Massachusetts Institute of Technology

Case Study: Personal Transportation System

16.410-13

October 26th, 2011

Masahiro Ono, Peng Yu

Model-based Embedded & Robotic Systems

CSAIL

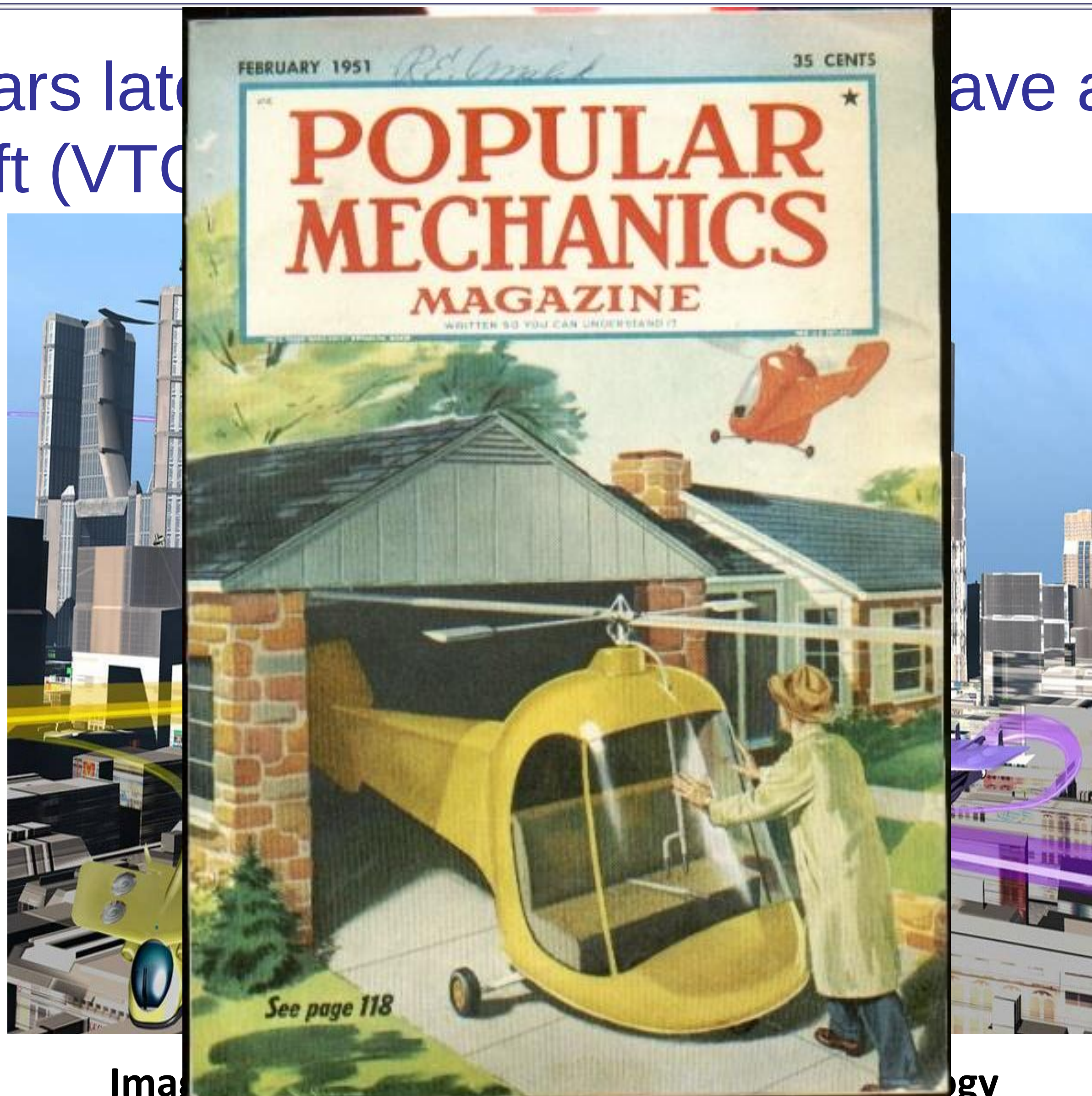
Reminder

- 16.413 Project Part 1:
  - Out last Wednesday.
  - Due Nov,14th.

- Mid-term:
  - Monday Oct, 31st, Halloween.
  - 1 letter-size help sheet, print or hand-written.
  - 9:30am, Rm 33-419.
  - 85 minutes.

Motivation

- 50 years lat… ave a personal aircraft (VTO…



Imag… gy

Motivation

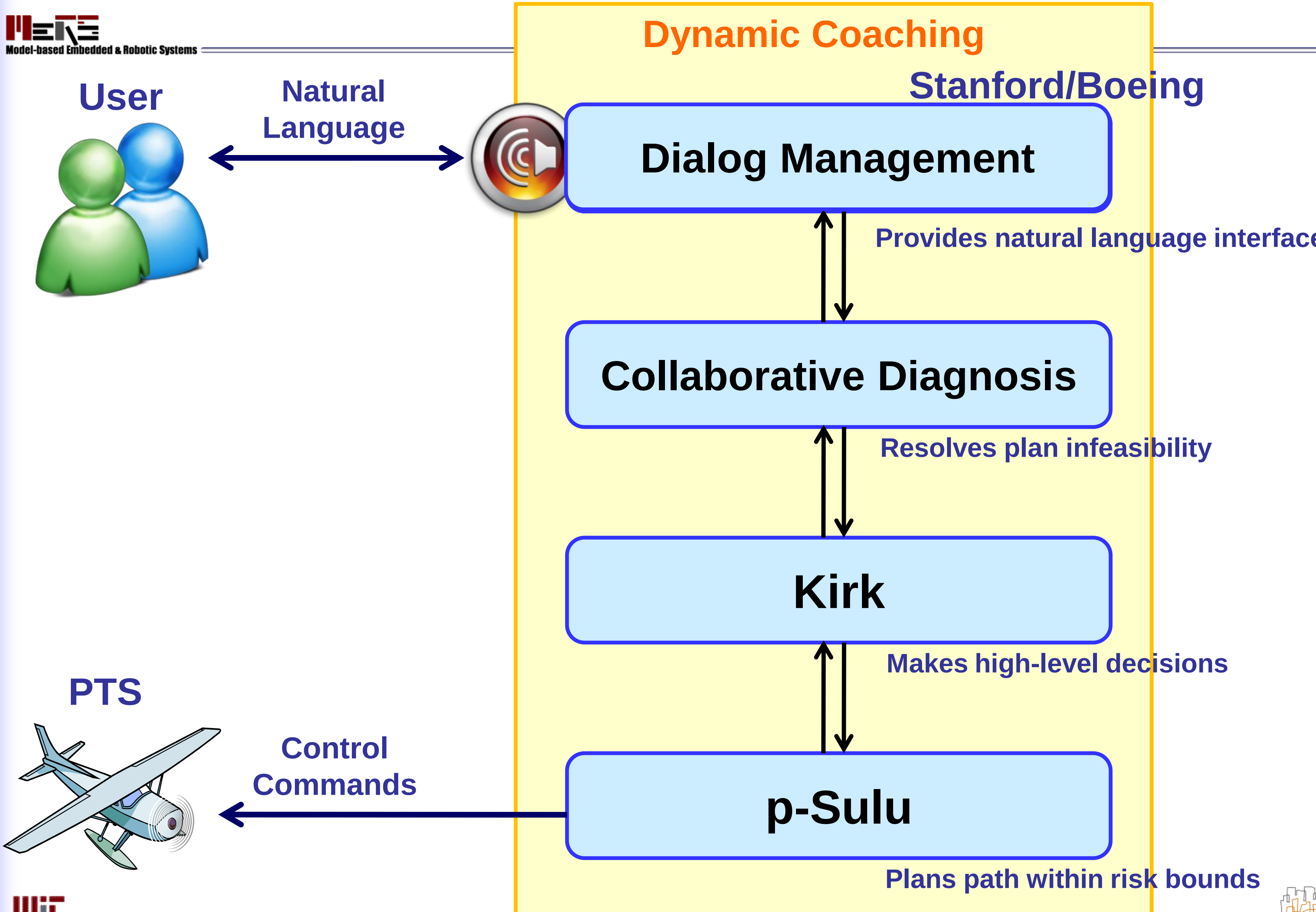
- However, flying aircraft is not easy:
  - Single Engine: 3 months
  - Multiengine Commercial: 6 months
  - Helicopter: 3 months

- Create a highly automated vehicle:
  - Provides point-to-point transportation like a taxi
  - Must be robust to uncertainty
  - Taxi driver!

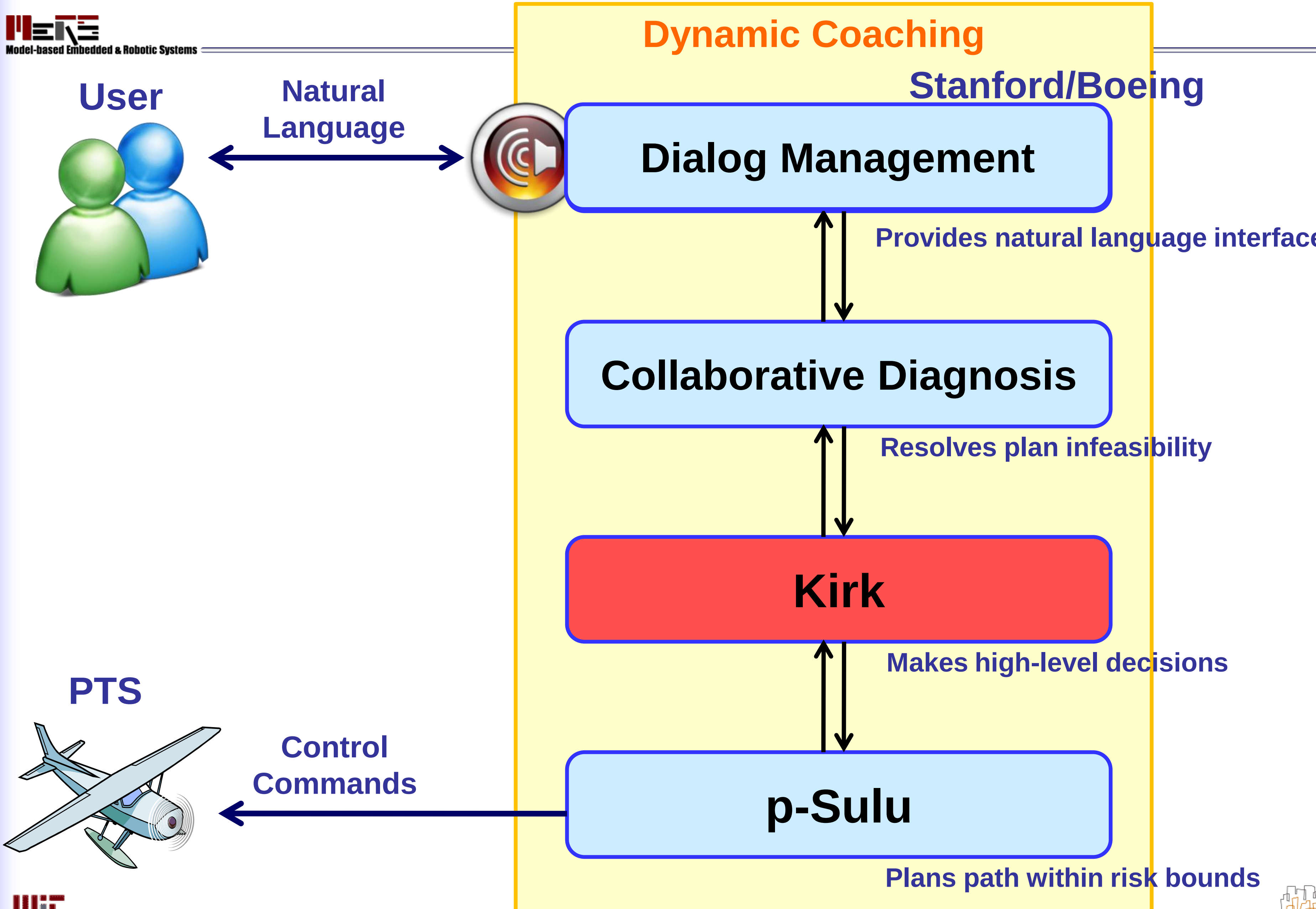
Demo

- The Personal Transportation System with X-Plane Simulation.

System Architecture



System Architecture

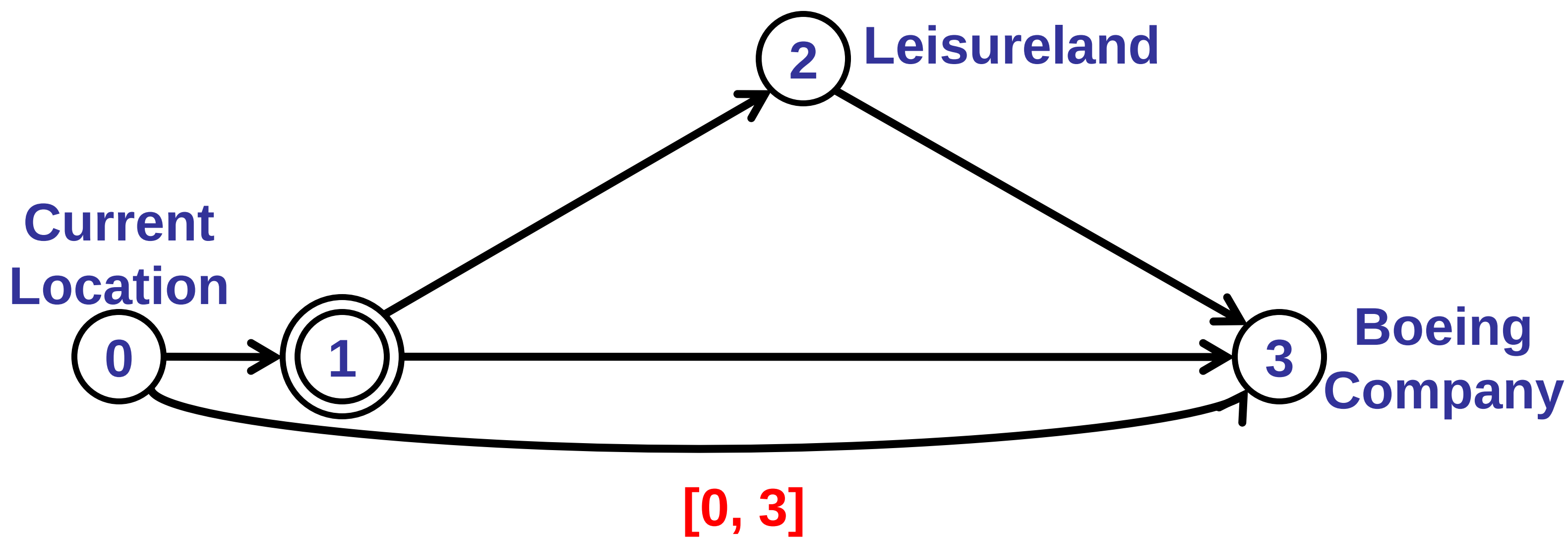


Generate Temporal Plan

- Convert user requirements into temporal plan.
  - I want to go to the Boeing company.
  - I want to be there in 3 minutes.
  - I want to use Harvey Field as backup landing sites.
  - I want to stop at Leisureland if possible.

