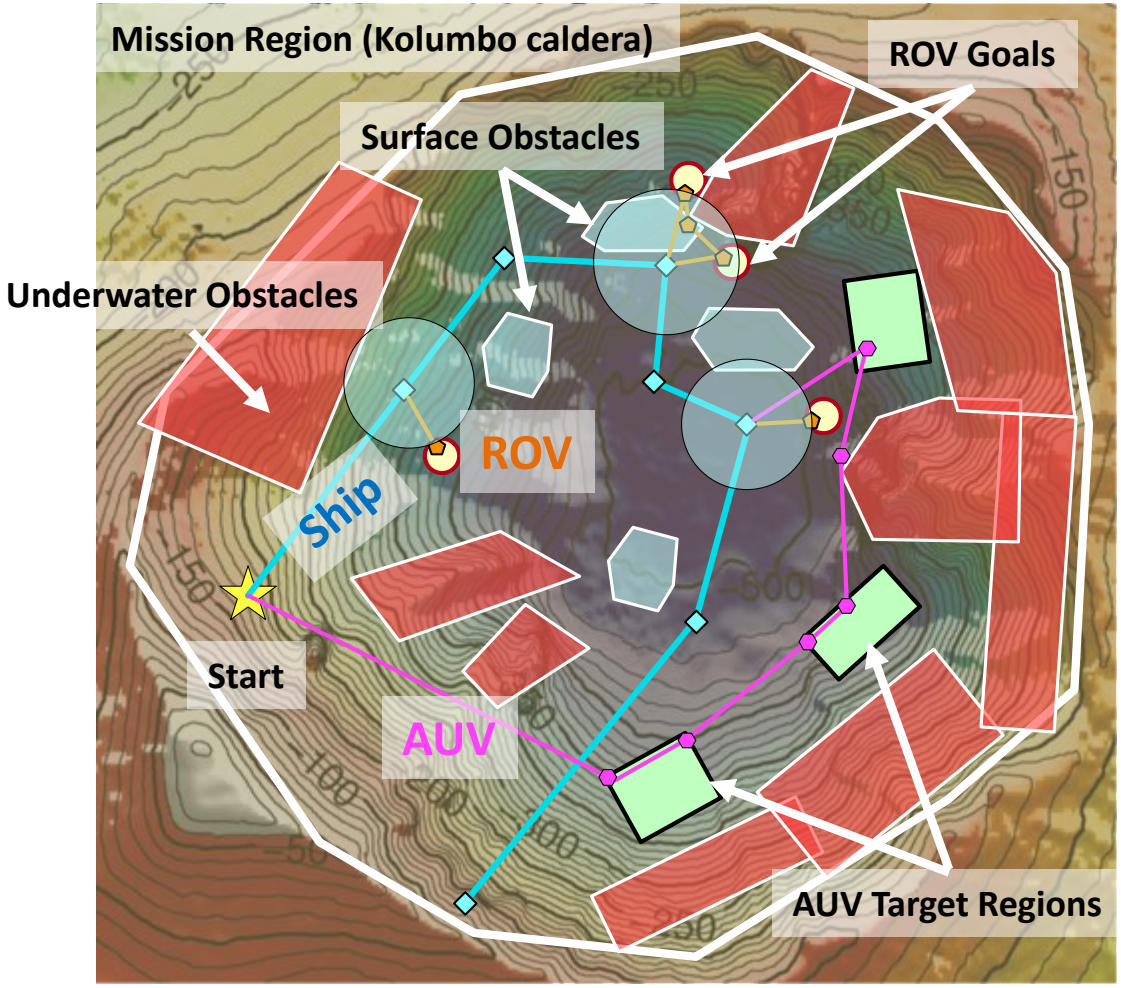
Performance



Outline

- Diving into the Real World
- Risk-bounded Planning as Optimization and Constraint Programming
- Generative Task and Motion Planning with Heuristic Forward Search
- Risk-bounded Planning by adding And / OR (AO*) Search

Hybrid Activity & Path Planning for the Columbo Volcano



Characteristics of the problem:

Coupling between activities and state constraints

ship-ROV tether range.
ship-AUV communication range.
AUV battery life.

Long horizons

missions span on order of hours or days.
long transit times.

Temporal constraints

maximum mission length.

Discrete actions and continuous paths

ship needs to pick up AUV to recharge.

Obstacles

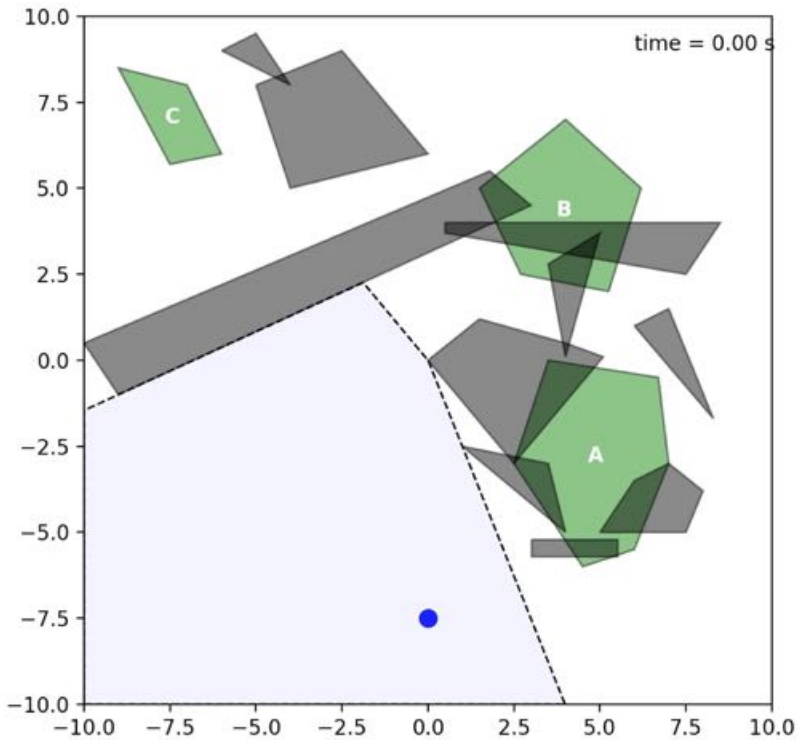
Planner needs to decide:

- **What activities** to perform.
- **When** to schedule them.
- The **vehicle paths.**

Scotty coordinates a tightly coupled robot fleet

Example scenario:

"**Ship** must drop off **ROV** to take samples at regions **A** and **B** and then deliver **ROV** to region **C** while minimizing **ship**'s travel distance. The **ROV** must stay within a certain distance of the **ship** at all times."



Scotty addresses these challenges:

1. Vehicles are coupled through space and time.
2. Reasons carefully about vehicle dynamics.
3. Plans over long horizons.

Hybrid Actions in Scotty PDDL

With a strategically selected dynamics model . . .

State Constraints:

$$\|\mathbf{x}_{ROV} - \mathbf{x}_{Ship}\|^2 \leq R^2_{tether}$$

$$\mathbf{x}_{AUV} \in \mathbf{R}_A$$

Scotty Hybrid Action Model

Control variables:

$$-2 \leq v_x \leq 2 \qquad \mathbf{v} = (v_x, v_y)$$

$$-2 \leq v_y \leq 2 \qquad \|\mathbf{v}\| \leq v_{max}$$

Continuous Effects:

- Linear combinations of control variables (e.g. vehicle motions)

$$x(t) = x_0 + v_x \cdot (t - t_0)$$

$$y(t) = y_0 + v_y \cdot (t - t_0)$$

- Norms or squared norms of control variable vectors (e.g battery consumption)

$$\dot{b}(t) = -k\|\mathbf{v}(t)\|^2$$

Hybrid Actions in Scotty PDDL

we can use convex optimization

State Constraints:

$$\|\mathbf{x}_{ROV} - \mathbf{x}_{Ship}\|^2 \leq R_{tether}^2$$

$$\mathbf{x}_{AUV} \in \mathbf{R}_A$$

Scotty Hybrid Action Model

Control variables:

$$-2 \leq v_x \leq 2$$

$$-2 \leq v_y \leq 2$$

$$\mathbf{v} = (v_x, v_y)$$

$$\|\mathbf{v}\| \leq v_{max}$$

Continuous Effects:

- Linear combinations of control variables (e.g. vehicle motions)

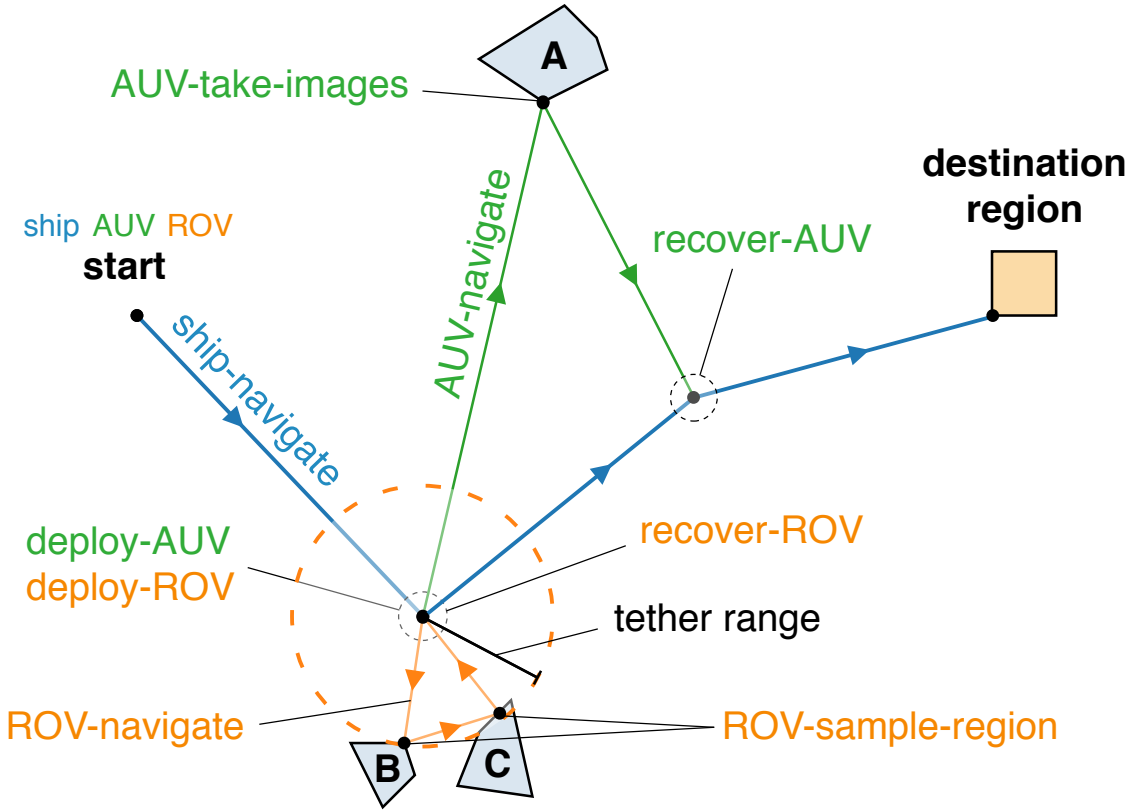
$$x(t) = x_0 + v_x \cdot (t - t_0)$$

$$y(t) = y_0 + v_y \cdot (t - t_0)$$

- Norms or squared norms of control variable vectors (e.g battery consumption)

$$\dot{b}(t) = -k\|\mathbf{v}(t)\|^2$$

Scotty Output: Hybrid Task and Motion Plan

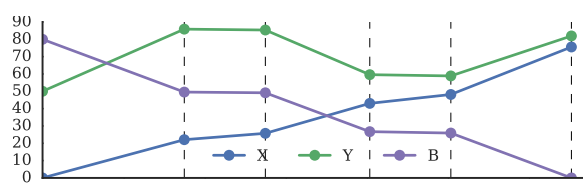


Solution Plan

Schedule	Sequence of activities
t=0	start-ship-navigate
t=15.0	start-deploy-AUV
t=17.0	end-deploy-AUV
t=17.5	start-deploy-ROV
t=20.5	end-deploy-ROV
t=22	start-AUV-navigate
…	start-ROV-navigate
	end-ROV-navigate
	start-ROV-sample-region
	end-ROV-sample-region
	start-AUV-take-images
	end-AUV-take-images
	start-ROV-navigate
	start-recover-AUV
	…