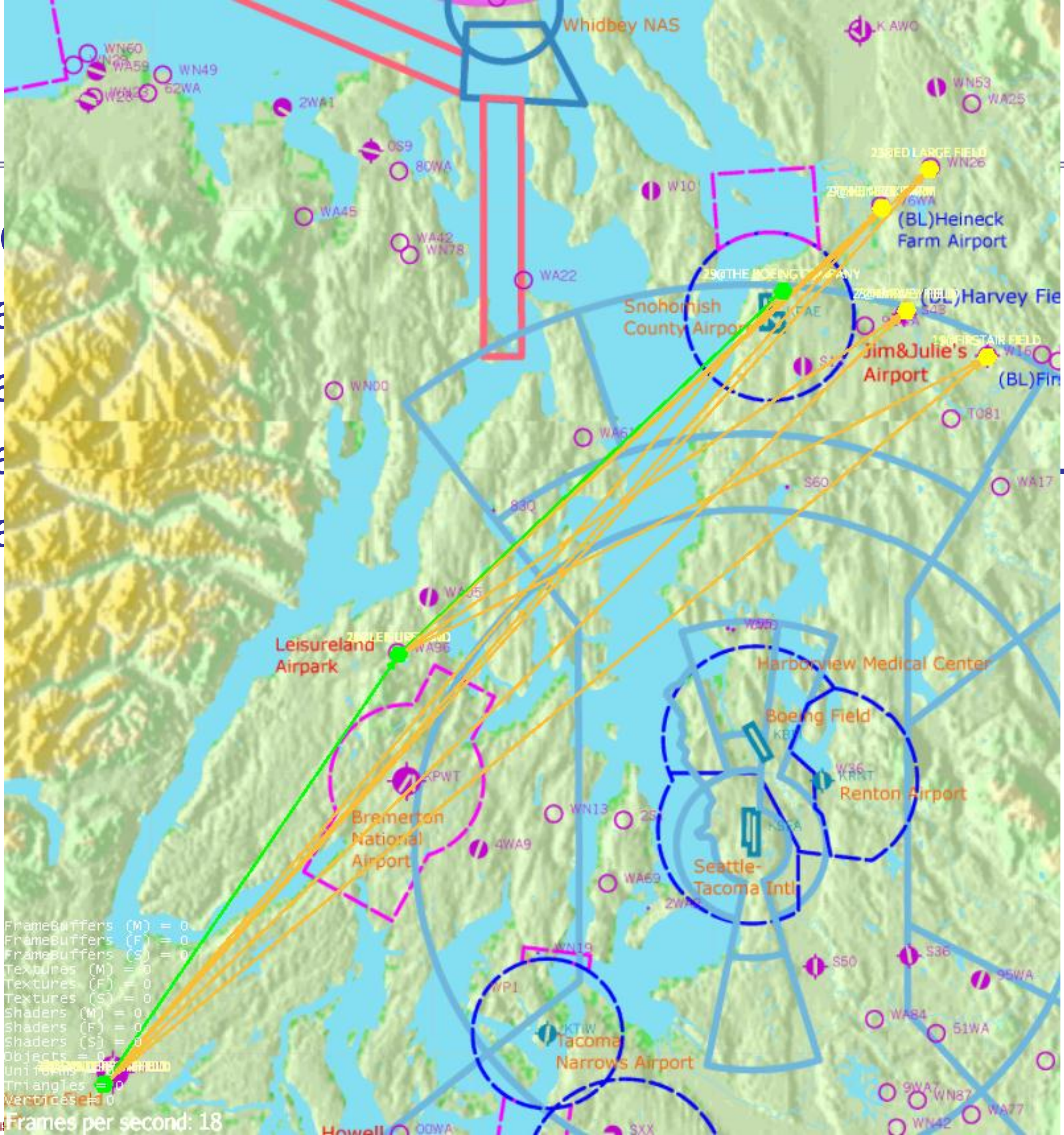
Generate Temporal Plan

- Convert user requirements into temporal plan.
  - I want to go to the Boeing company.
  - I want to be there in 3 minutes.
  - I want to use Harvey Field as backup landing sites.
  - I want to stop at Leisureland if possible.



Model-based Embedded & Robotic Systems

- Conv
  - I wa
  - I wa
  - I wa .
  - I wa

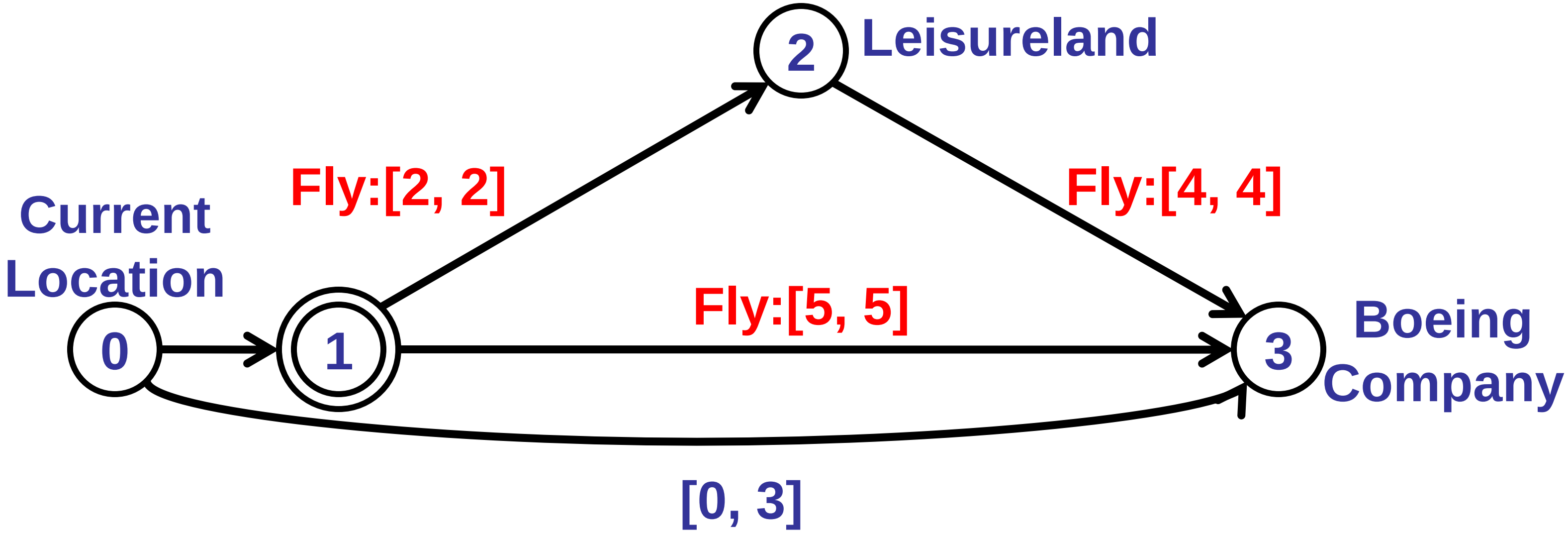


CSAIL

Generate Temporal Plan

Model-based Embedded & Robotic Systems

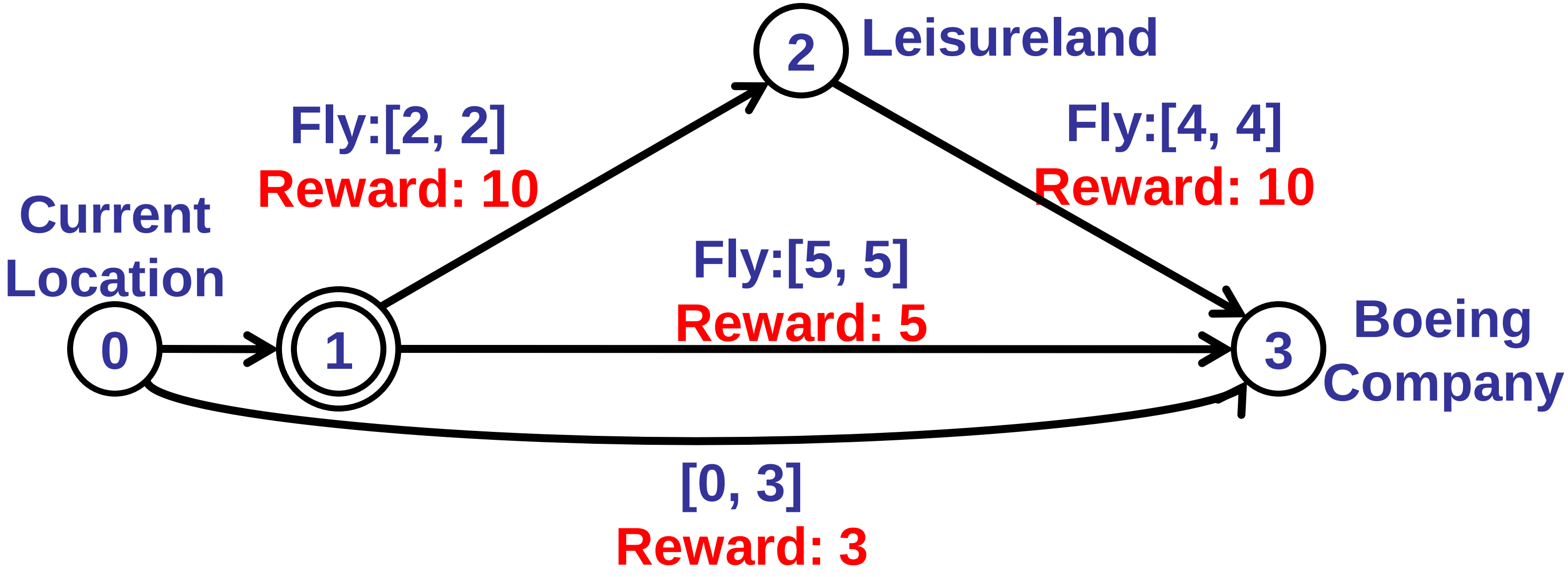
- Convert user requirements into temporal plan.
- Estimate the flight durations.



Generate Temporal Plan

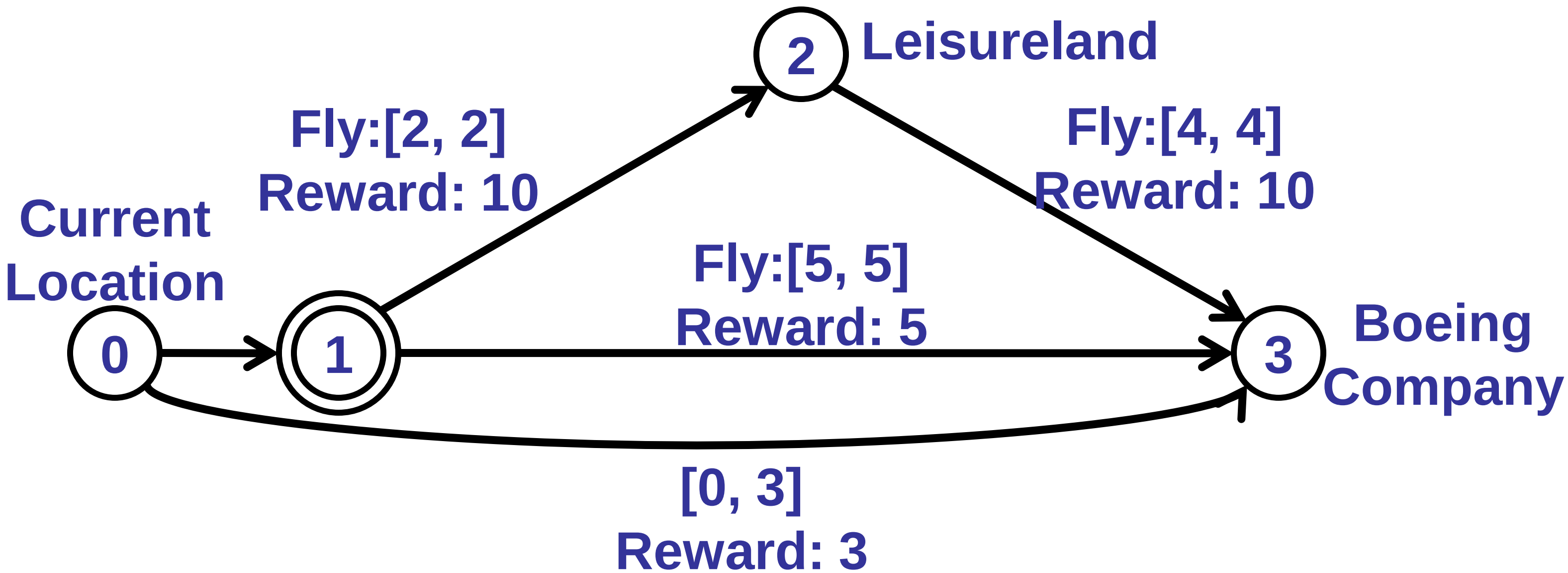
Model-based Embedded & Robotic Systems

- Convert user requirements into temporal plan.
- Estimate the flight durations.
- Add user preferences.



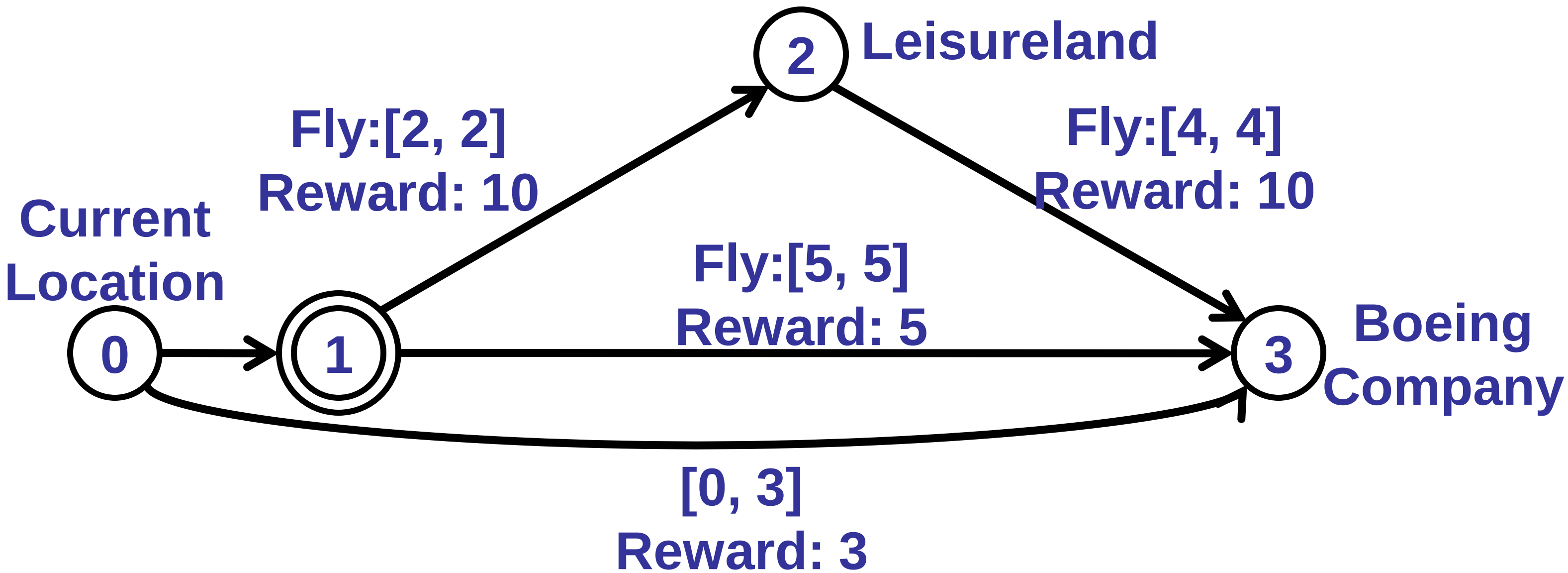
Temporal Plan Network (Kim, Williams and Abramson, 2001)

- Augmented from Simple Temporal Networks.
  - Addition of decision nodes.
  - Rewards/costs.
  - Symbolic constraints.



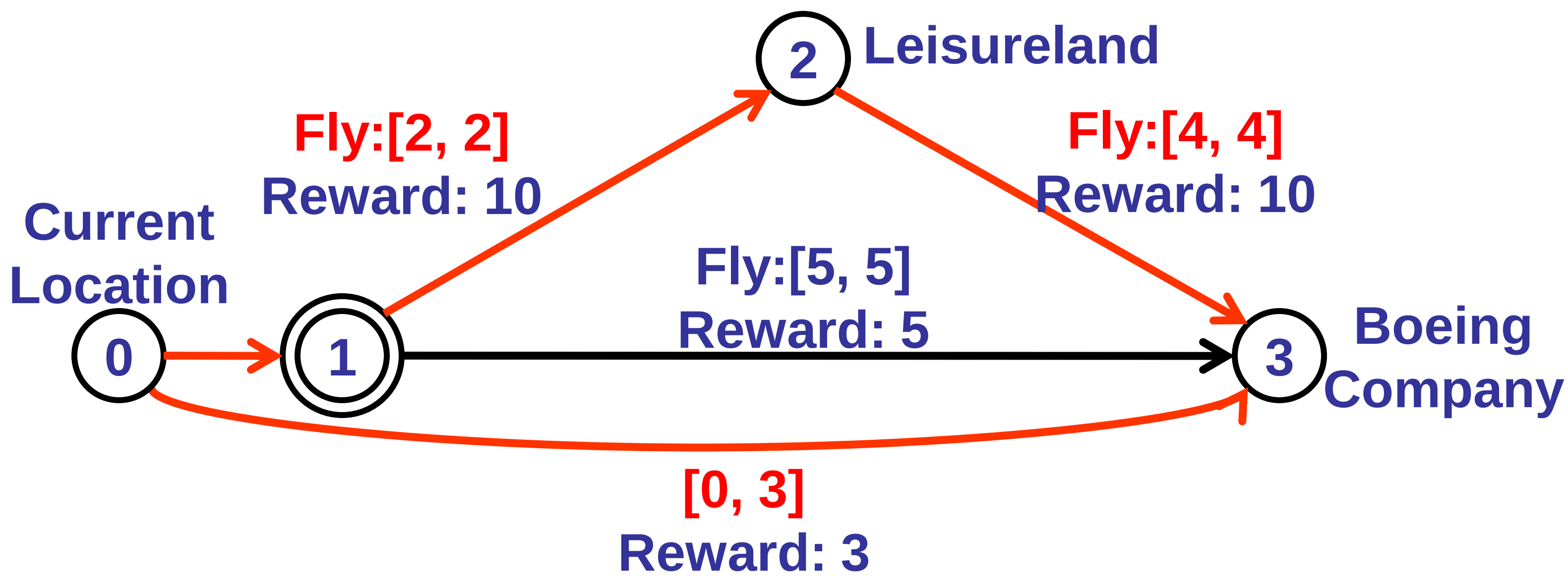
Solve a TPN

- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency.
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



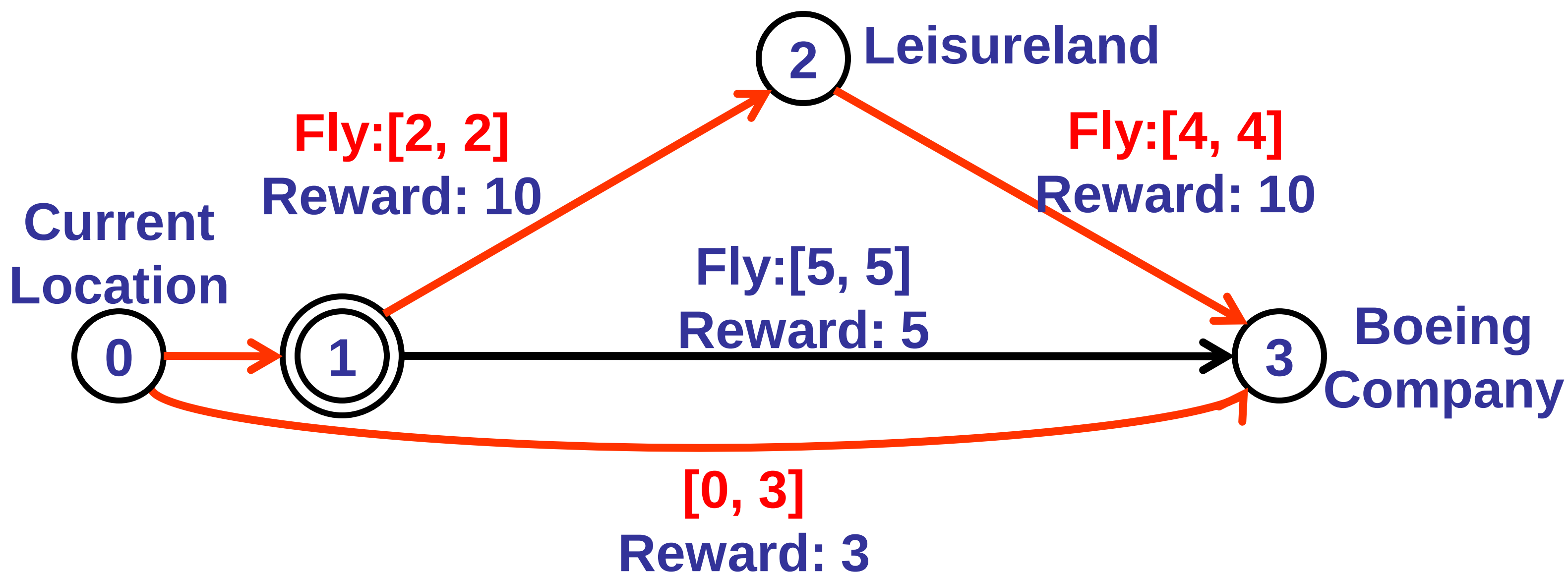
Solve a TPN

- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency.
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



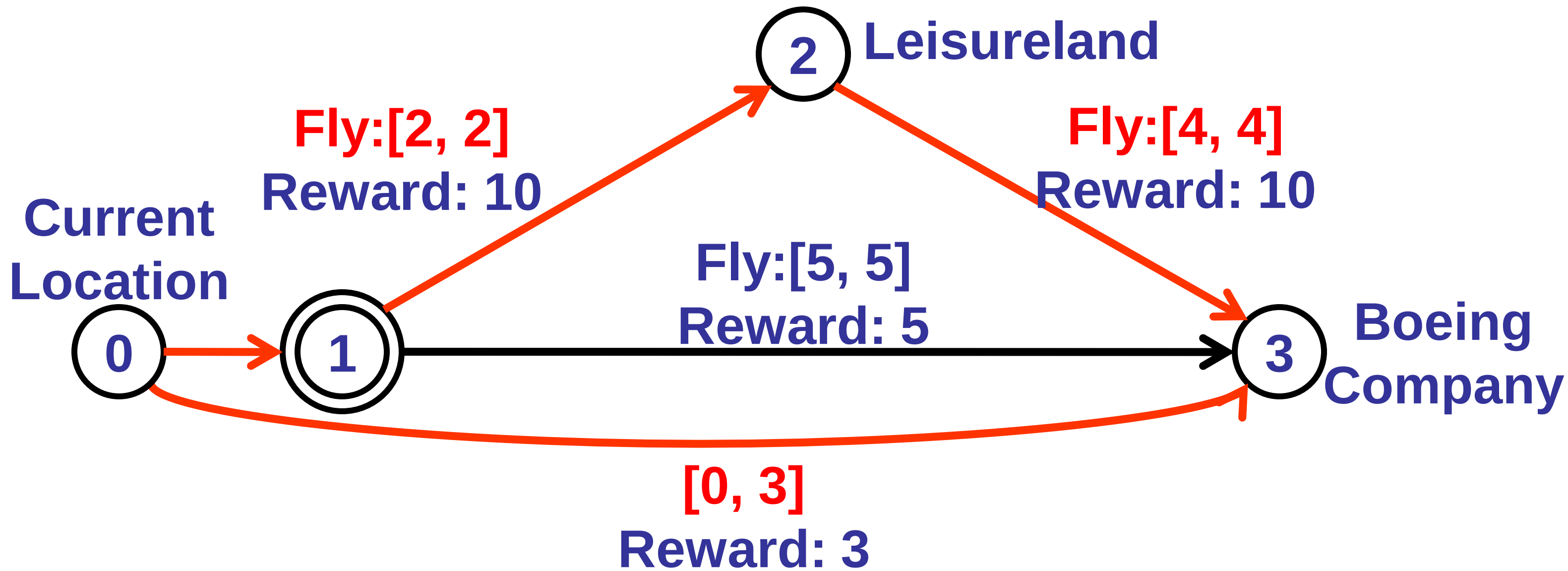
Solve a TPN

- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency.
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



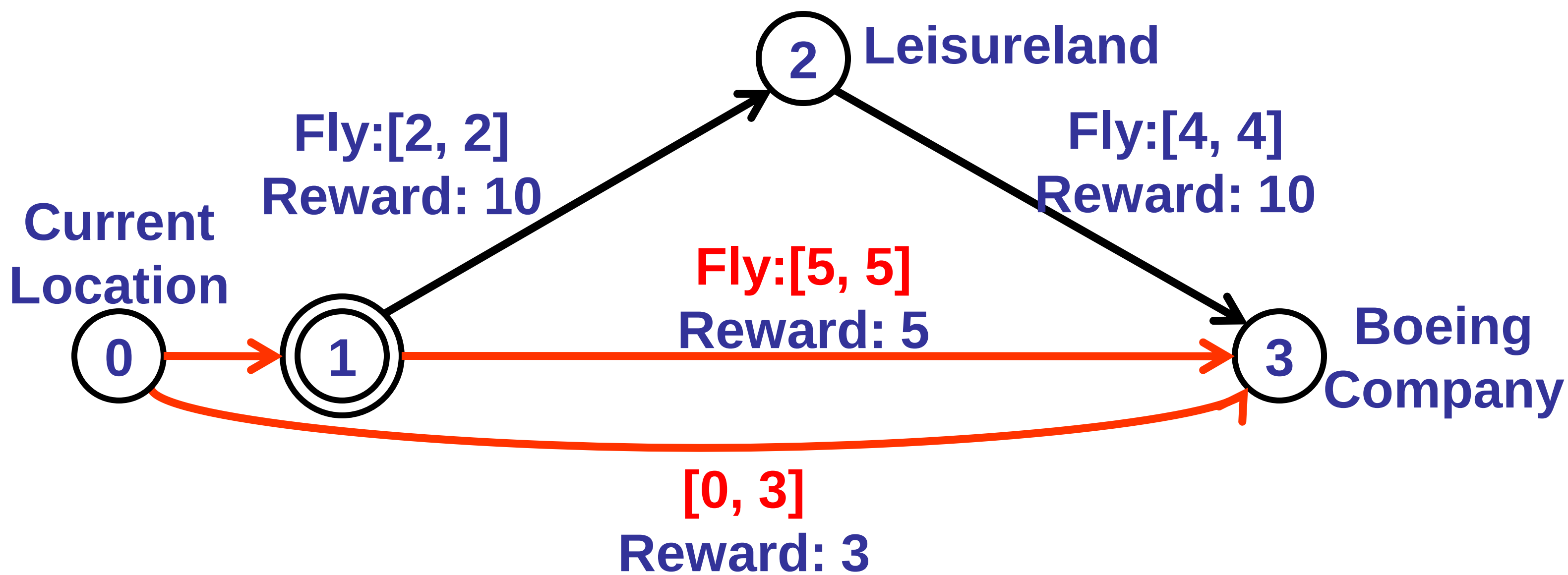
Solve a TPN

- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency. Not consistent!
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



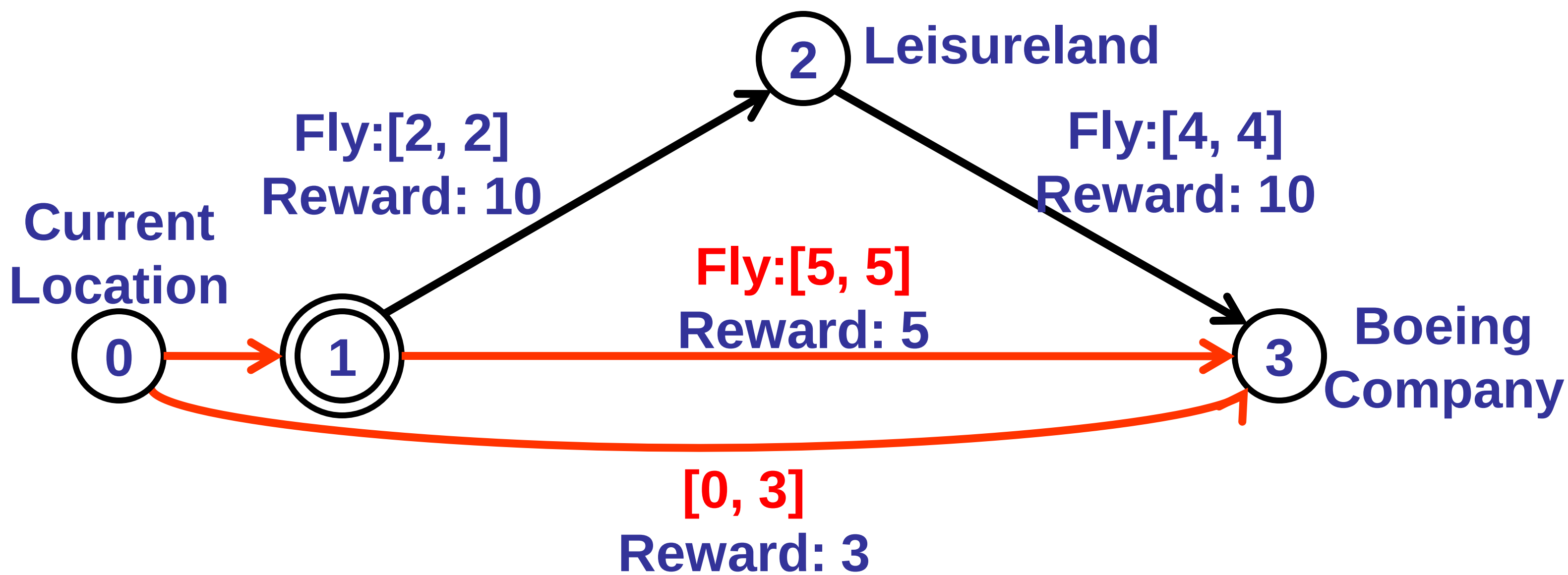
Solve a TPN

- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency.
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



Solve a TPN

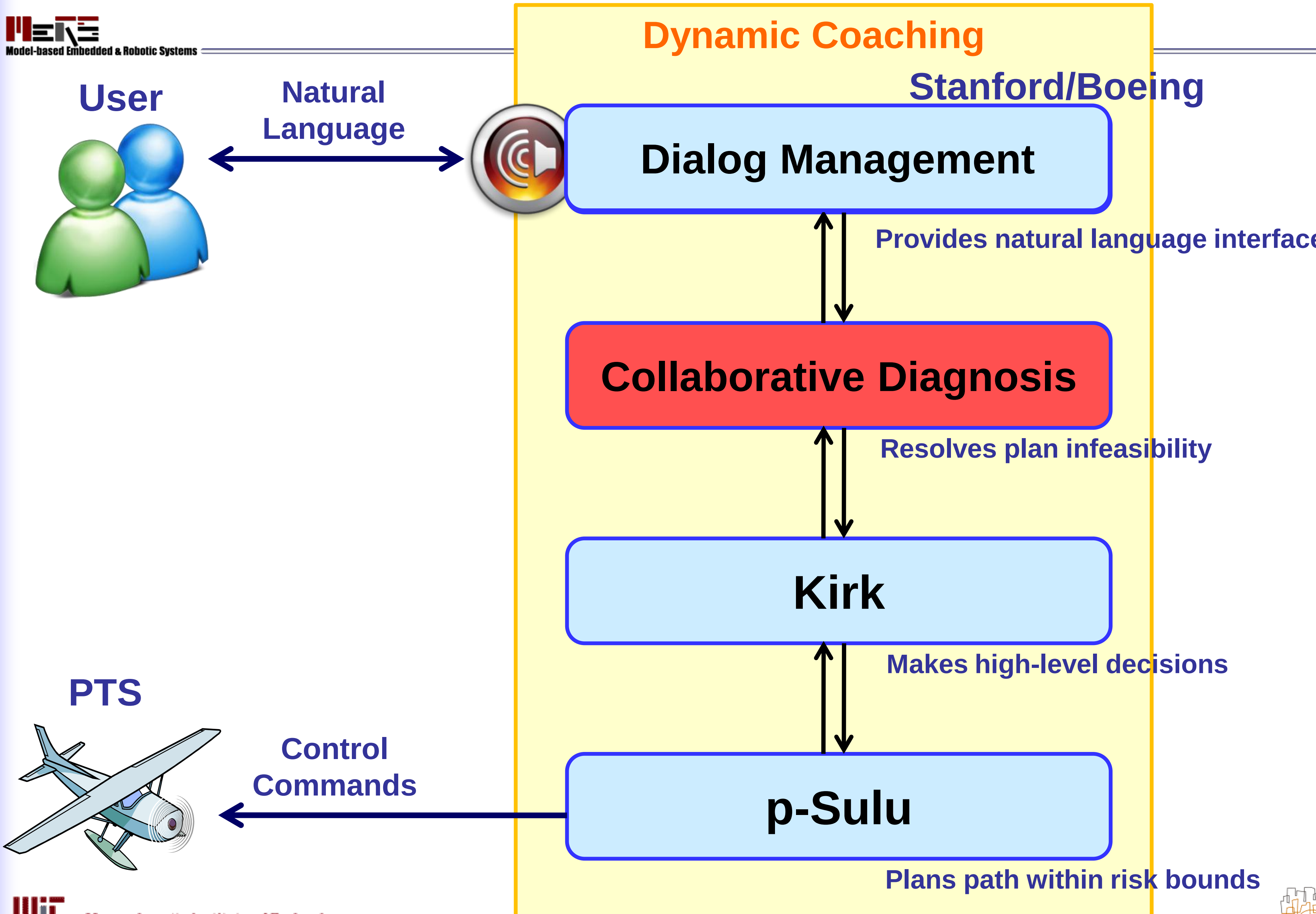
- To find the most preferred/least cost plan.
  - Generate the best candidate.
  - Check temporal consistency. Not consistent!
  - Return solution (if candidate consistent) or start over (generate the next best candidate).