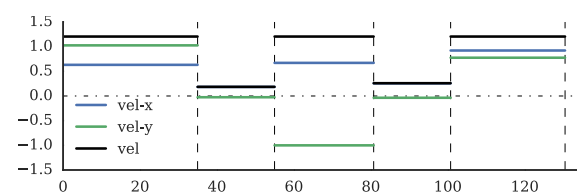
State trajectory

Piecewise linear robot positions



Control trajectory

Piecewise constant robot velocities

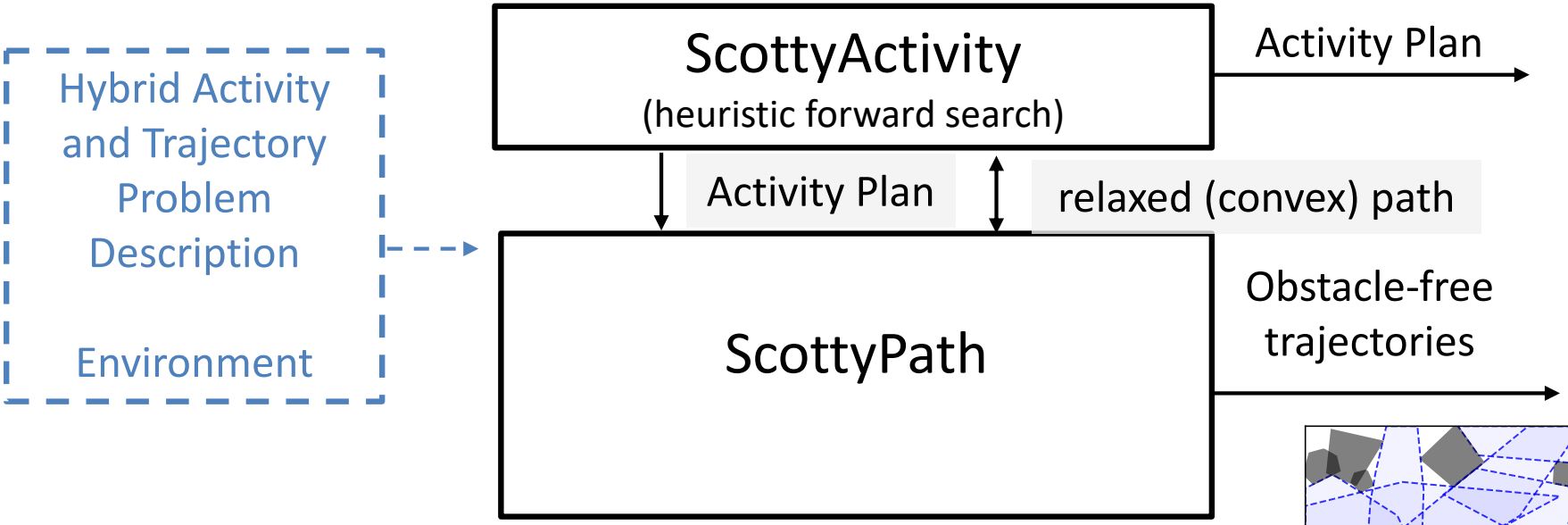


The Scotty Planner

Greedy Heuristic Forward Search Activity Planner

+

Heuristic Based on Relaxed Convex Scotty Path



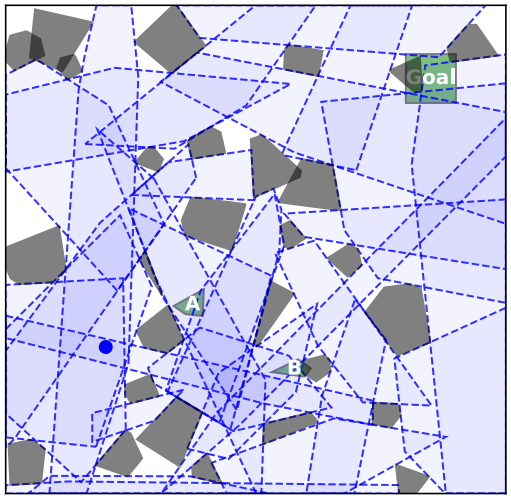
"Convexify" Obstacles:

- **Construct region graph of overlapping convex regions.**
- **Perform Heuristic Forward Search over region graph.**

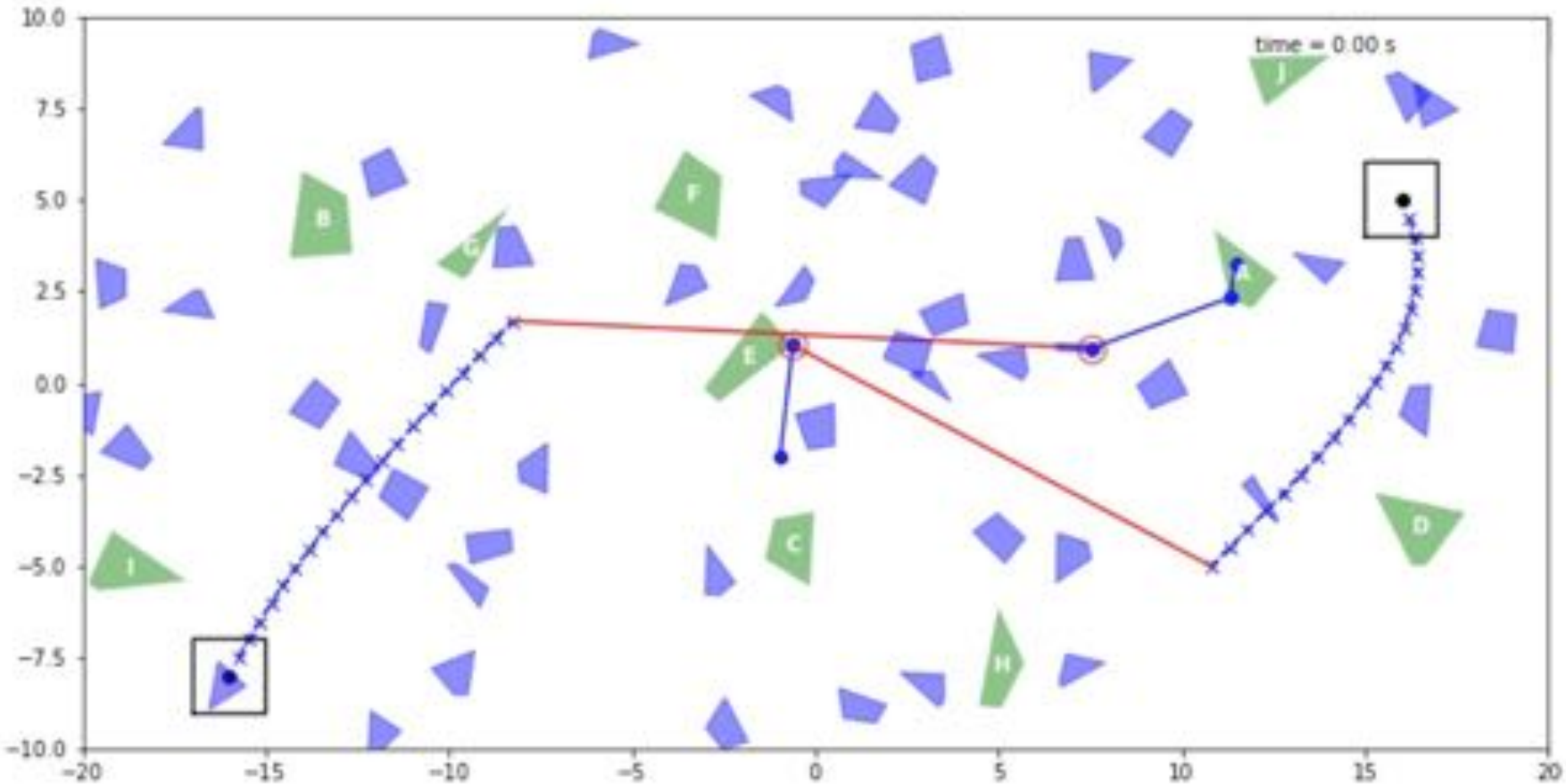
+

Long horizon, convex multi-vehicle path planning over time (QSP):

- **Use a continuous state and time formulation.**
- **Encode as a Second Order Cone Program.**

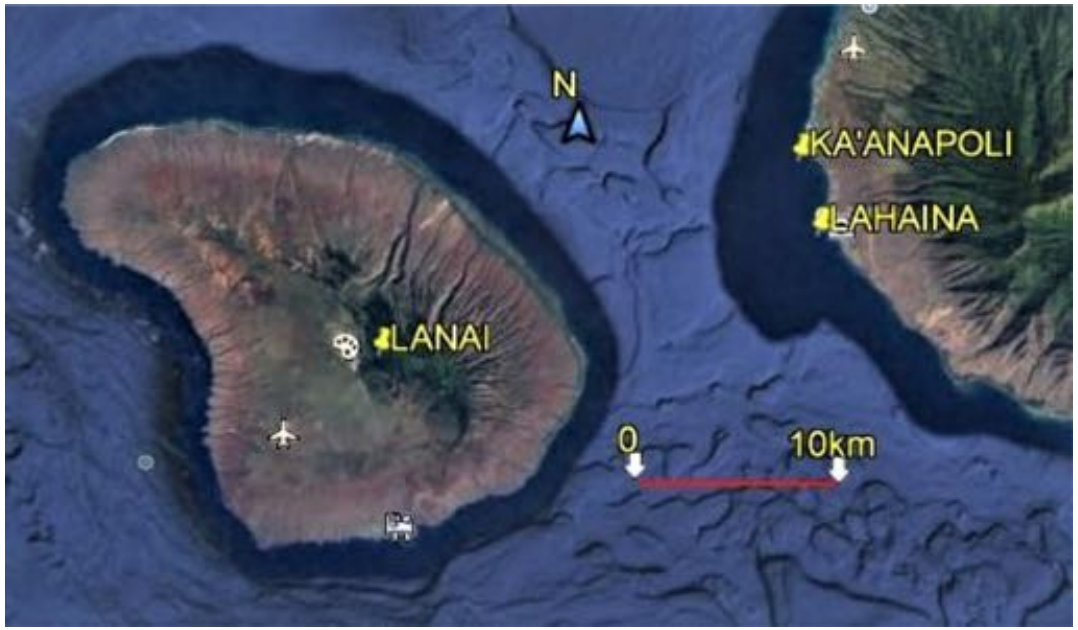


Online Scotty continuously refines paths while executing, using detailed dynamics

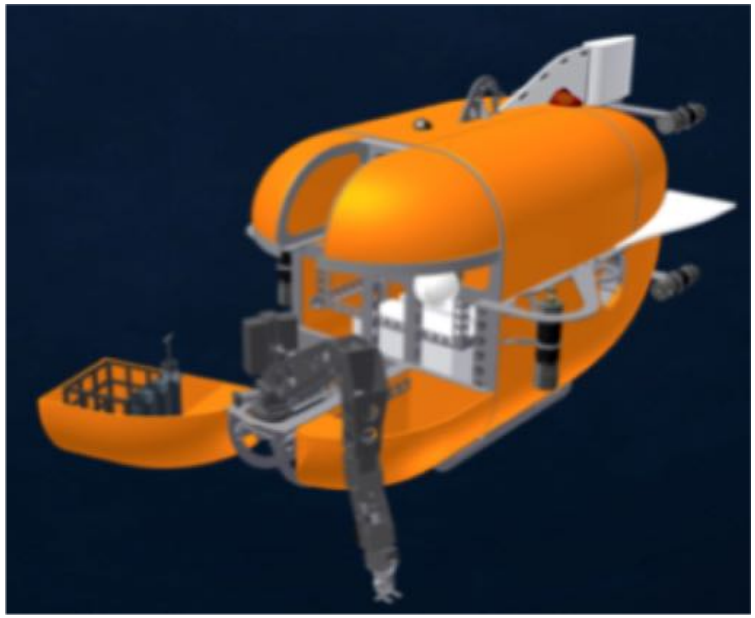


[Reeves, Fernandez, Williams in progress]

January, 2018: Adaptive Science by Coordinating Human and Robot Teams along Au Au Channel, Hawaii



December, 2018: Safe Manipulation and Sampling off Costa Rica



Outline

- Diving into the Real World
- Risk-bounded Planning as Optimization and Constraint Programming
- Generative Task and Motion Planning with Heuristic Forward Search
- Risk-bounded Planning by adding And / OR (AO*) Search