What if no solution exists…

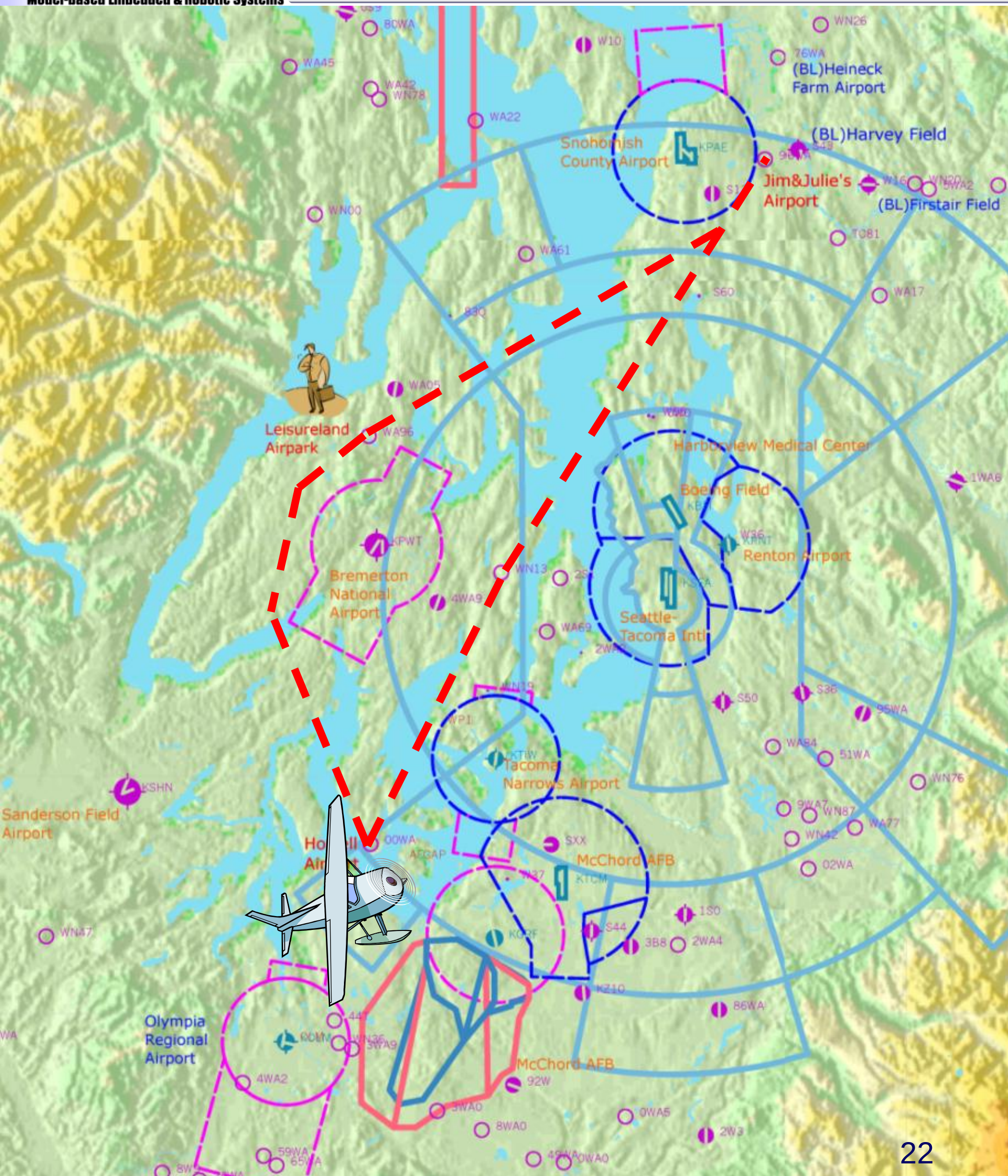
- Tell the user I cannot find a solution.
- Let the user figure out the problem and input a new set of requirements.
- OR
- Diagnose the over-constrained plan and find a relaxation for the user.
  - "If you relax your constraints or fly faster, I can find a feasible plan for you."

System Architecture

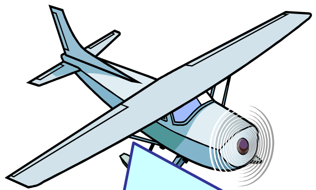


In the PTS Scenario

Model-based Embedded & Robotic Systems



PTS



User

You cannot get there in 3 minutes but you can get there in 6 minutes.

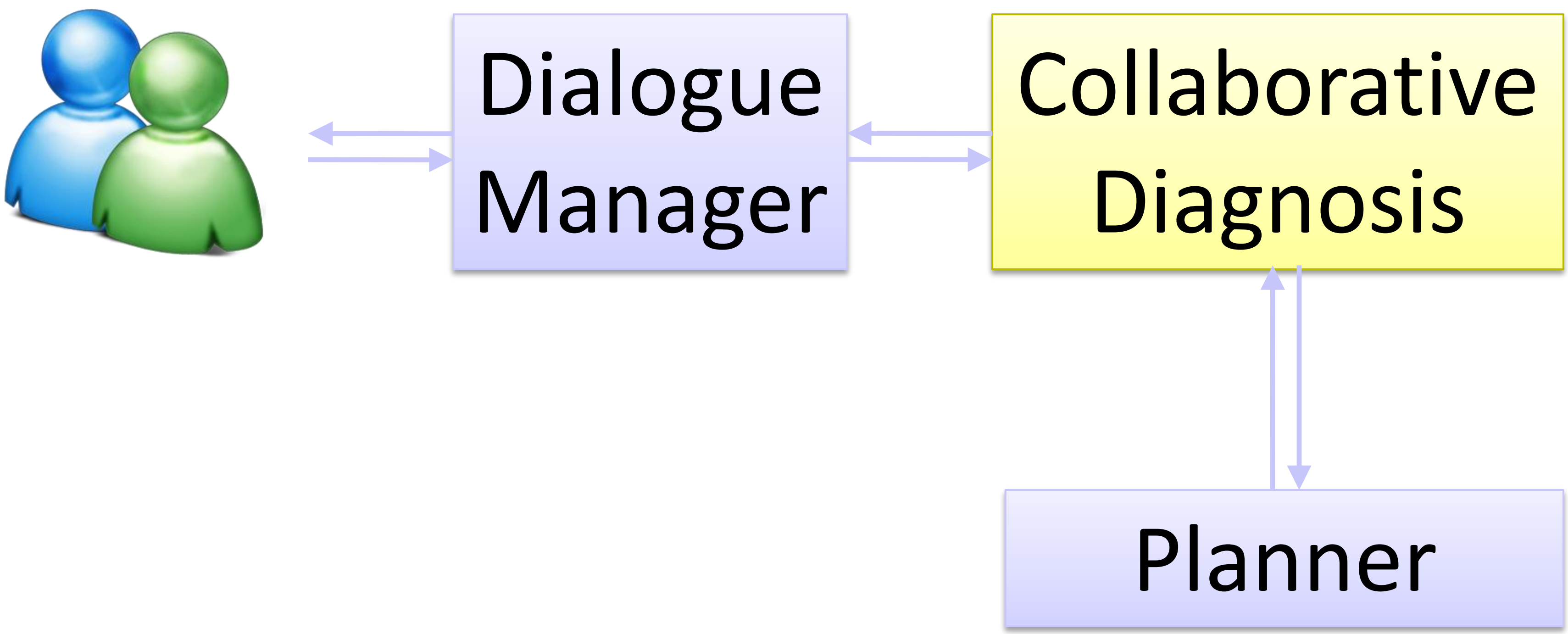
Collaborative Diagnosis:

- Generate plan.
- Detect and diagnose conflicts.
- Present diagnoses and repair options to user.

CSAIL

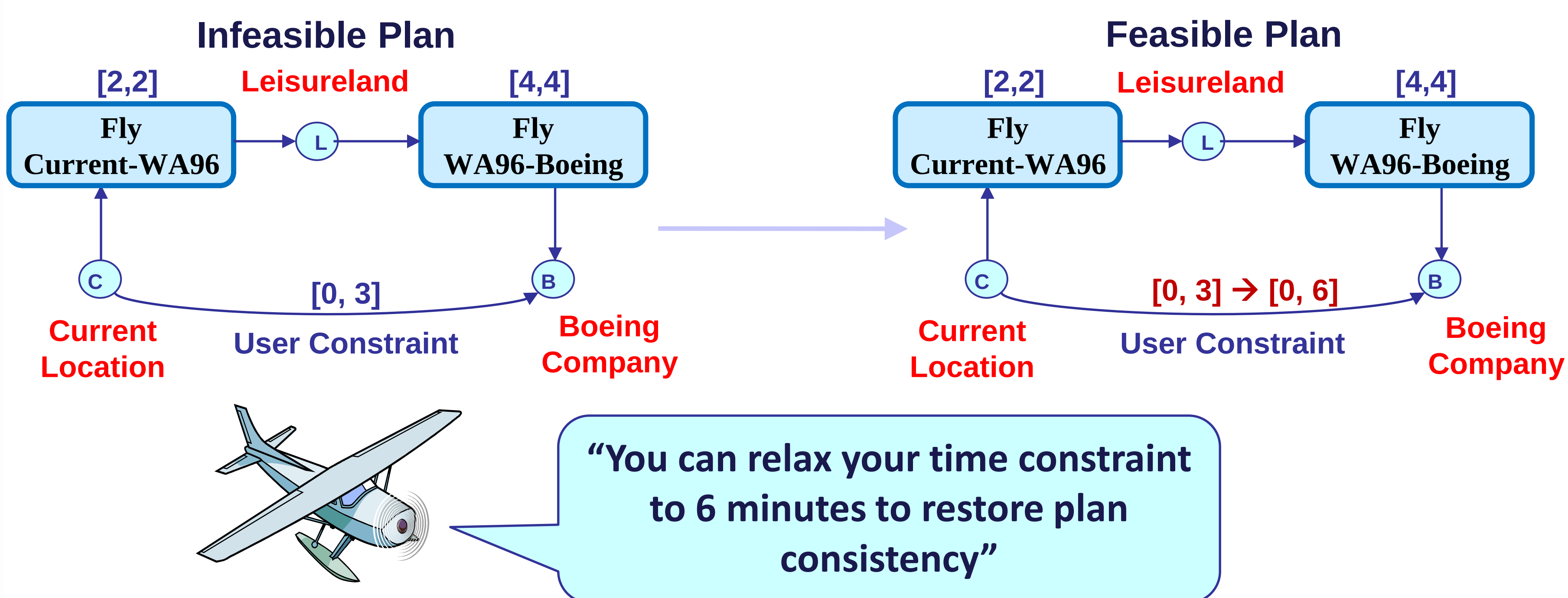
Collaborative Diagnosis - Introduction

- Definition
  - An interface between the computer and the user.



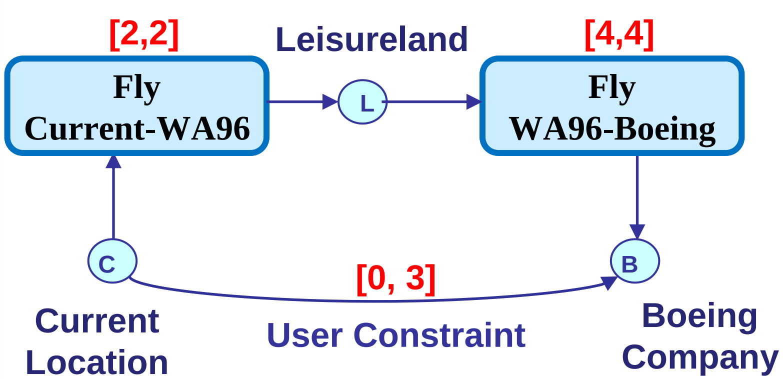
Collaborative Diagnosis - Introduction

- Definition
  - An interface between the computer and the user.
- Objective
  - Help the user resolve infeasible plans.



Challenge and Key Idea

- Challenge: Too many options to take.
- Key Idea: Implement the diagnosis concepts and reduce the size of results by intelligently pruning meaningless options.
  - Current-WA96 →{IN, OUT}.
  - WA96-Boeing →{IN, OUT}.
  - Current-Boeing →{IN, OUT}.



Working Principle

Why is the plan infeasible?

↓

How to repair the plan?

↓

What is the best way to repair?

Identify the Cause of Failure

Generate minimal perturbations to the goals

Present the user with possible options

Working Principle

Why is the plan infeasible?

Identify the Cause of Failure

How to repair the plan?

Generate minimal perturbations to the goals

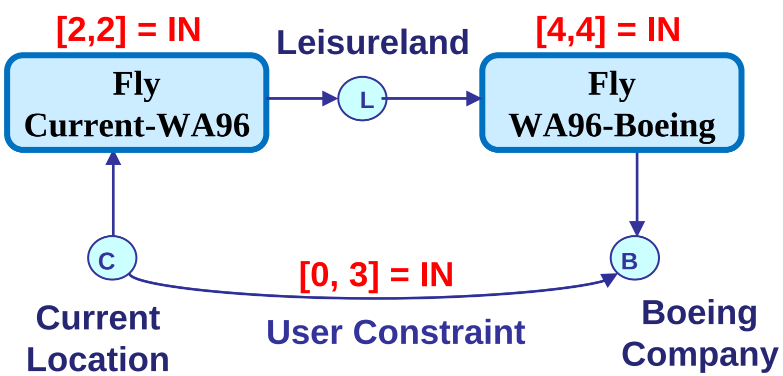
What is the best way to repair?

Present the user with possible options

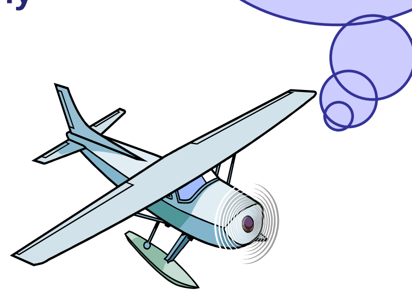
Identify Cause of Failure

Why is the plan infeasible?

We employed Conflict-directed A* algorithm to find and resolve the conflicts that cause inconsistency.