Risky Business

Navigating the Oceans
of Boston Traffic

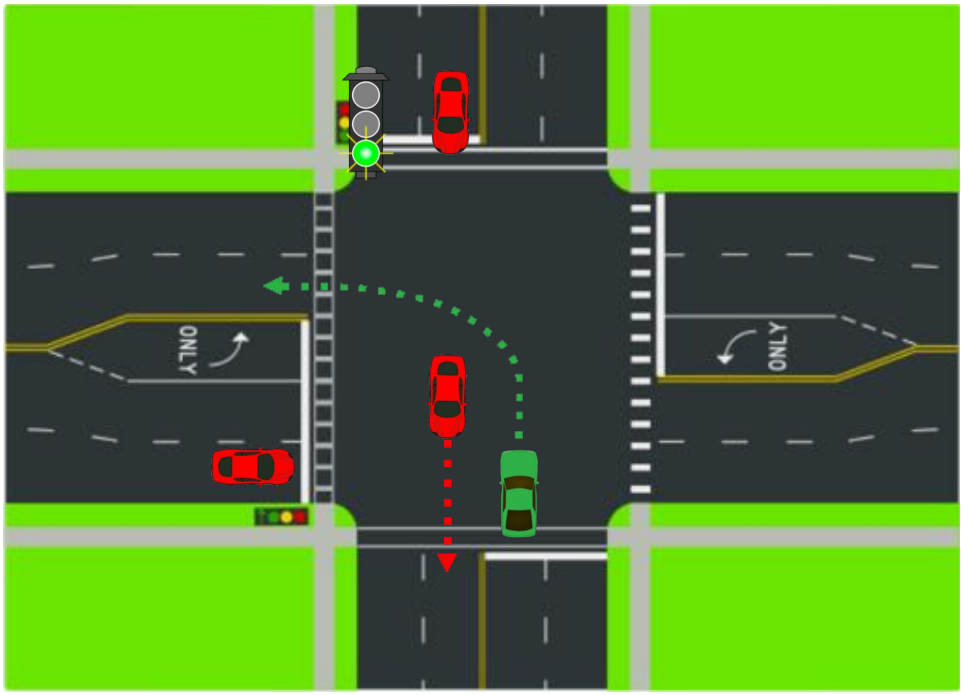
Problem:
Driving is stressful and dangerous

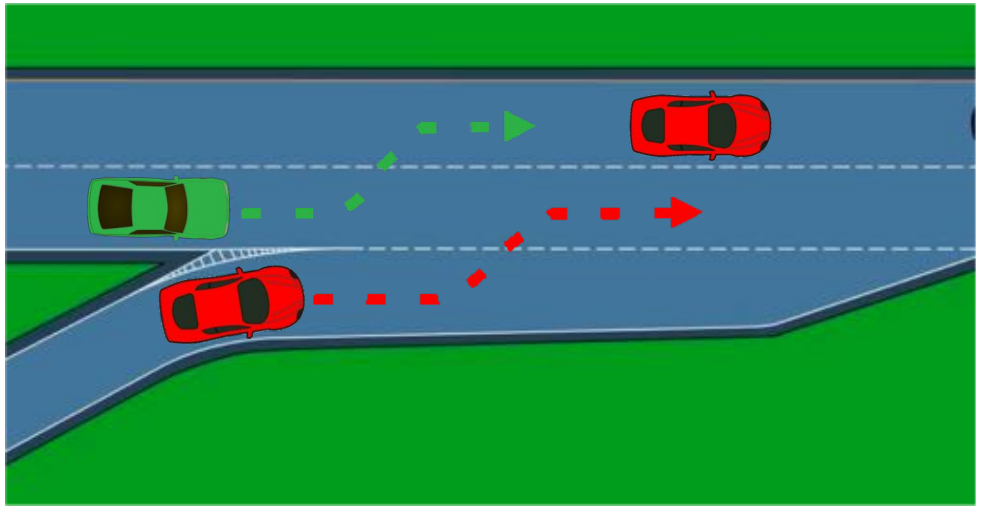
- Safety needs to be improved.
- Driving needs to be more relaxing.

- Cognitive overload needs to be reduced.
- Transfer of control between driver and vehicle must be smooth.

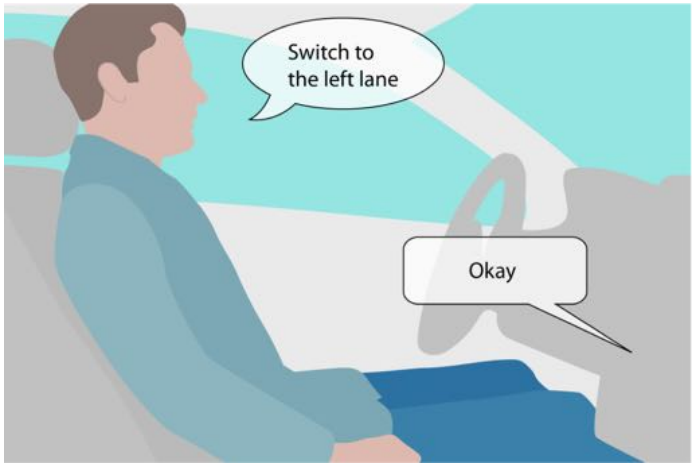
vehicle.

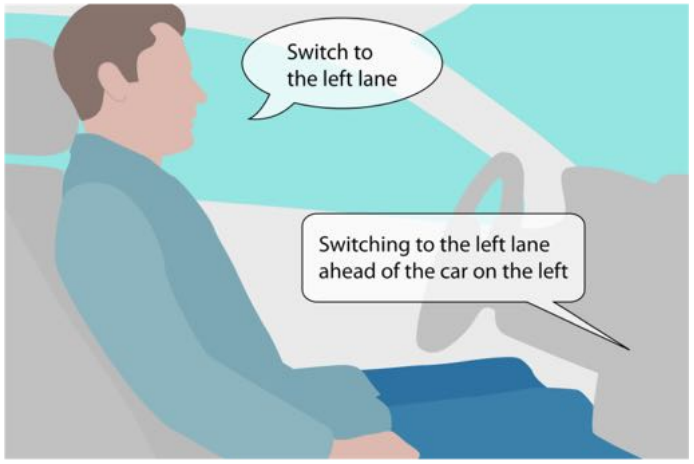
Need Safely Navigate these Dangerous Scenarios

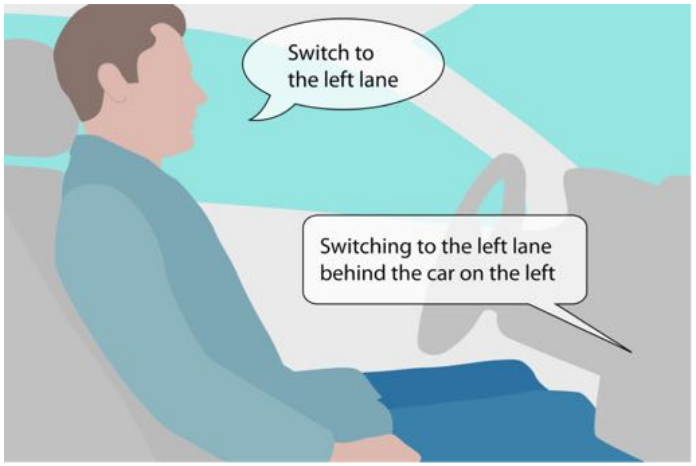


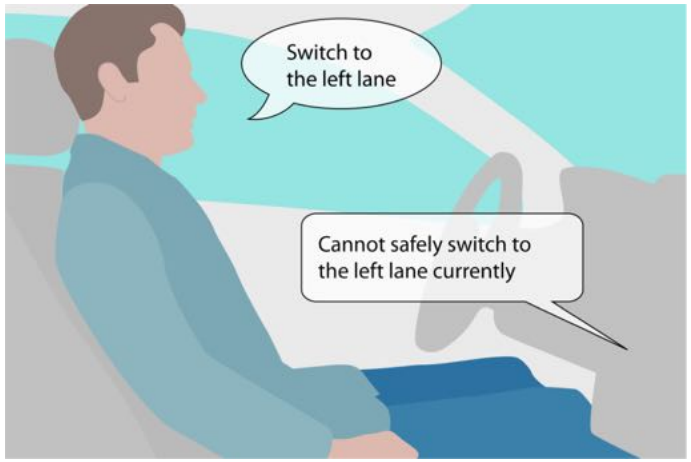


Approach: Drive using **Qualitative Maneuvers**









- Human "still drives."
- More latitude.
- Less cognitive load.
- Simpler transition.

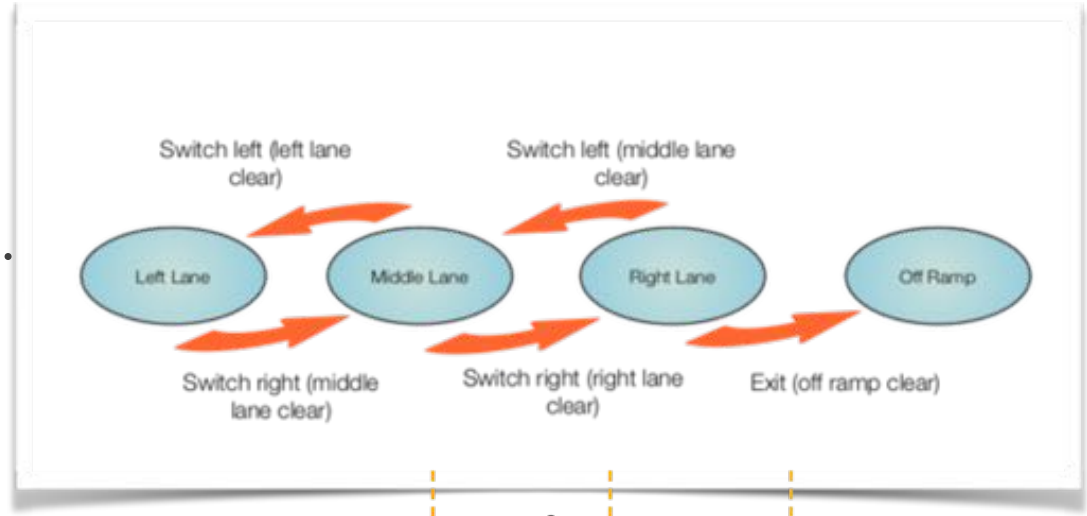
Geordi

- Vehicle performs requested maneuver safely, in context of its environment.

Maneuver models for vehicle control

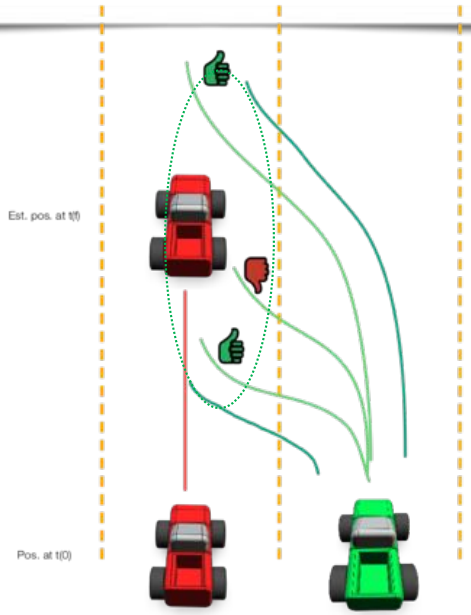
Qualitative:

- Maneuvers are "actions" specifying poses over time.
  - Switch left,
    switch right,
    exit



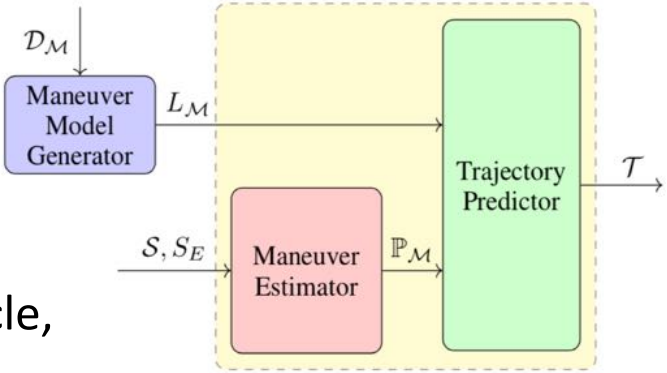
Quantitative:

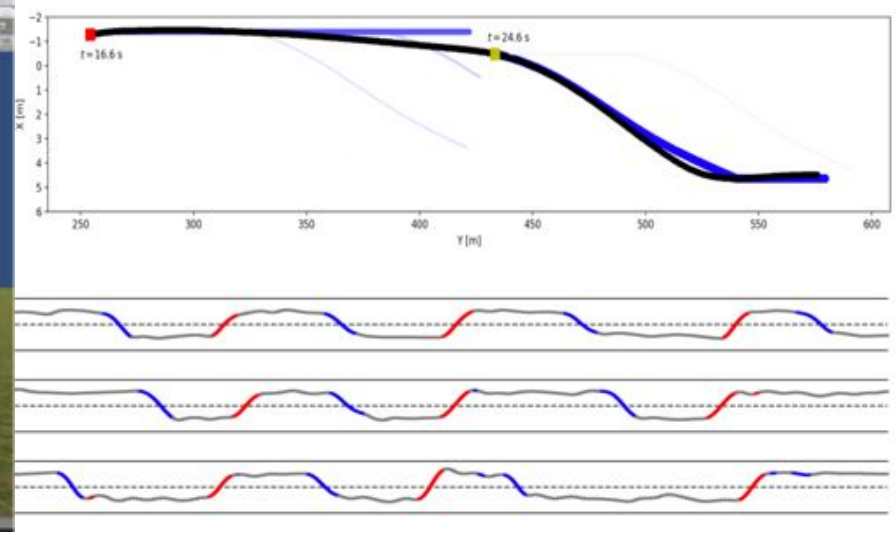
- A maneuver is a bundle of trajectories, called a flow tube.
- A maneuver is possible if at least one trajectory in its flow tube is unblocked.



1. **Geordi** predicts the intent and style of other drivers.

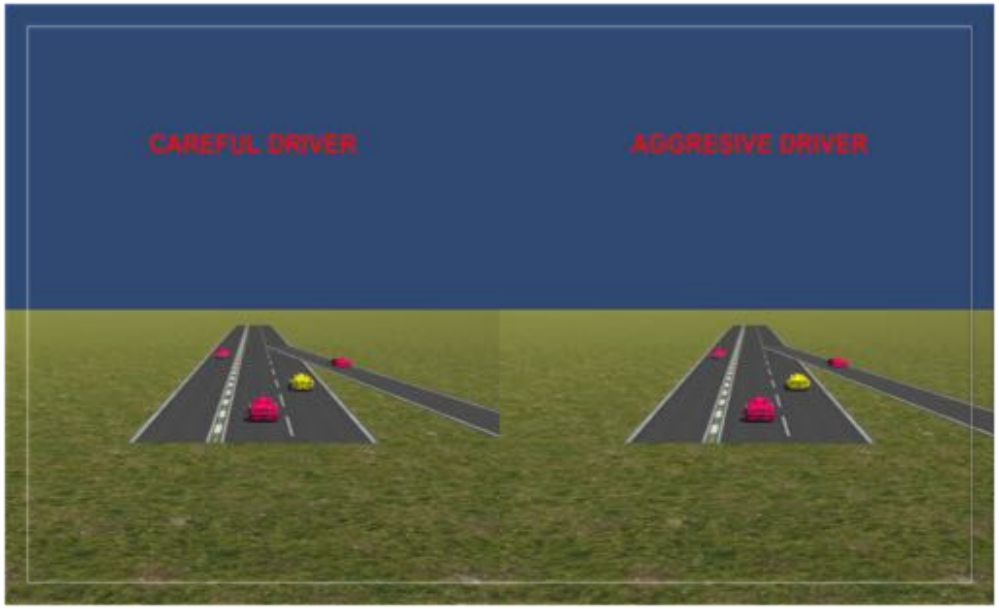
- Geordi predicts probabiliy of driver type, for each surrounding vehicle.
- Geordi predicts probabilities of possible maneuvers, for each surrounding vehicle, given driver type.
- Geordi predicts possible trajectories for each surrounding vehicle, given its maneuver sequence and driver type.

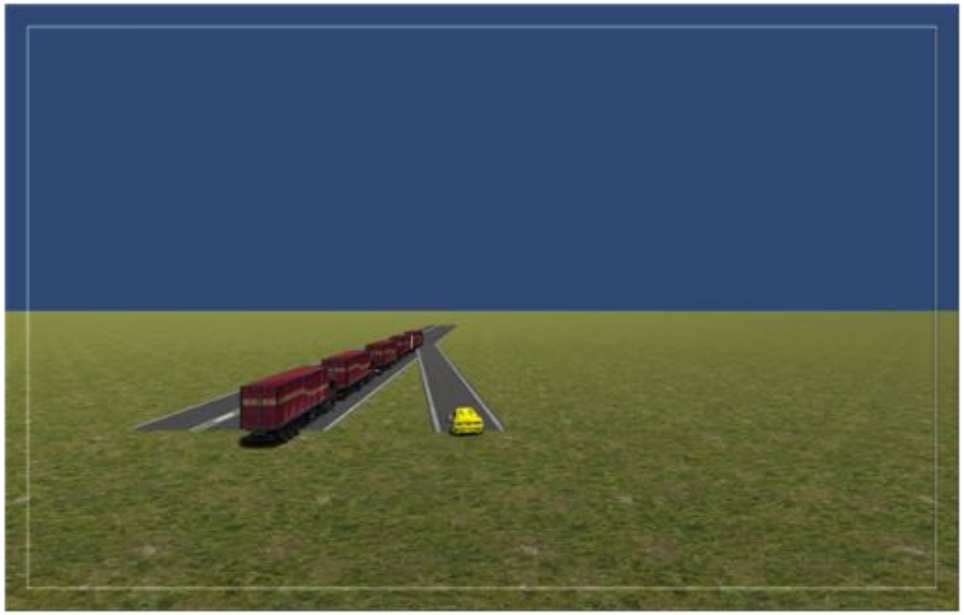




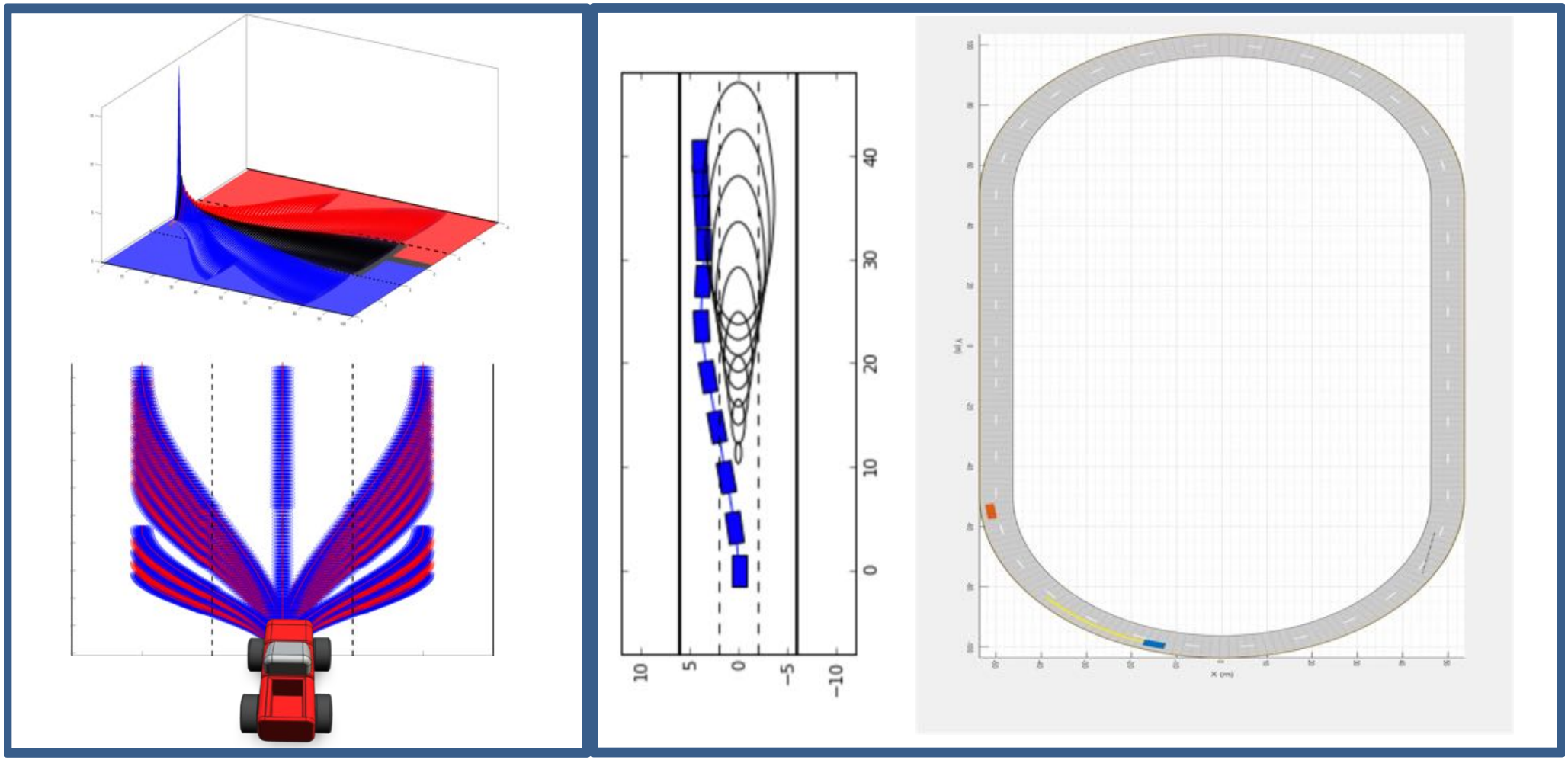
2. Geordi *"proactively"* plans maneuvers in crowded situations, while constantly assessing risks.

- **Geordi** generates risk-bounded *"high level"* maneuver sequences, which humans can understand and trust.
- **Geordi** generates "backup" sequences across contingencies, to produce safe proactive driving behaviors.





3. Geordi plans safe, risk-aware vehicle trajectories.



How about Probabilistic Planning of Maneuver Sequences?

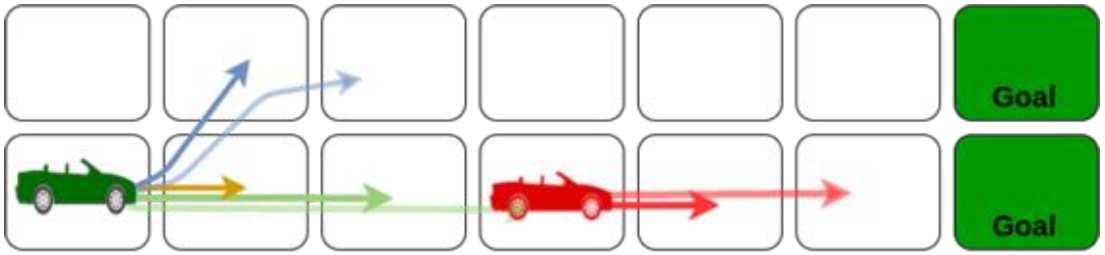
"Always have a backup"

Approach: Pre-compute conditional plan for all possible outcomes, using probabilistic model of traffic.