The user constraint is in conflict with the flight duration……



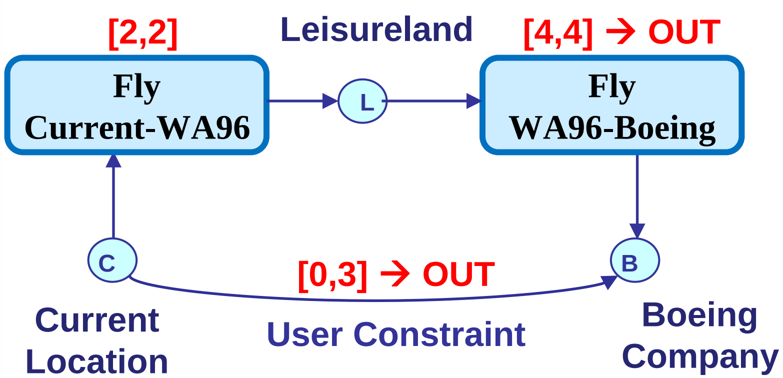
Working Principle

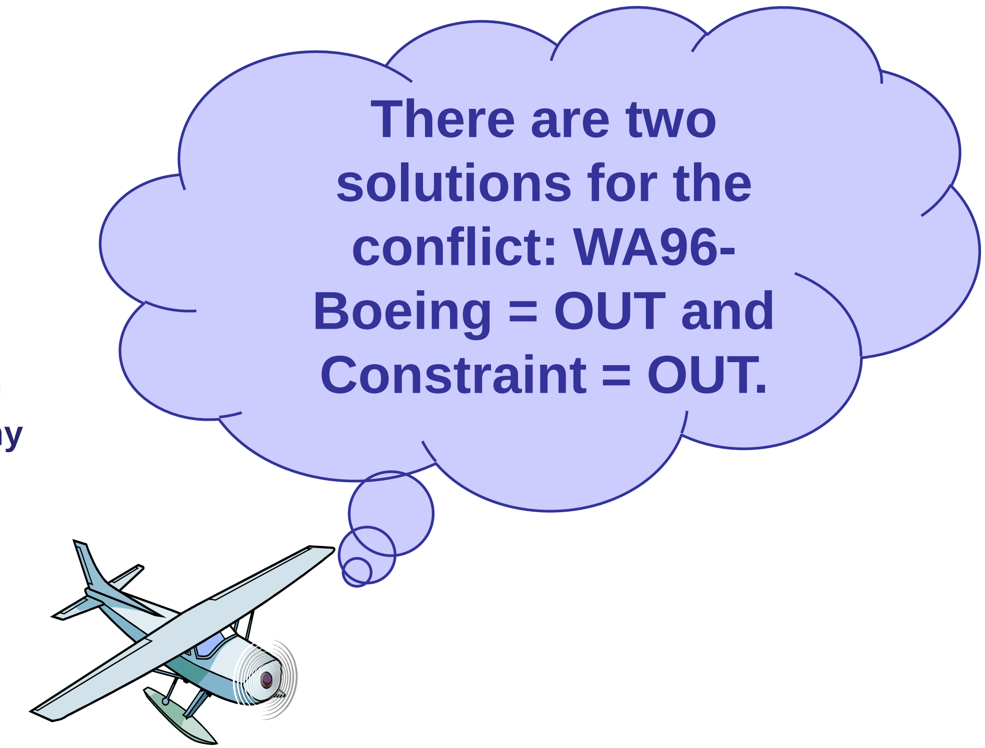
Why is the plan infeasible?	**Identify the Cause of Failure**
How to repair the plan?	**Generate minimal perturbations to the goals**
What is the best way to repair?	**Present the user with possible options**

Generate Possible Options

How to repair the plan?

First, we resolve the conflicts by removing constraints (assign "OUT").

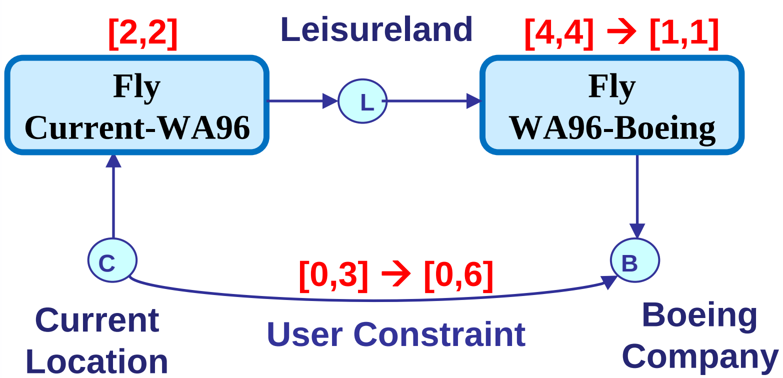




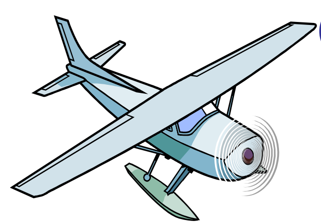
Generate Possible Options

How to repair the plan?

Second, we calculate the minimal relaxation for the removed constraints.



WA96-Boeing and Constraint can be relaxed to [1, 1] and [0,6].



Working Principle

Why is the plan infeasible?

↓

How to repair the plan?

↓

What is the best way to repair?

Identify the Cause of Failure

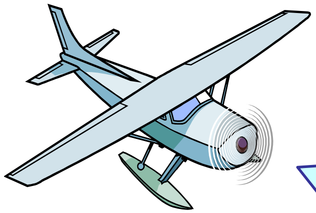
Generate minimal perturbations to the goals

Present the user with possible options

Present Results

What is the best way to repair?

We present possible options to the user and let the user decide if they want to execute.



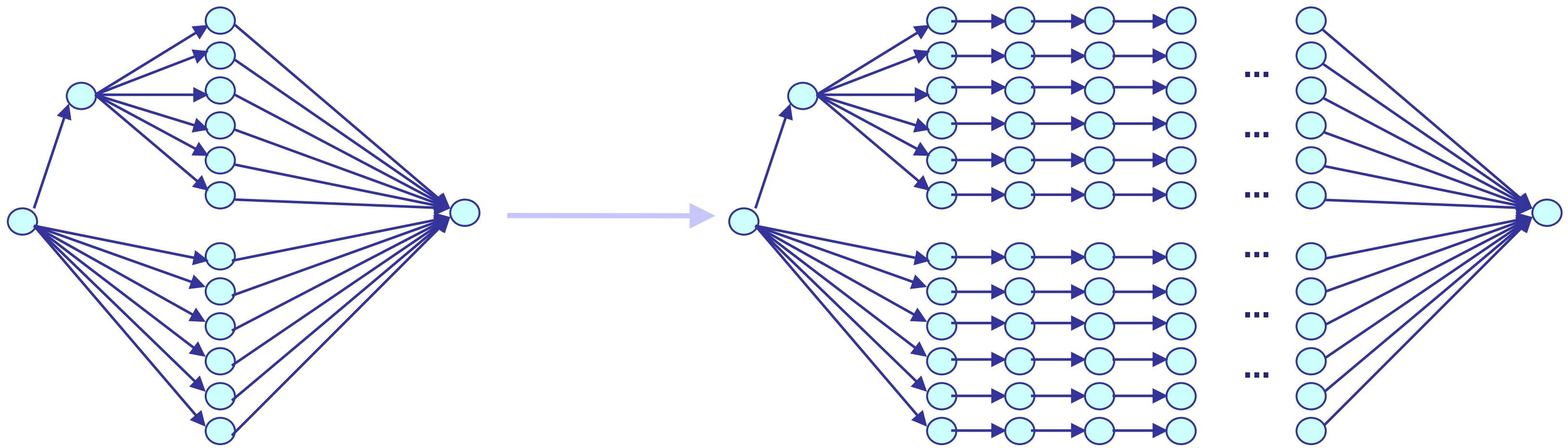
I found two options for you, which one do you prefer?

"Relax your time constraint to 6 minutes"

"Fly from Leisureland to the Boeing Company in 1 minute"

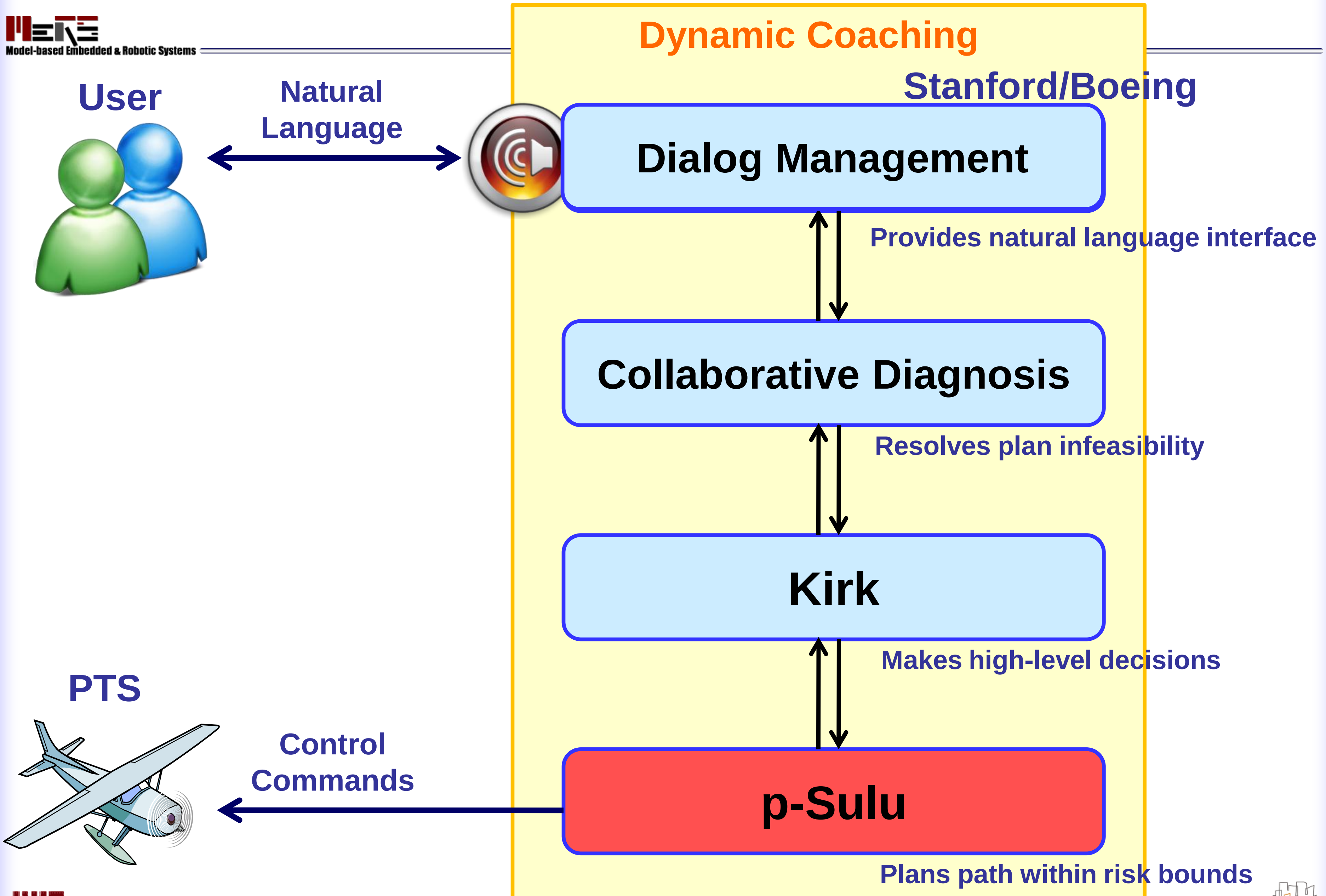
Limit

- Not efficient enough for real world problems (> 1000 episodes).



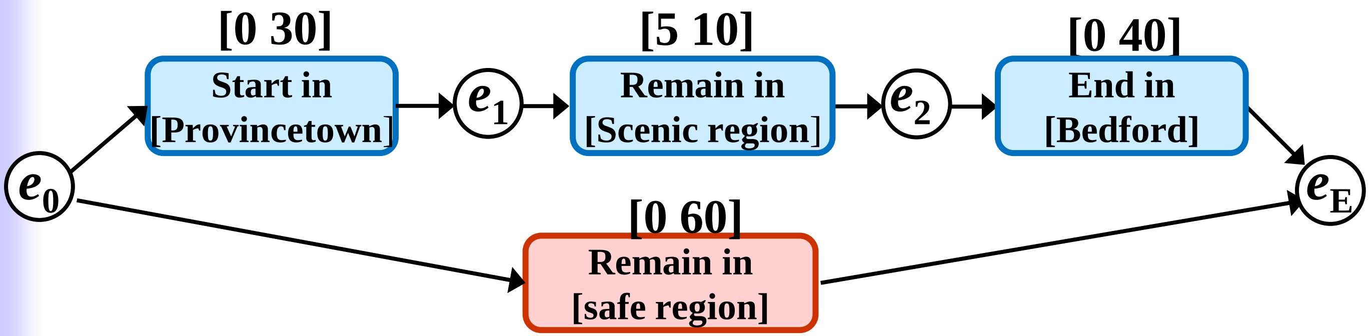
	Current Scenario		Future Scenario	
	# of Constraints	Computation Time	# of Constraints	Computation Time
Diagnosis Algorithm	15	0.1 sec	1000	> 1 day

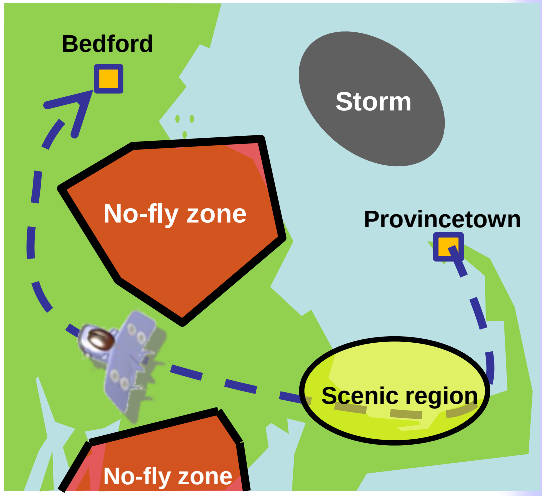
System Architecture



Sample PTS Scenario

Model-based Embedded & Robotic Systems





Chance constraints:

- c_1 : (blue box) $\cdots \quad \delta = 1\%$
- c_2 : (red box) $\cdots \quad \delta = 0.0001\%$

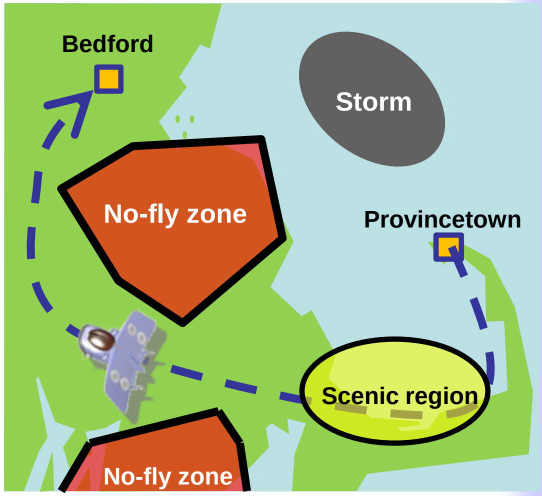
The passenger of the PAV wants to:

- go from Provincetown to Bedford within 60 minutes
- go through a scenic area and remain there between 5 and 10 minutes
- limit the risk of penetrating the NFZ or the storm to 0.001%

Three types of constraints

[0 30] Start in [Provincetown] → e_1 → [5 10] Remain in [Scenic region] → e_2 → [0 40] End in [Bedford] → e_E

e_0 → [0 60] Remain in [safe region] → e_E



Chance constraints: c_1 : [blue box] $\cdots \quad \delta = 1\%$

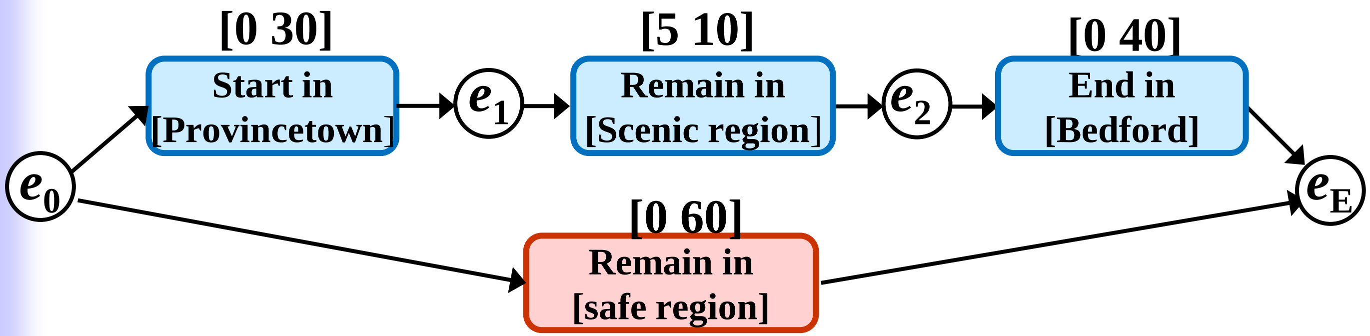
c_2 : [red box] $\cdots \quad \delta = 0.0001\%$