AO* = A* + And/Or Graphs + Bellman Backup ➡ Conditional Plan

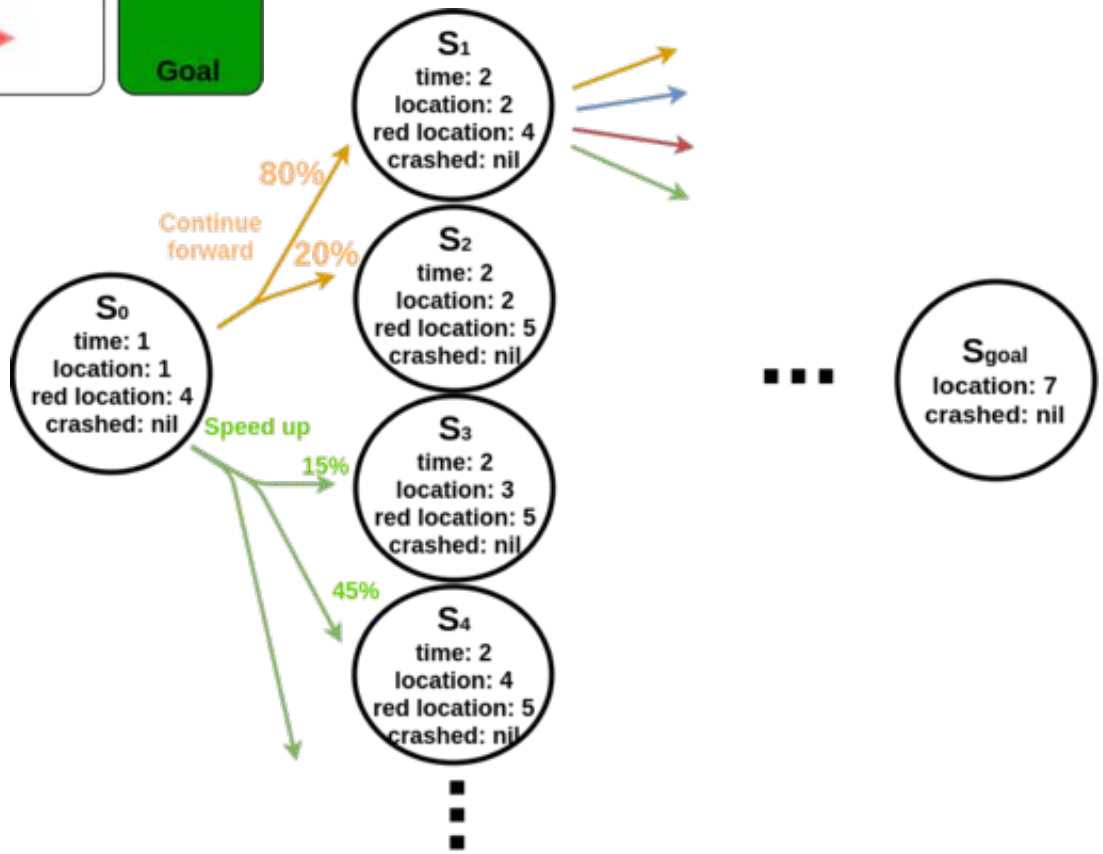
Monte Carlo Tree Search = Sampling + UCB + . . .

How about Probabilistic Planning of Maneuver Sequences?



Feature:

- Fast response online.

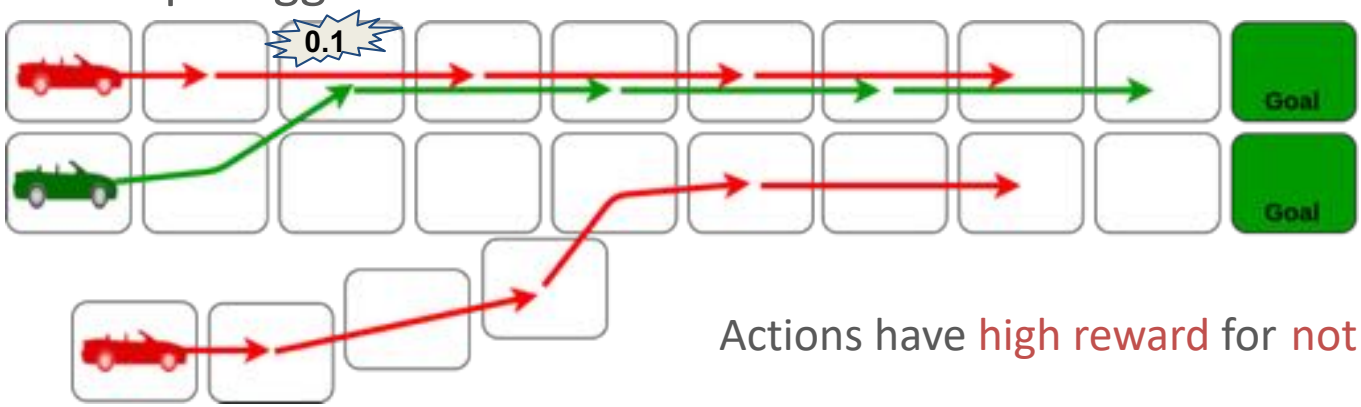


Conditional Planning? - High-Risk Policies

Problem 1: Maneuver sequence may take **arbitrary high risk**

for arbitrary high reward.

Example aggressive maneuver



Actions have high reward for not slowing down

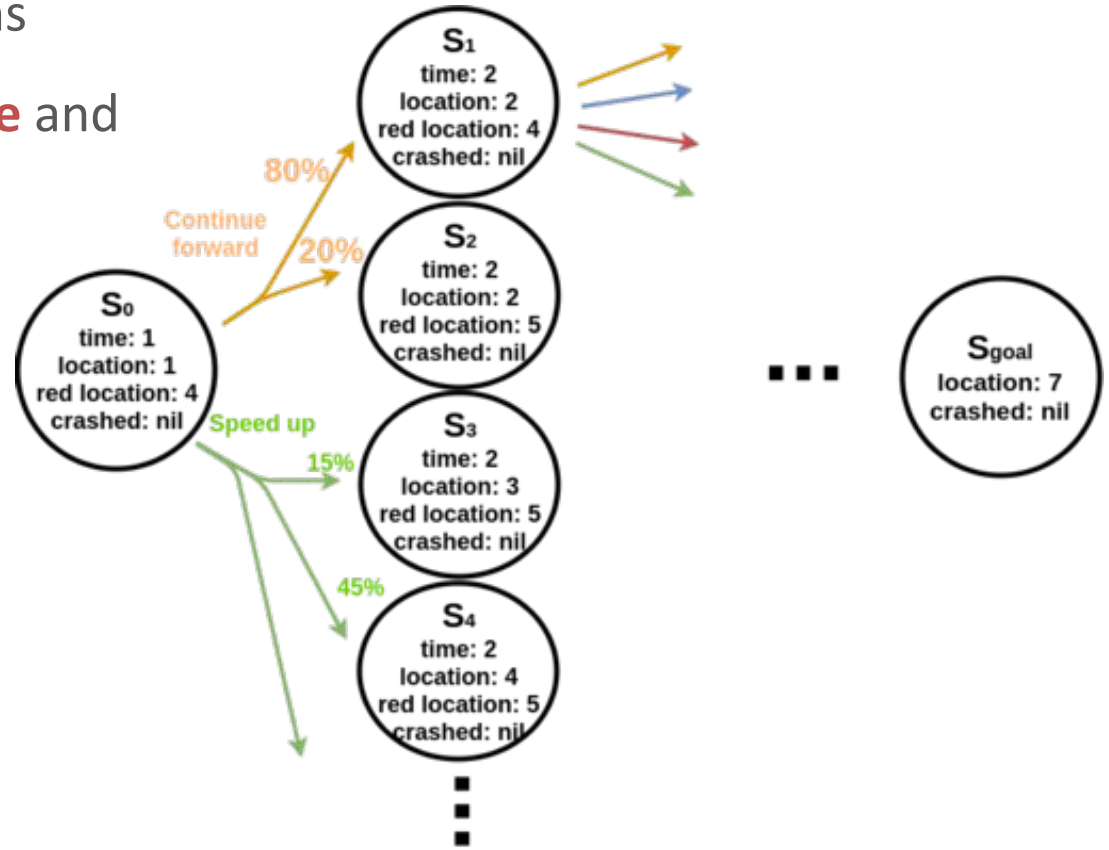
but also a 10% chance of near collision

Conditional Planning? - Search Space Size

Problem 2:

- AO* search over stochastic actions explores a **very large search space** and generates **very large policies**.
- MCT reduces space, but **doesn't bound risk**.

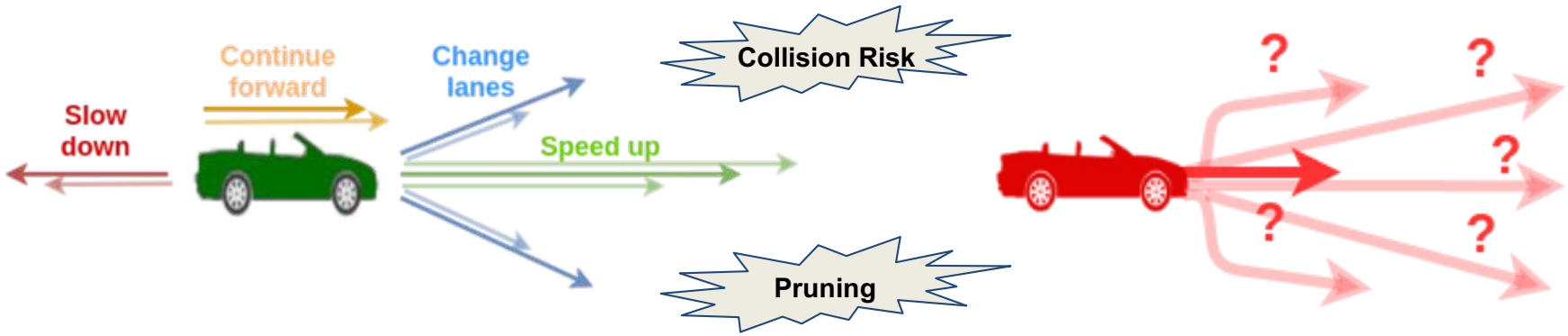
Example search space



Risk-bounded Conditional Planning

RAO* = AO* + Execution Risk Estimation + Risk Cutoffs

- **Prunes actions** from search space that are **too risky**.
- Greedily plans for most likely outcomes first, and
- **Terminates** planning as soon as risk bound is **guaranteed**.



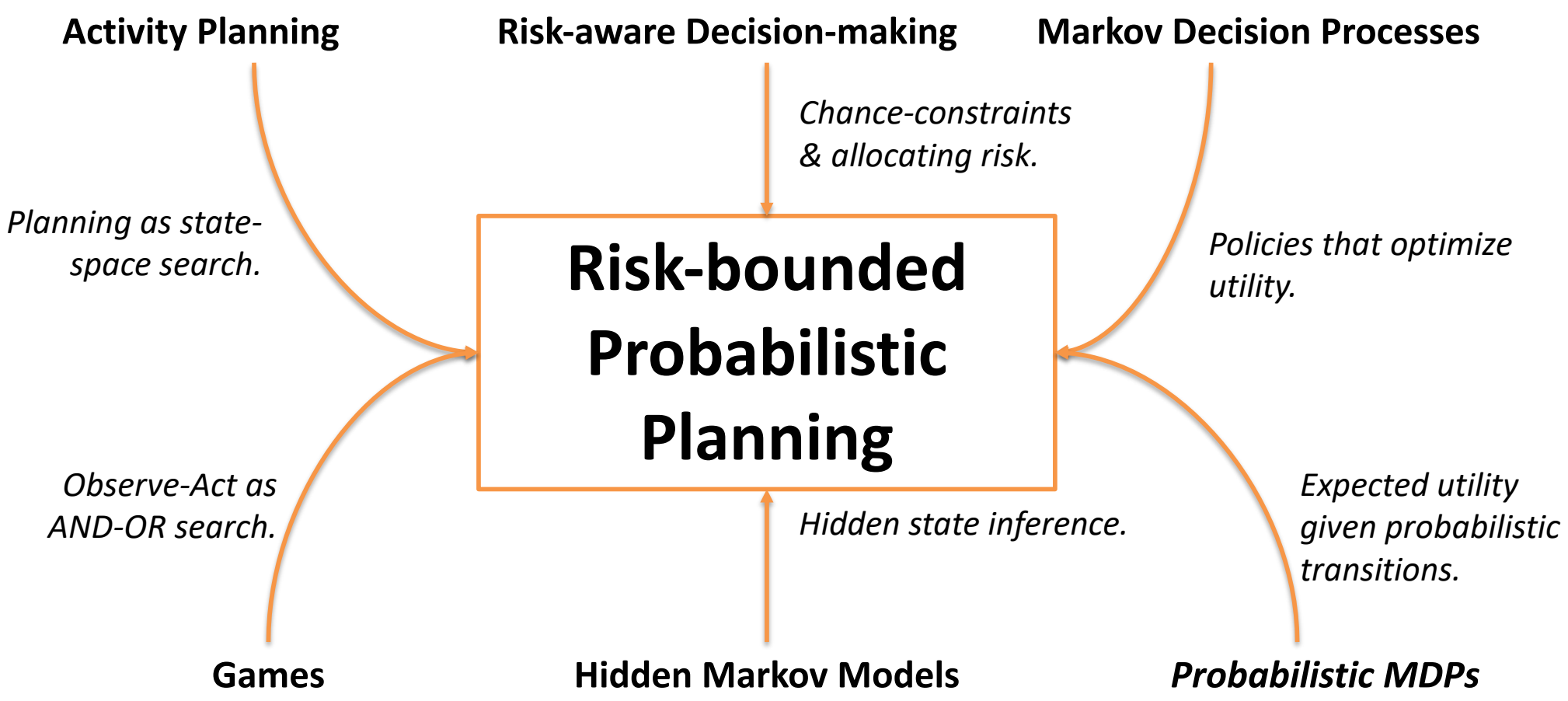
Online, Risk-bounded Conditional Planning

Problem 3: What if outcomes that were ignored, start **looking likely**?

"Always have a backup to the backup"

Approach:

- Update **risk-estimates** for the conditional plan **on-line**, based on belief state up-dates.
- **Expand conditional plan** for most likely uncovered outcomes, when conditional plan no-longer satisfies risk-bounds.


Activity Planning
Risk-aware Decision-making
Markov Decision Processes
Chance-constraints & allocating risk.
Planning as state-space search.
Risk-bounded Probabilistic Planning
Policies that optimize utility.
Observe-Act as AND-OR search.
Hidden state inference.
Expected utility given probabilistic transitions.
Games
Hidden Markov Models
Probabilistic MDPs

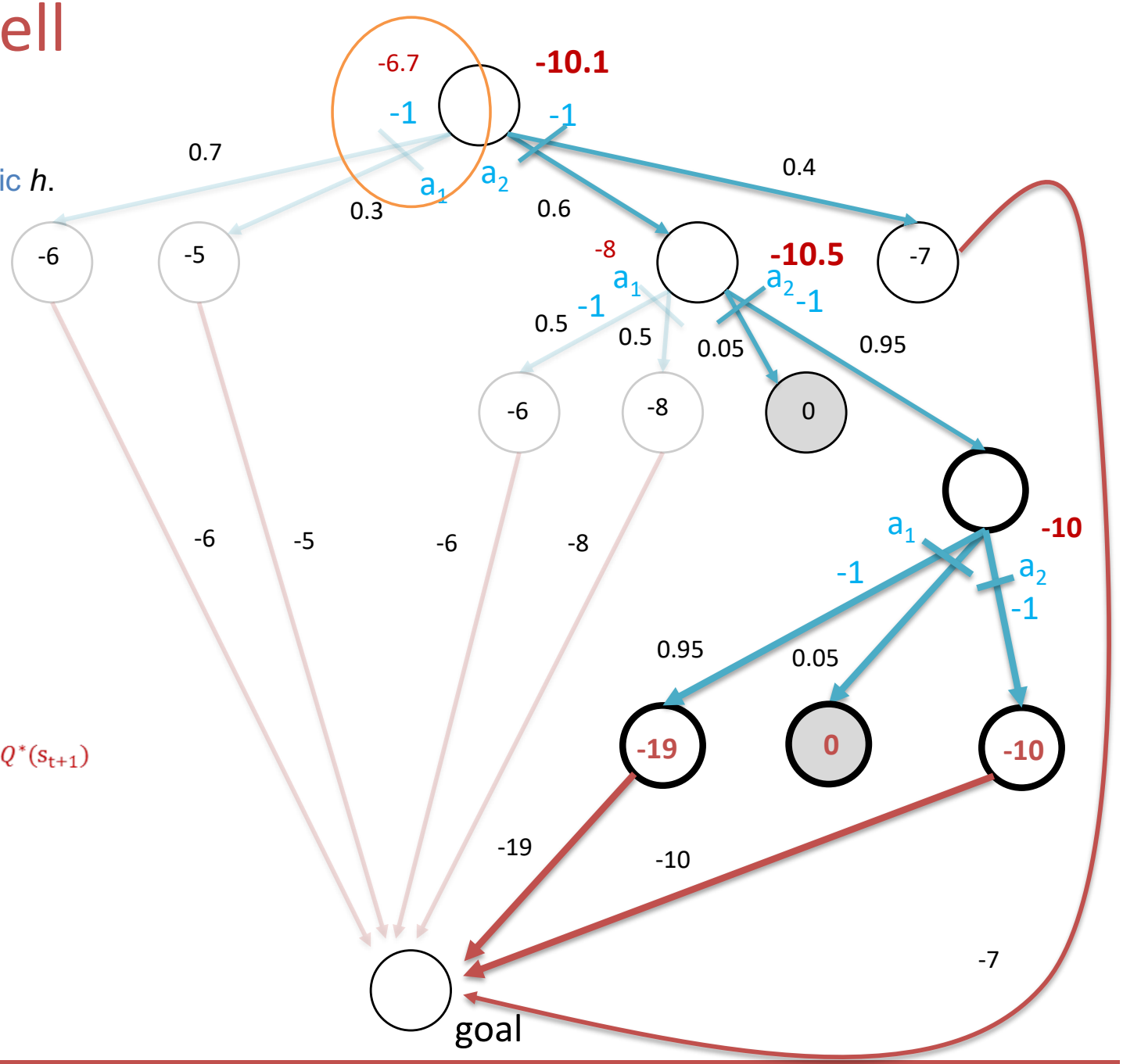
AO* in a Nut Shell

(Nilson, 1982)

Input: $< \mathbb{S}, \mathbb{A}, T, R, s_0, \mathbb{S}_g >$, heuristic h.
Output: Policy π: $\mathbb{S} \rightarrow \mathbb{A}$

1. Expand node
2. Backup value
3. Update policy

$$Q^*(s_t) = \min_{a \in A(s_t)} c(s_t, a) + \sum_{s_{t+1}} \Pr(s_{t+1} | s_t, a)\, Q^*(s_{t+1})$$



Adding bounds on risk of plan failure

$$\underbrace{\text{“Probability of violating constraint } \mathbb{c} \text{ at } \textit{some time} \text{ during execution.”} \quad \leq \Delta}_{\text{chance constraint}}$$

$$\equiv \quad \underbrace{er(s_0, \mathbb{c} \mid \pi)}_{\text{execution risk}} \leq \Delta$$

Chance-constrained MDP: $< \mathbb{S}, \mathbb{A}, T, R, s_0, \mathbb{S}_g, \mathbb{c}, \Delta >$

Execution risk of plan π

A plan (π) is "safe" from s_i onward

$\Leftrightarrow$

s_i and its successors do not violate constraint $\mathbb{c}$

Sa_i *is True when agent is safe (satisfies* $\mathbb{c}$) at the i-th step

$$er(s_k, \mathbb{c}|\pi) = 1 - \Pr\left(\bigwedge_{i=k}^{T} Sa_i | s_k, \pi\right)$$

(Ono et al., 2012)

Probability of violating constraint from step *k* onwards.

Probability of remaining safe from step *k* onwards.

Estimating execution risk

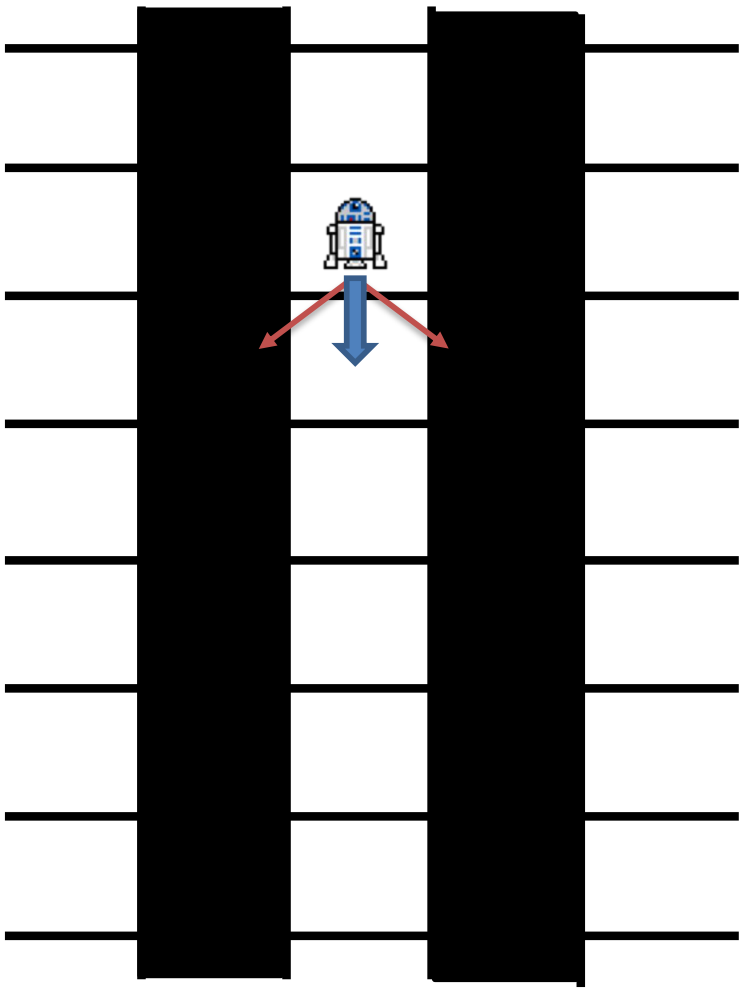
(Ono et al., 2012)

$$er(s_k, \mathbb{c}|\pi) = 1 - \Pr\left(\bigwedge_{i=k}^{T} Sa_i | s_k, \pi\right)$$

Recursive form (for backup):

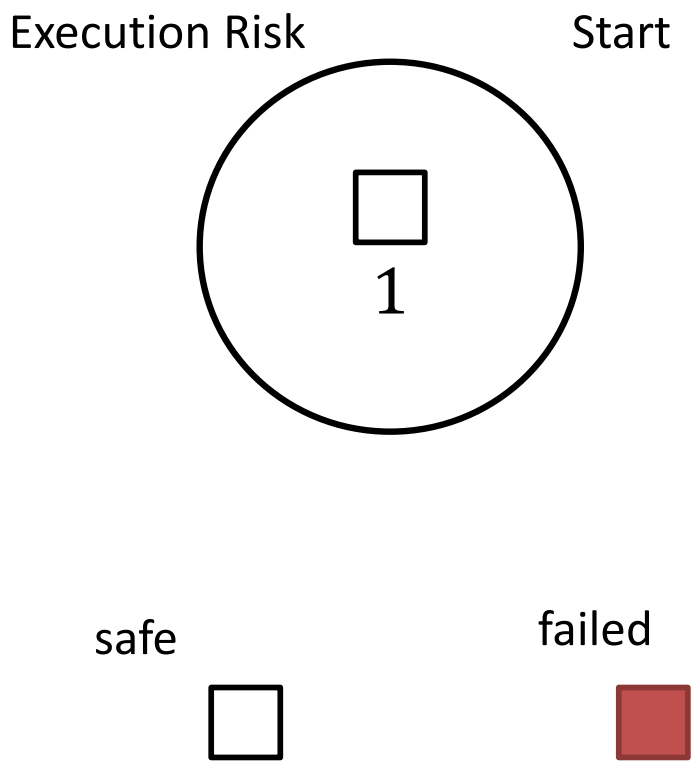
$$er(s_k|\pi) = \begin{cases} 1 & if\ s_k\ violates\ c. \\ \underbrace{\sum_{s_{k+1}} \underbrace{T(s_k, a_k, s_{k+1})}_{\text{Probability of transitioning to } s_{k+1}.} er(s_{k+1}|\pi)}_{\text{Expected risk of violating } \mathbb{c} \text{ after k.}} & otherwise. \end{cases}$$

Example: Execution risk at Start

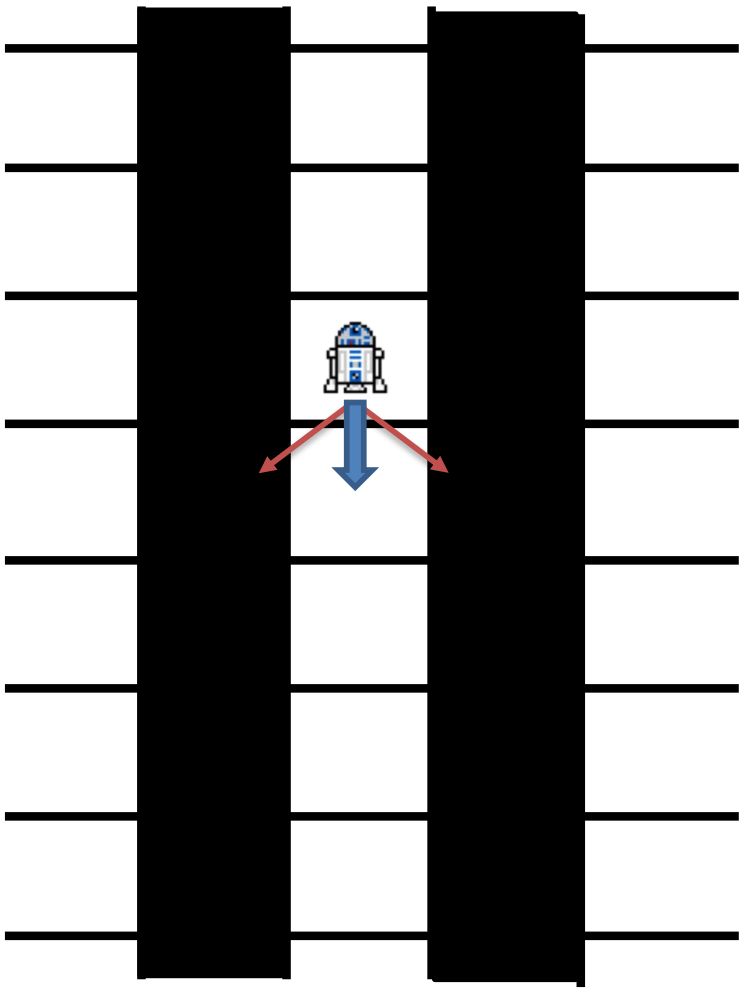


Robot model

"**R2D2** is initially safe. When it moves it reaches the desired cell with probability 90%, and slips to either side with probability 5%."