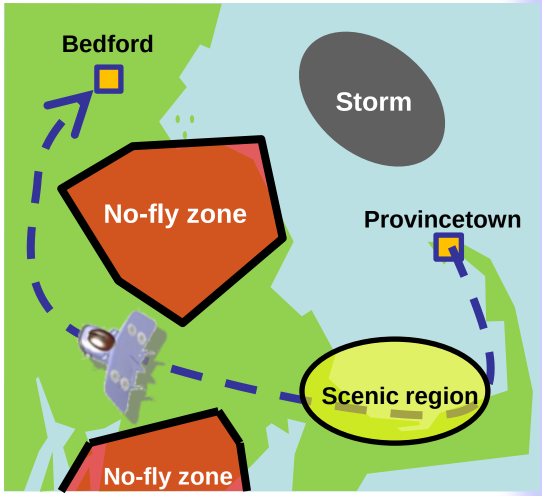
The passenger of the PAV wants to:

- go from **Provincetown** to **Bedford** within 60 minutes
- go through **a scenic area** and remain there between 5 and 10 minutes
- limit the risk of penetrating the NFZ or the storm to 0.001%

State constraints

Three types of constraints





Chance constraints:
c_1 : ⋯ $\delta = 1\%$
c_2 : ⋯ $\delta = 0.0001\%$

The passenger of the PAV wants to:

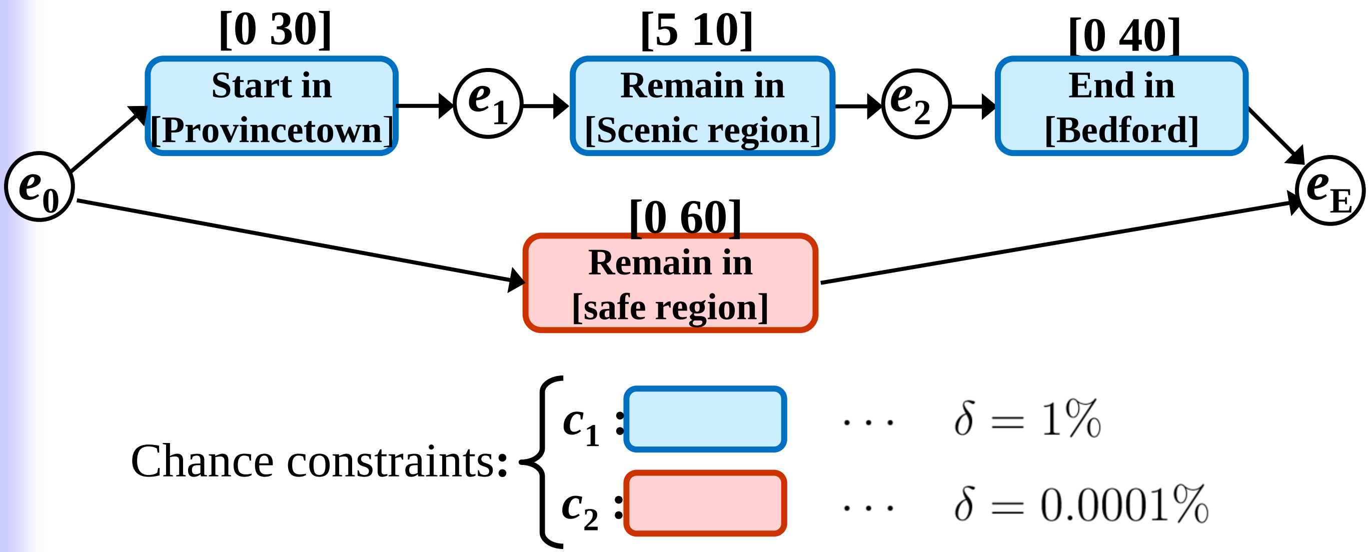
- go from **Provincetown** to **Bedford** **within 60 minutes**
- go through **a scenic area** and remain there between **5 and 10 minutes**
- limit the risk of penetrating the NFZ or the storm to 0.001%

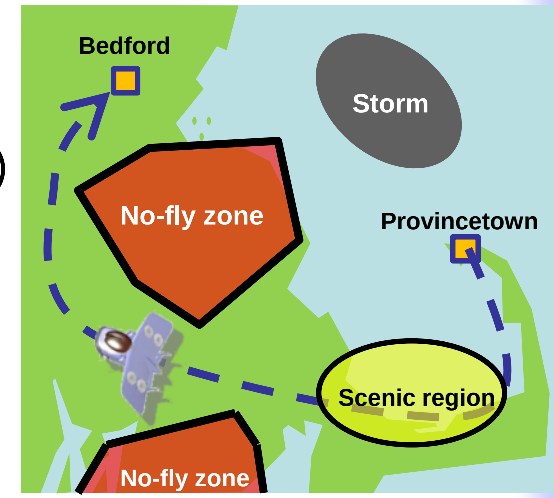
State constraints

Temporal constraints

Three types of constraints

Model-based Embedded & Robotic Systems





The passenger of the PAV wants to:

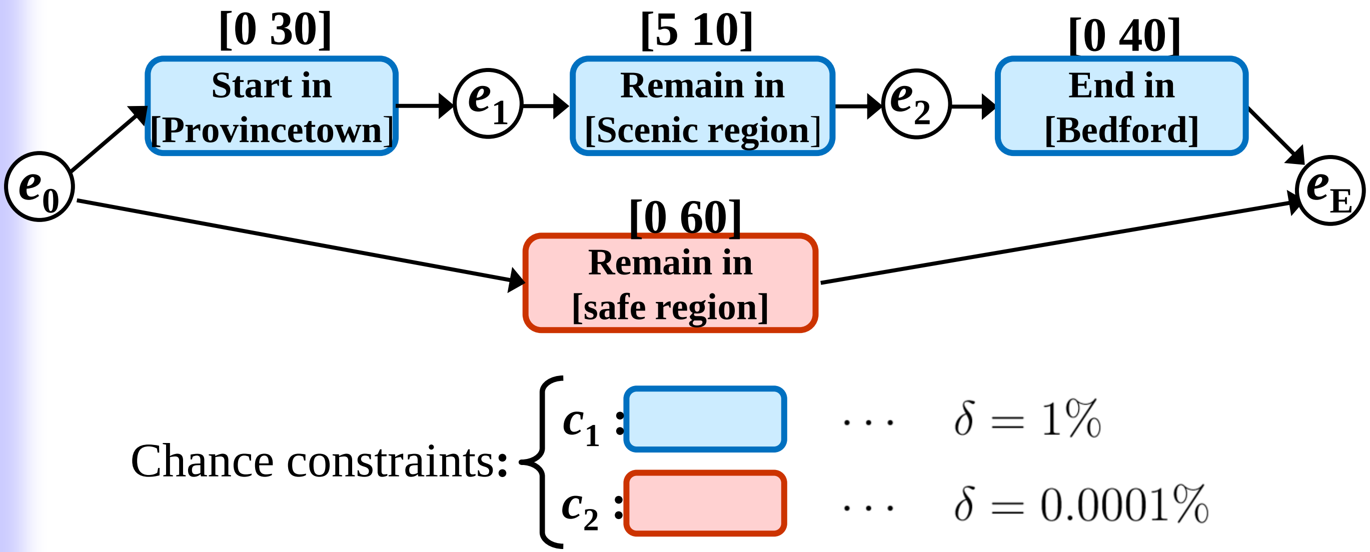
- go from **Provincetown** to **Bedford** **within 60 minutes**
- go through **a scenic area** and remain there between **5 and 10 minutes**
- limit the risk of penetrating the NFZ or the storm to **0.001%**

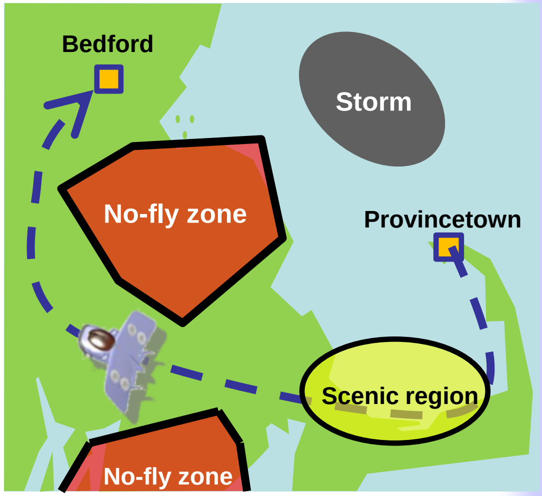
State constraints

Temporal constraints

Chance constraints

Three required capabilities





The passenger of the PAV wants to:

- go from **Provincetown** to **Bedford** **within 60 minutes**
- go through **a scenic area** and remain there between **5 and 10 minutes**
- limit the risk of penetrating the NFZ or the storm to **0.001%**

State constraints • **Goal-directed planning**

Temporal constraints • **Planning in continuous domain**

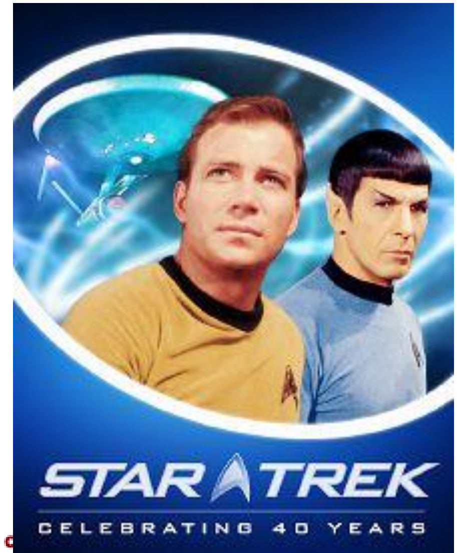
Chance constraints • **Risk-sensitive planning**

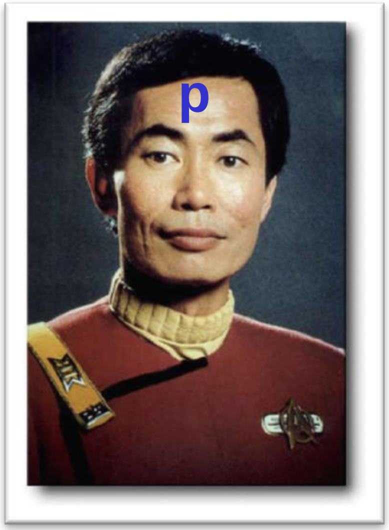
p-Sulu RH

VERY roughly speaking…

p-Sulu RH = probabilistic receding horizon Sulu

p-Sulu RH	**=**	**Iterative Risk Allocation**	**+**	**Receding horizon control**





Lieutenant Hikaru Sulu, from *Star Trek*

pSulu RH

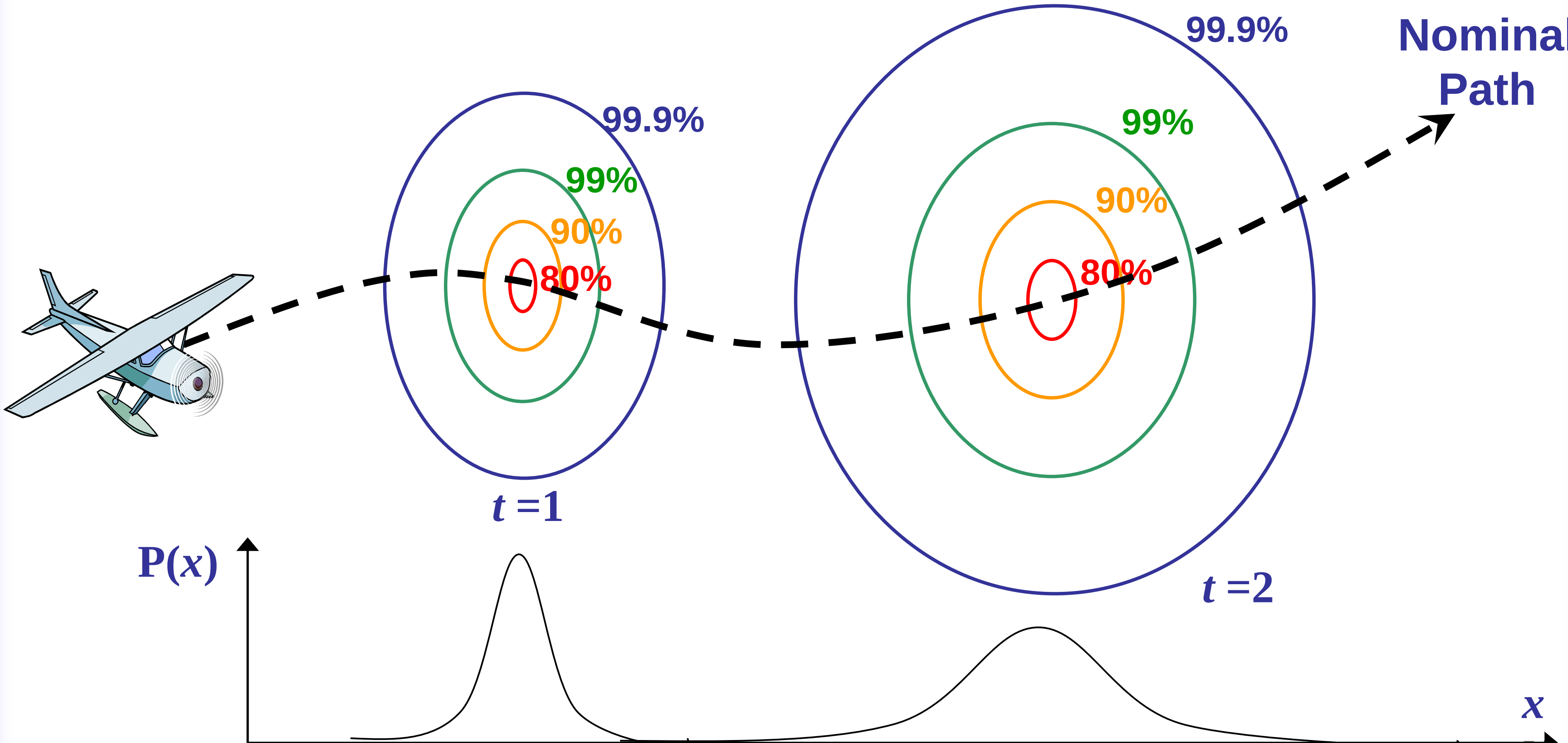
VERY roughly speaking…

p-Sulu RH = Iterative Risk Allocation + Receding horizon control

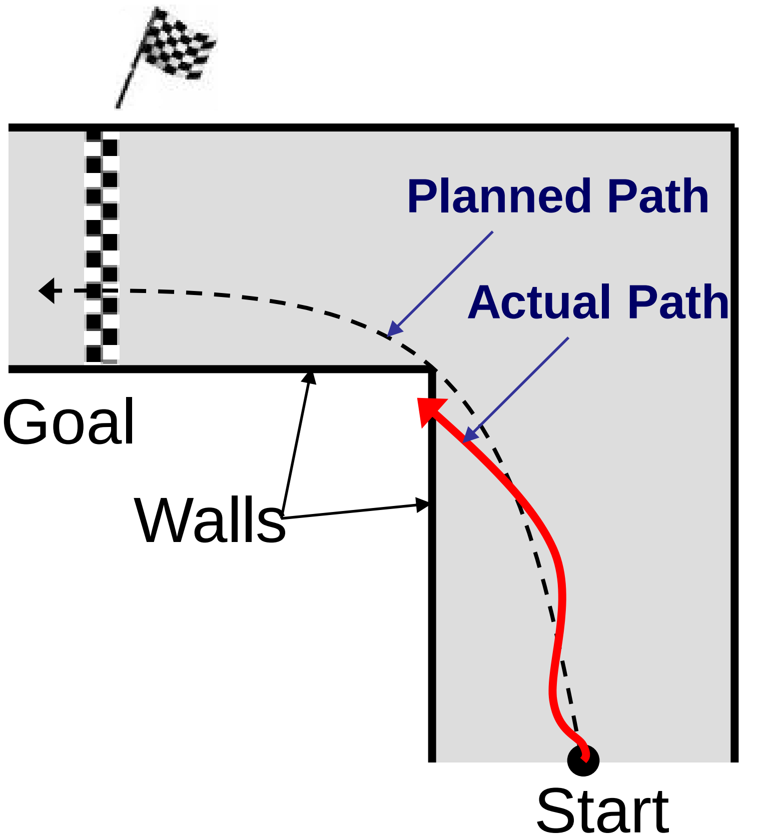
Optimal control Under Stochastic Uncertainty

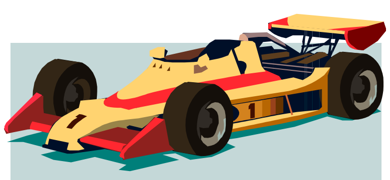
- **Exogenous disturbance**
- **State estimation error**

Risk of constraint violation



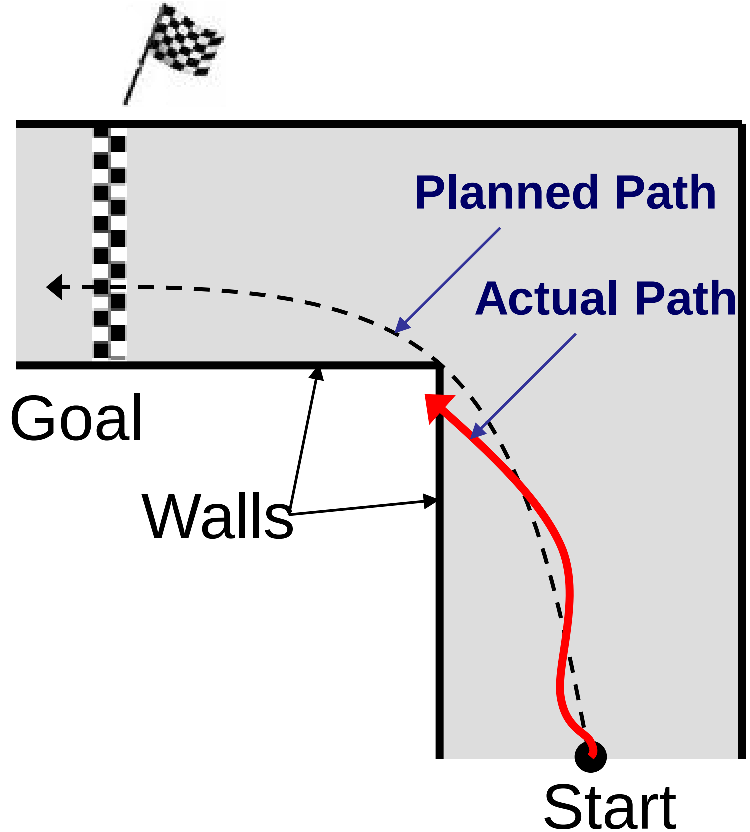
Example: Race Car Path Planning

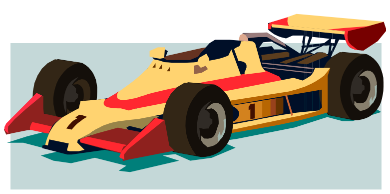




- A race car driver wants to go from the start to the goal as fast as possible
- Crashing into the wall may kill the driver
- Actual path may differ from the planned path due to uncertainty

Example: Race Car Path Planning





Problem

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

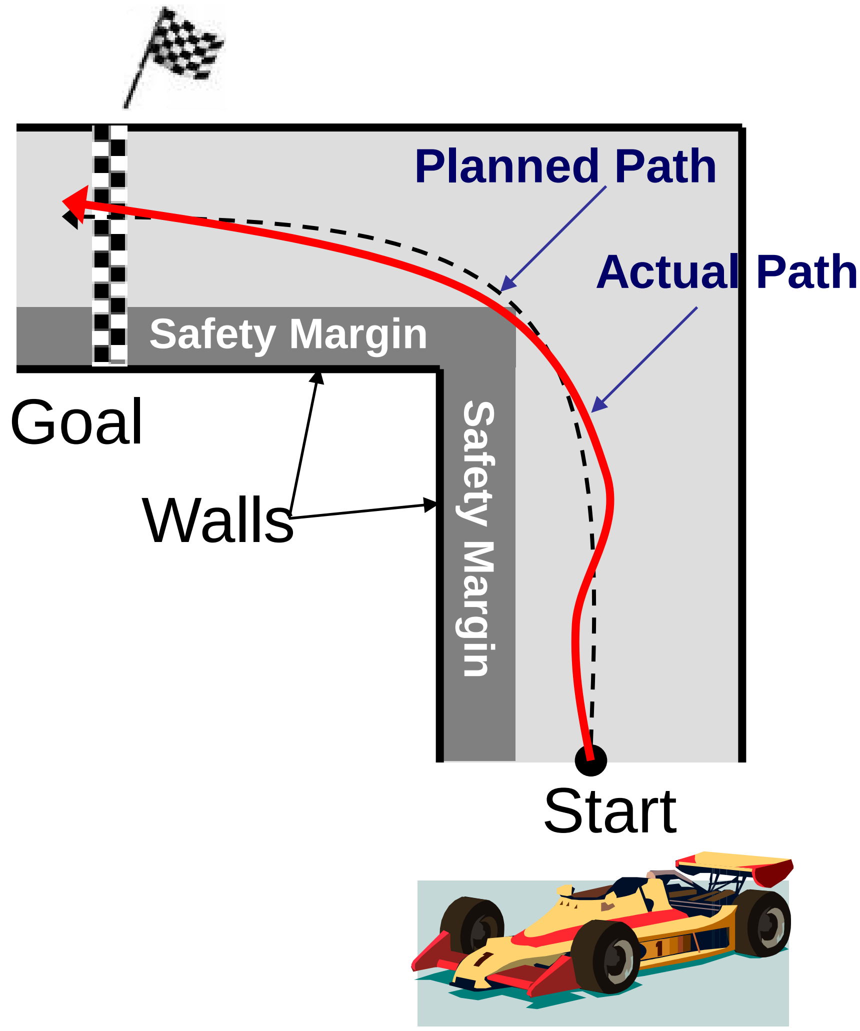
Risk bound

- Cannot guarantee 100% safety
- Driver wants a probabilistic guarantee:

  $$P(\text{crash}) < 0.1\%$$

  – **Chance constraint**

Example: Race Car Path Planning



Problem

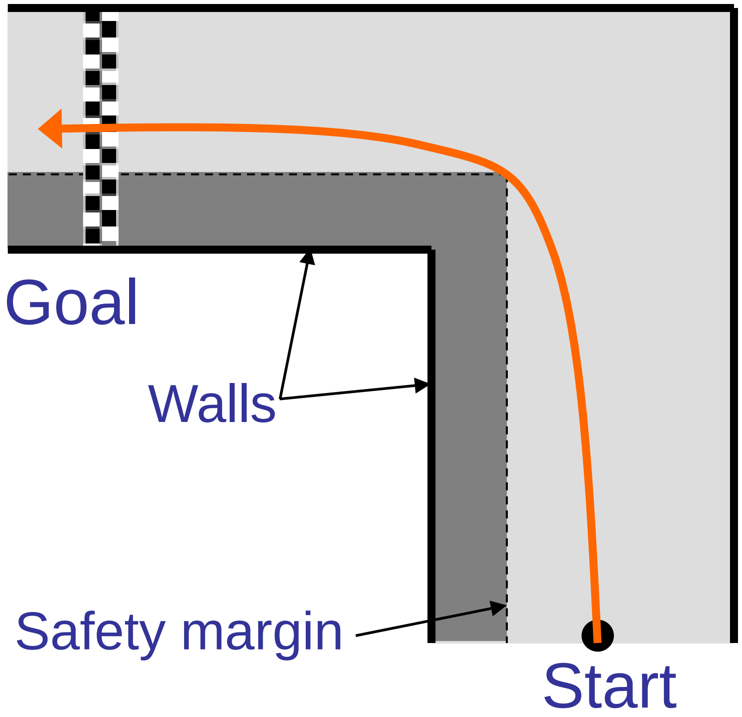
Find the fastest path to the goal, while limiting the probability of crash throughout the race *to 0.1%*

Risk bound

- Approach: set **safety margin** that guarantees the specified risk bound from start to the goal

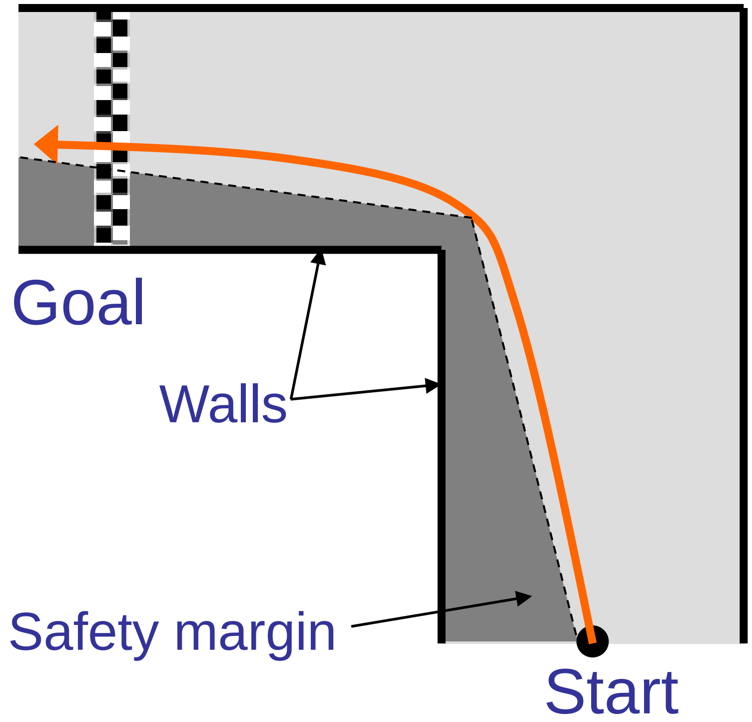
Optimization of Safety Margin

Uniform width



Longer path

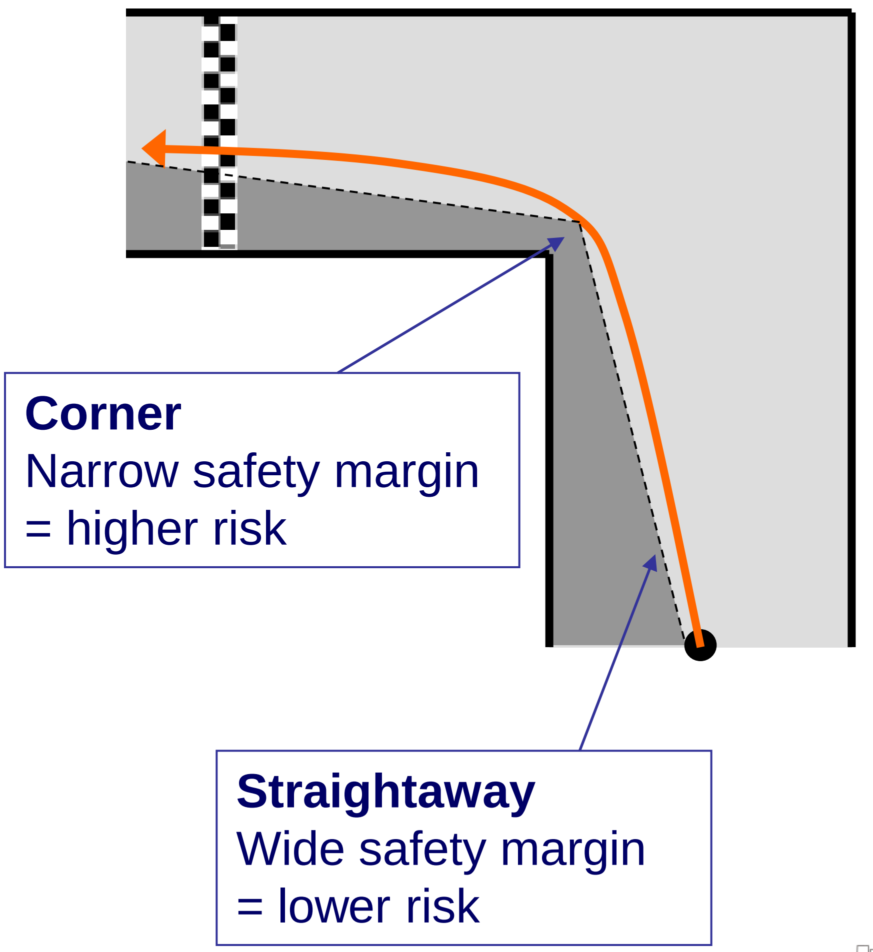
Non-uniform width



Shorter path

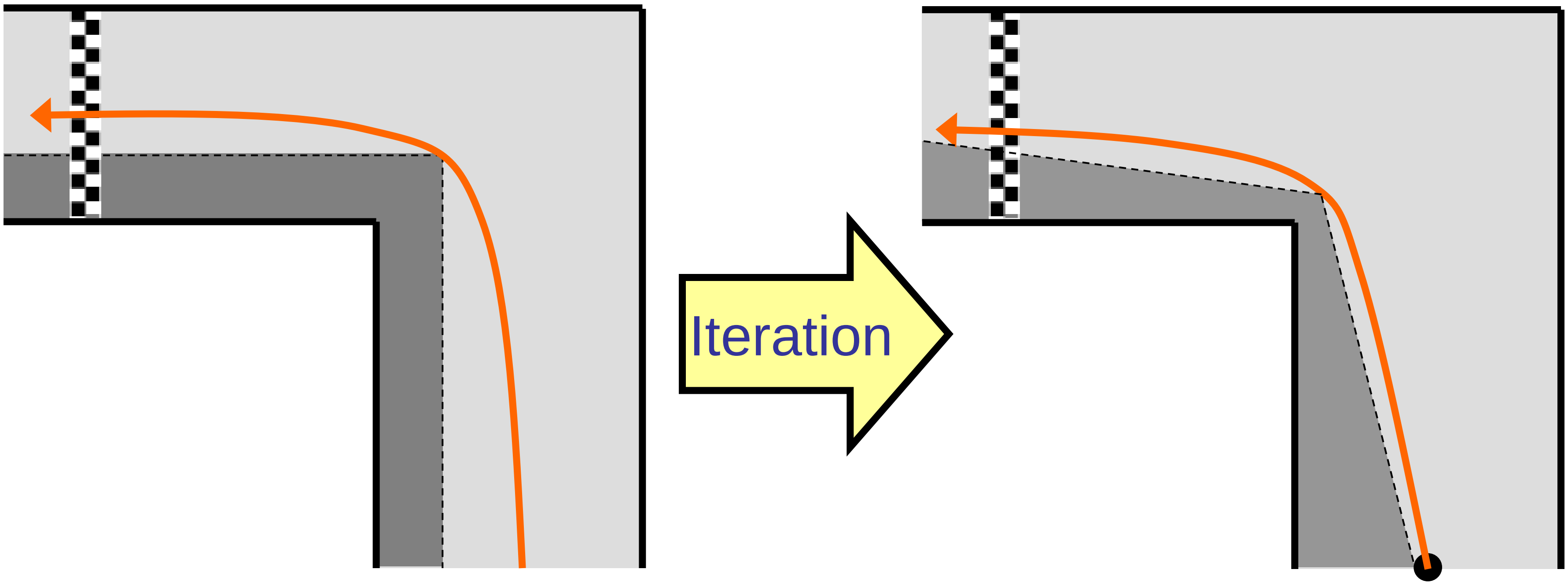
Key Idea - *Risk Allocation*

- Taking a risk at the corner results in a shorter path than taking the same amount of risk at the straightaway
- **Sensitivity** of path length to risk **is higher at the corner**
- ***Risk Allocation***
  - Need to optimize the allocation of risk to time steps and constraints