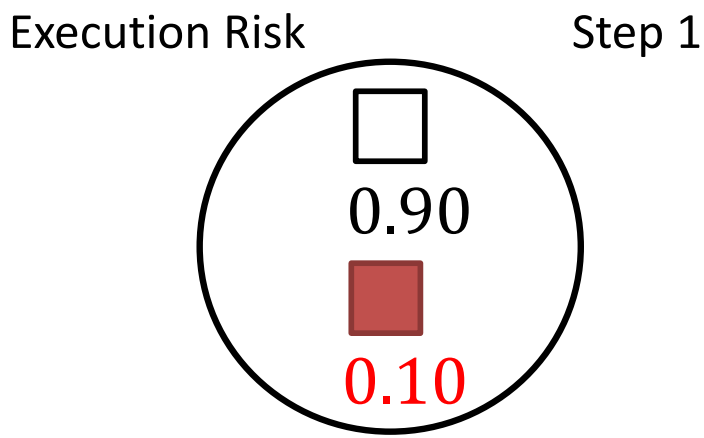


Example: Execution risk after <down>

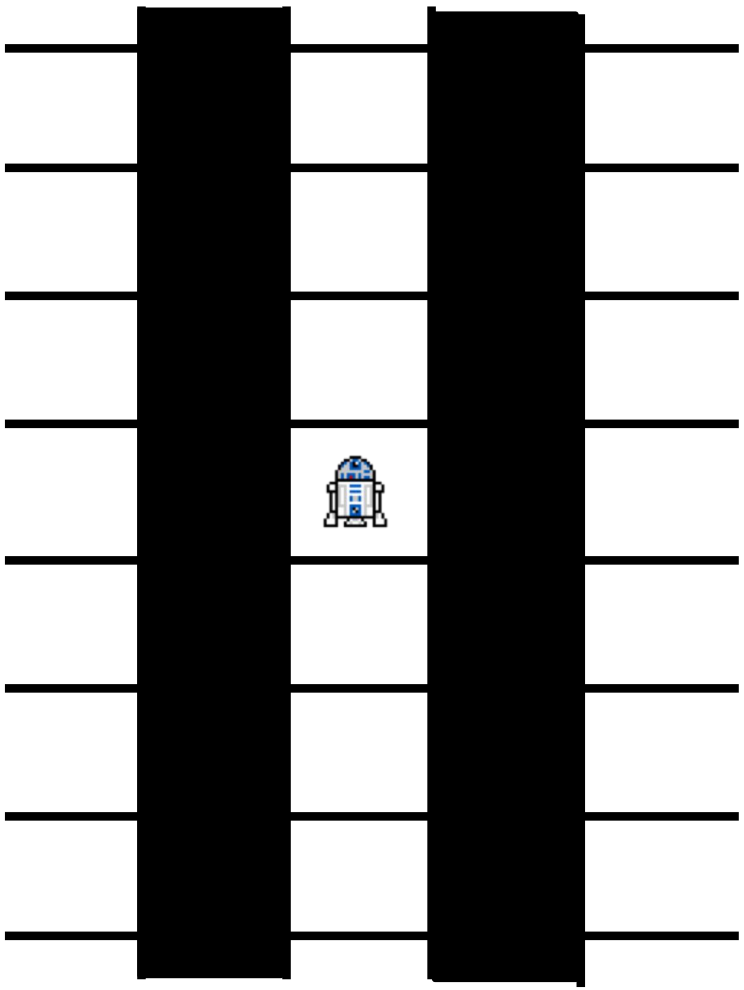


Robot model

"**R2D2** is initially safe. When **it** moves it reaches the desired cell with probability 90%, and slips to either side with probability 5%."

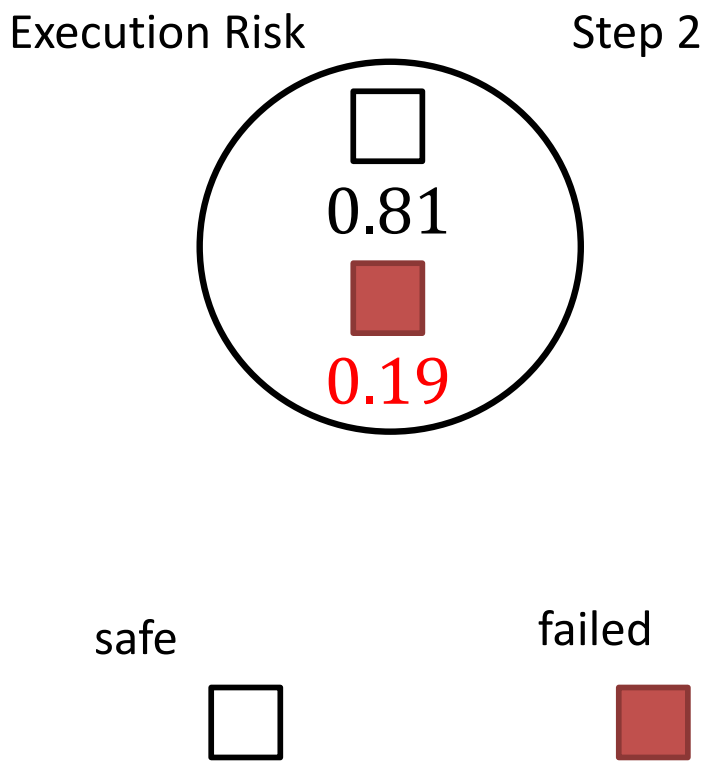


Example: Execution risk after <down, down>



Robot model

"**R2D2** is initially safe. When **it** moves it reaches the desired cell with probability 90%, and slips to either side with probability 5%."



Planning while Bounding the Risk of Failure

Idea 1: **Prune** actions that are **too risky**.

→ Safe

Idea 2: **Stop** when enough **success is accumulated**.

→ Efficient

Idea 1: **Prune** actions that are **too risky**.

lower bound h_{er} on execution risk at s_k
upper bound Δ_k on acceptable risk at s_k

$\Delta_k < h_{er}$ for action → prune

Bounding execution risk from below h_{er}

Bound: "Optimistic" estimate of future execution risk

$$h_{er}(s_k|\pi) \leq er(s_k|\pi)$$

Select to be "easy" to compute at leaves.

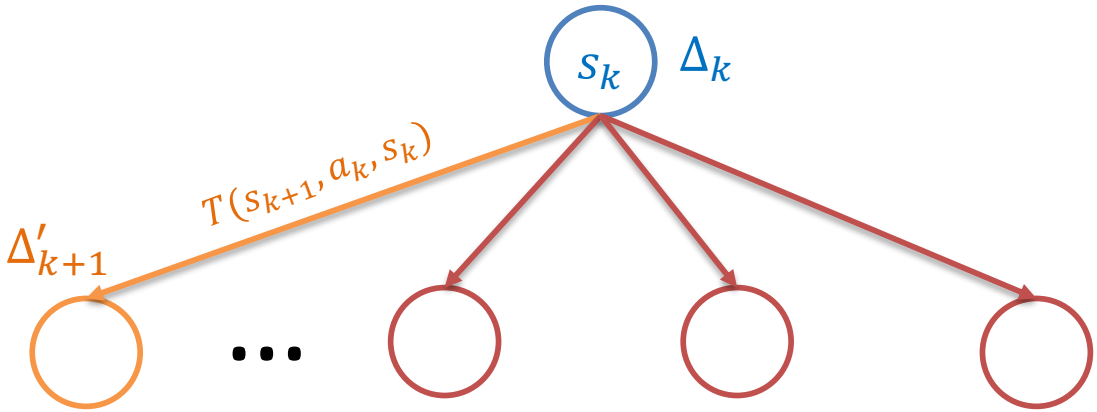
E.g., "$h_{er}(s_k|\pi) = 0 \; if \; c(s_k), else \; 1$" is always admissible

From above:

$$h_{er}(s_k|\pi) = \begin{cases} 1 & if \; s_k \; violates \; c. \\ \sum_{s_{k+1}} T(s_k, a_k, s_{k+1}) h_{er}(s_{k+1}|\pi) & otherwise. \end{cases}$$

Propagating upper bound Δ_k on acceptable risk

Idea: Child Δ'_{k+1} is the remainder of parent Δ_k ← max
not consumed by siblings (their execution risk h_{er}). ← mins



To upper bound Δ'_{k+1} . . .

. . . start with execution risk equation

. . . plug in lower bounds of siblings (h_{er})

. . . Rearrange.

$$\Delta_k = \left(T(s_{k+1}, a_k, s_k)\Delta'_{k+1} + \sum_{s_{k+1} \neq s'_{k+1}} T(s_{k+1}, a_k, s_k)\, h_{er}(s_{k+1}|\pi) \right)$$

hence:

$$\Delta'_{k+1} = \frac{1}{T(s_{k+1}, a_k, s_k)} \left(\Delta_k - \sum_{s_{k+1} \neq s'_{k+1}} T(s_{k+1}, a_k, s_k) h_{er}(s_{k+1}|\pi) \right)$$

Risk-bounded AO* in a Nut Shell

(Santana, Thiebaux & Williams, 2016)

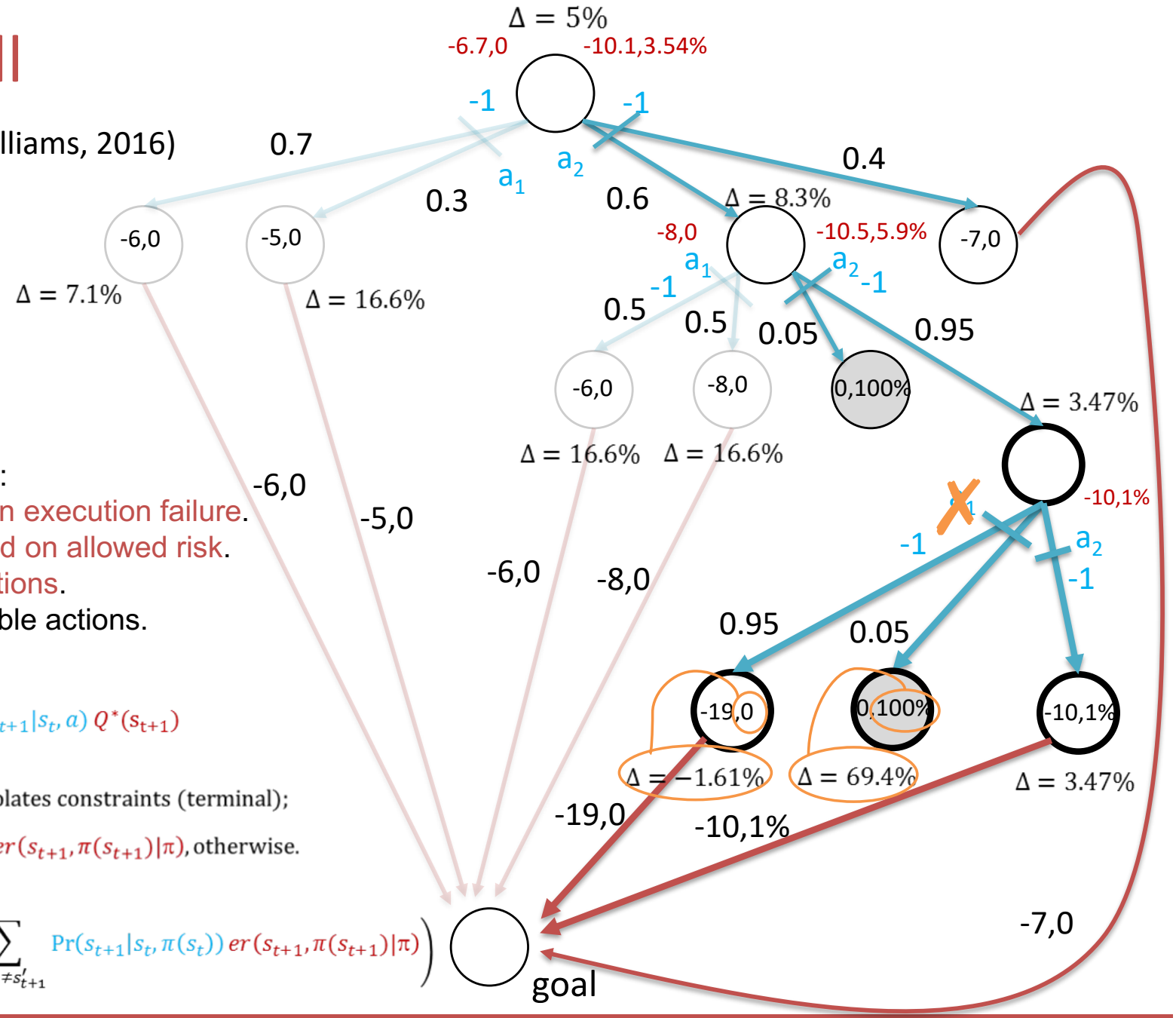
AO*:
1. Expand node
2. Backup value
3. Update policy

Risk-bounded AO* Backup:
a. Backup lower bound on execution failure.
b. Propagate upper bound on allowed risk.
c. Prune risk violating actions.
d. Backup value for feasible actions.

$$Q^*(s_t) = \min_{a \in A(s_t)} c(s_t, a) + \sum_{s_{t+1}} \Pr(s_{t+1}|s_t, a)\, Q^*(s_{t+1})$$

$$er(s_t|\pi) = \begin{cases} 1, & \text{if } s_t \text{ violates constraints (terminal);} \\ \sum_{s_{t+1}} \Pr(s_{t+1}|s_t, \pi(s_t))\, er(s_{t+1}, \pi(s_{t+1})|\pi), & \text{otherwise.} \end{cases}$$

$$\Delta'_{t+1} = \frac{1}{\Pr(s'_{t+1}|s_t, \pi(s_t))} \left(\Delta_t - \sum_{s_{t+1} \neq s'_{t+1}} \Pr(s_{t+1}|s_t, \pi(s_t))\, er(s_{t+1}, \pi(s_{t+1})|\pi) \right)$$

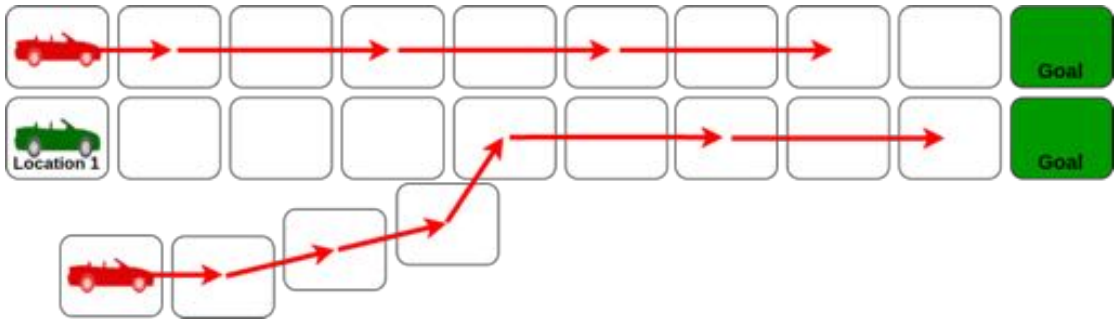


Representative scenario

Discretized actions near an on ramp

Highway on and off ramps have high accident rates

Take Low Risk Scenario

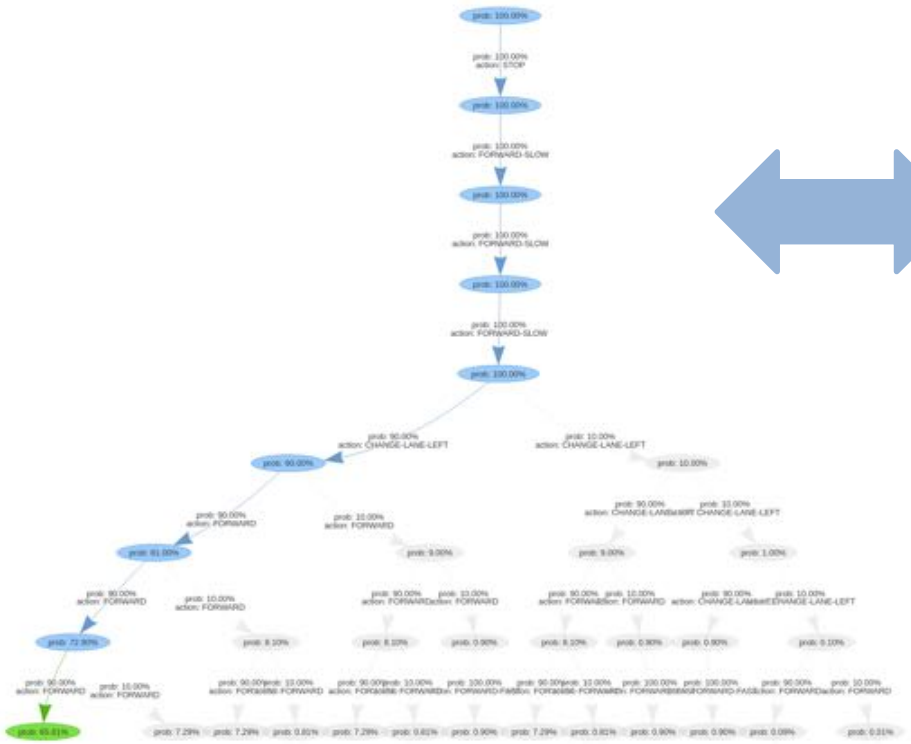


1. Our driver in the right lane,
2. a vehicle entering on ramp,
3. a passing vehicle.

Risk of near collision bounded to 0.001.

Resulting Conditional Plan

Take Low Risk Scenario



Most Likely Maneuver Sequence:

DECELERATE-TO-SLOW

CONTINUE-FORWARD x 3

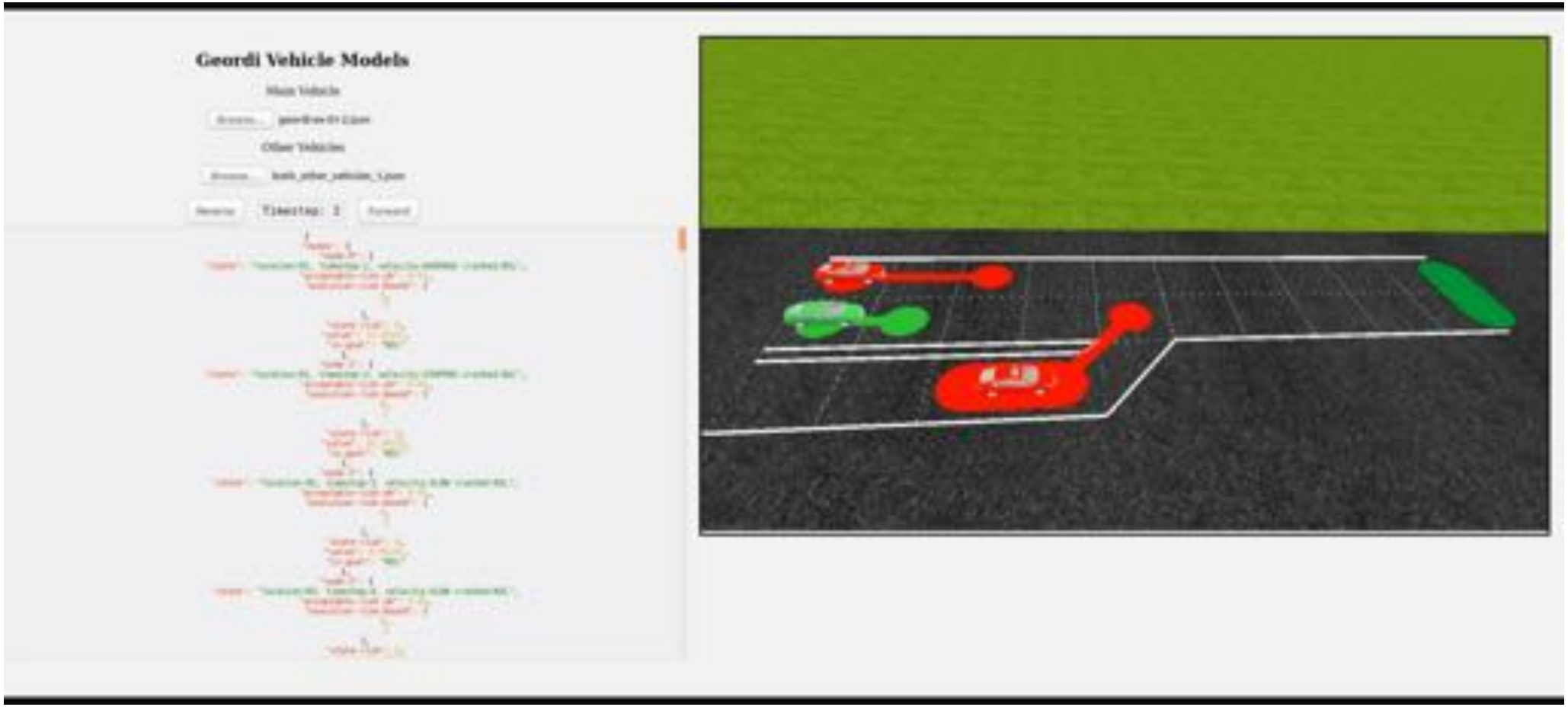
LEFT-LANE-CHANGE

ACCELERATE-TO-AVERAGE

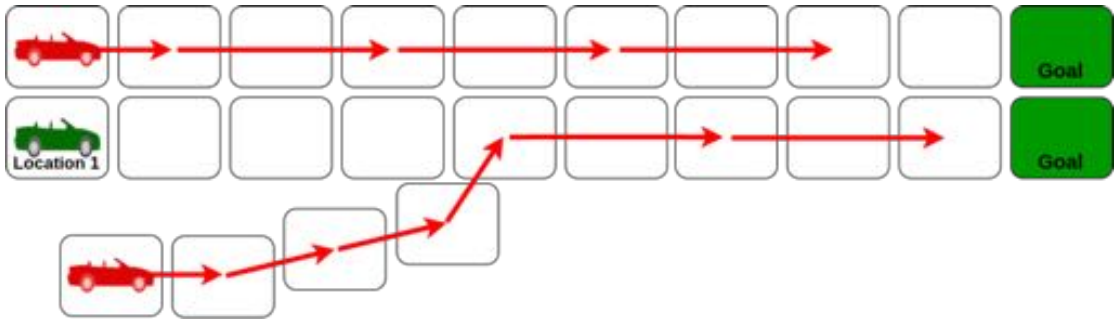
CONTINUE-FORWARD

https://www.youtube.com/watch?v=8WqO3KeYHj4

Take Low Risk Scenario



Take High Risk Scenario



Same scenario

Increase: Risk of near collision bounded to 0.70

Resulting Conditional Plan

Red nodes represent predicted collision states

Increasing risk bound to 0.70 produces more aggressive maneuvers, with up to ~ 50% chance of collision.

Take High Risk Scenario

Outline

- Diving into the Real World
- Risk-bounded Planning as Optimization and Constraint Programming
- Generative Task and Motion Planning with Heuristic Forward Search
- Risk-bounded Planning by adding And / OR (AO*) Search

Some takeaways

1. *Execution risk* applies broadly to risk-aware planning
   - Incorporate into many types of planners to endow with a keen sensitivity to risk;

2. Risk-bounded planning improves utility over risk-minimizing alternatives, while offering strict safety guarantees;

3. Efficient risk-aware constraint solvers are required for risk-aware planning;

Questions?