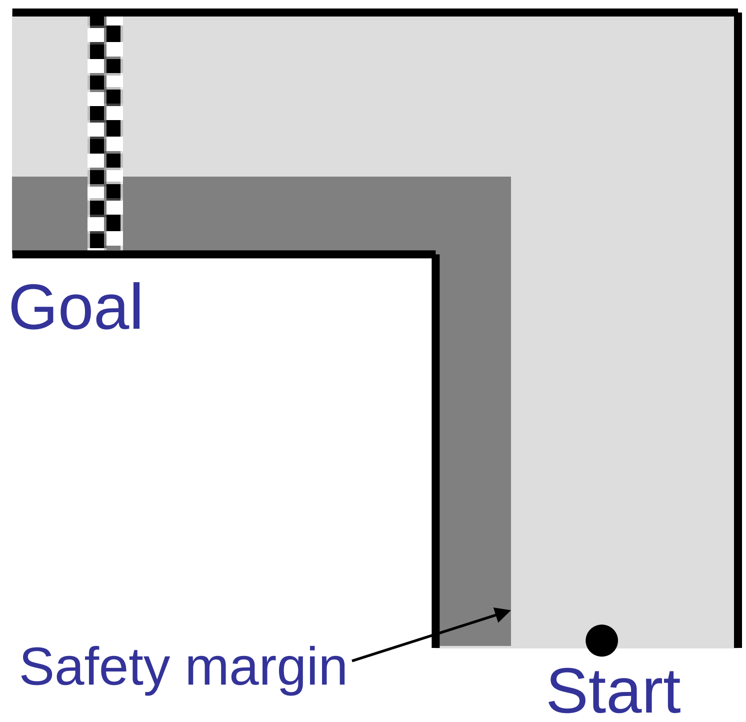


Iterative Risk Allocation (IRA) Algorithm

- Starts from a suboptimal risk allocation
- Improves the risk allocation by iterations



Iterative Risk Allocation Algorithm



Algorithm IRA

1 Initialize with arbitrary risk allocation

2 Loop

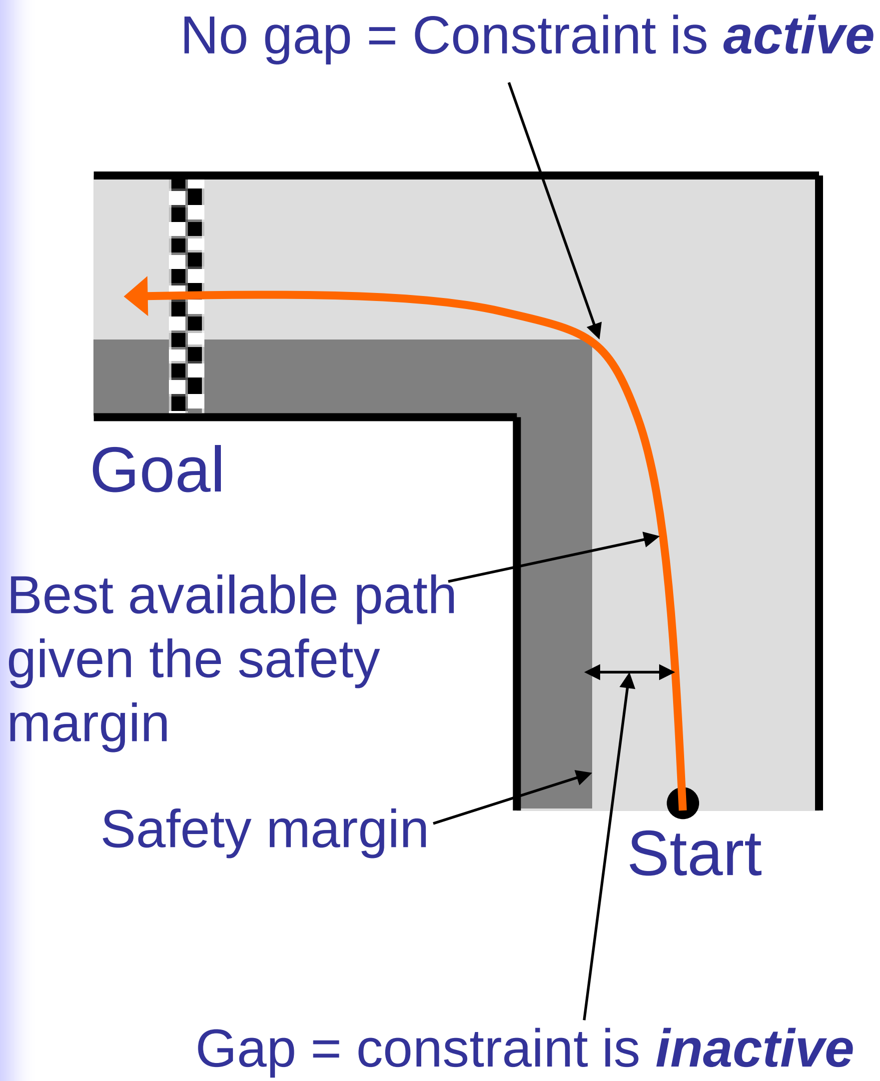
3 Compute the best available path given the current risk allocation

4 Decrease the risk where the constraint is inactive

5 Increase the risk where the constraint is active

6 End loop

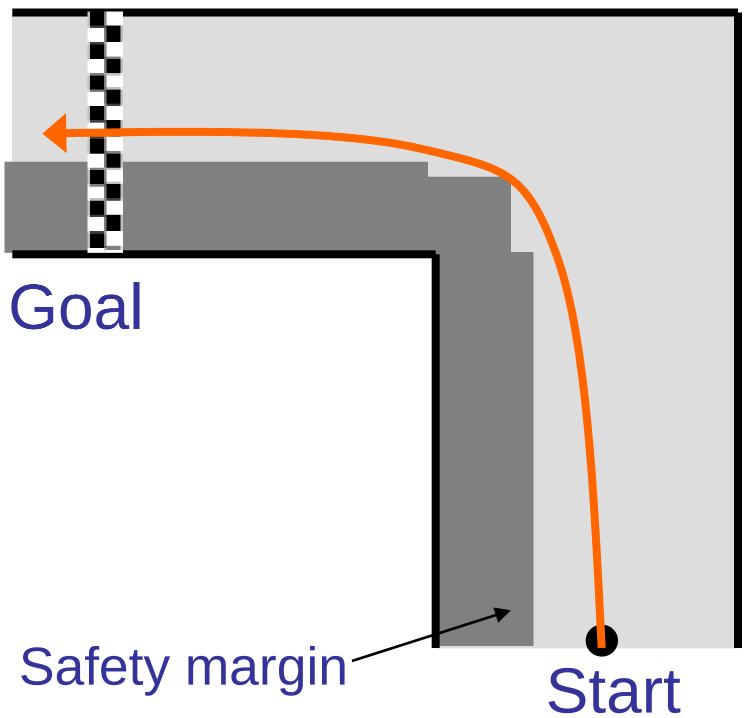
Iterative Risk Allocation Algorithm



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



Algorithm IRA

1 Initialize with arbitrary risk allocation

2 Loop

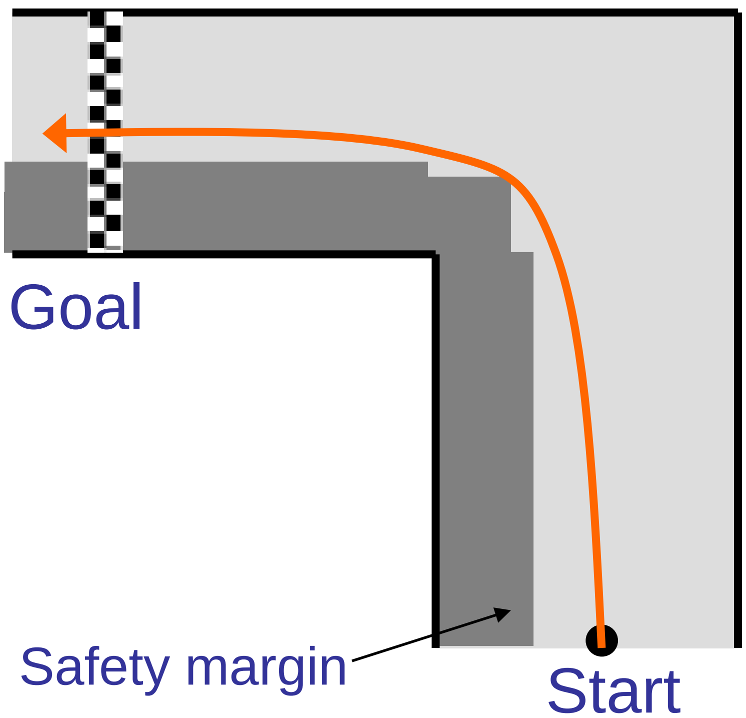
3 Compute the best available path given the current risk allocation

4 Decrease the risk where the constraint is inactive

5 Increase the risk where the constraint is active

6 End loop

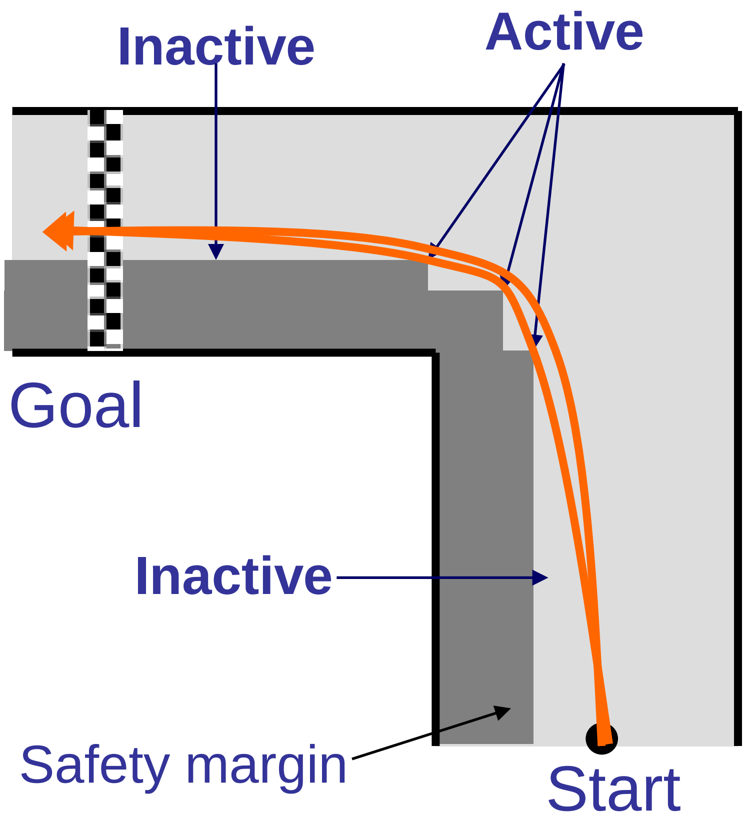
Iterative Risk Allocation Algorithm



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



Algorithm IRA

1 Initialize with arbitrary risk allocation

2 Loop

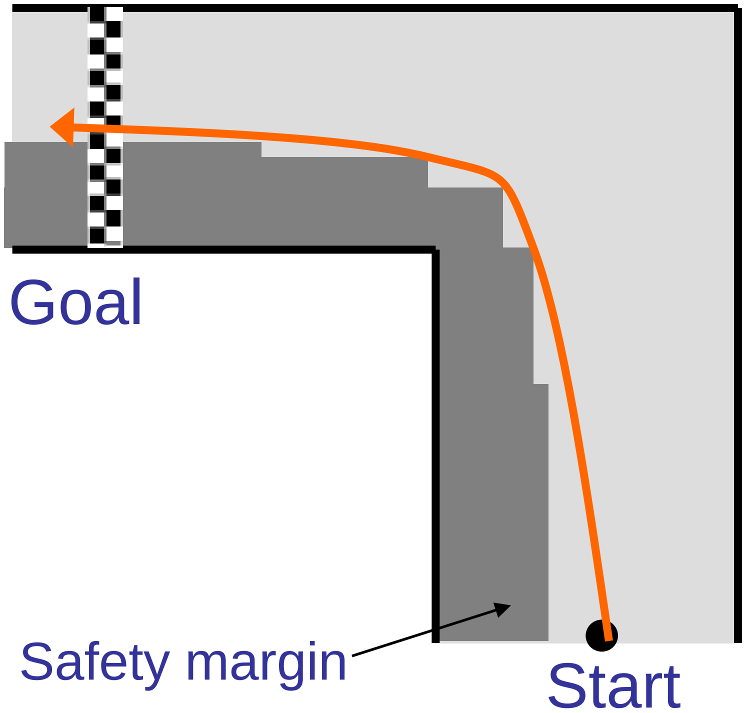
<u>**3**</u> Compute the best available path given the current risk allocation

4 Decrease the risk where the constraint is inactive

5 Increase the risk where the constraint is active

6 End loop

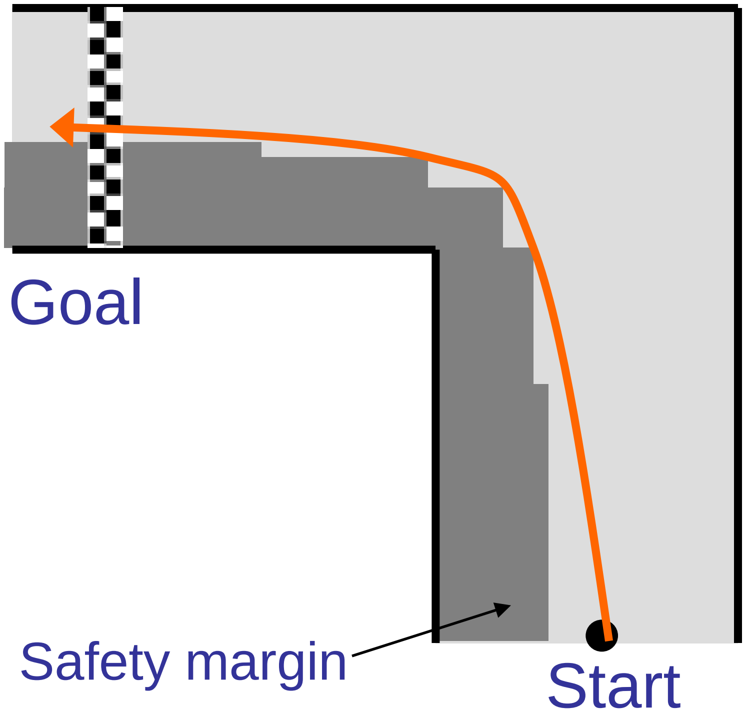
Iterative Risk Allocation Algorithm



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



Algorithm IRA

1 Initialize with arbitrary risk allocation

2 Loop

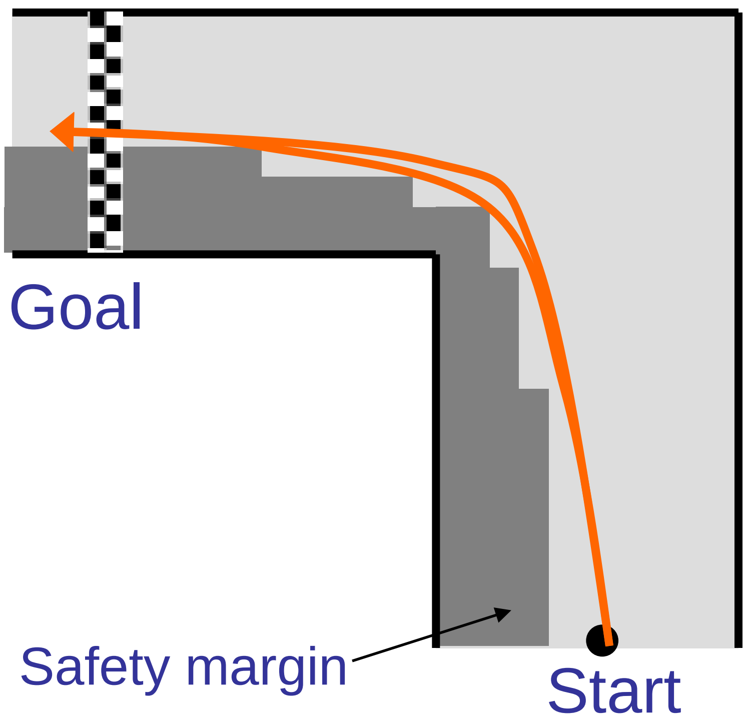
3 Compute the best available path given the current risk allocation

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

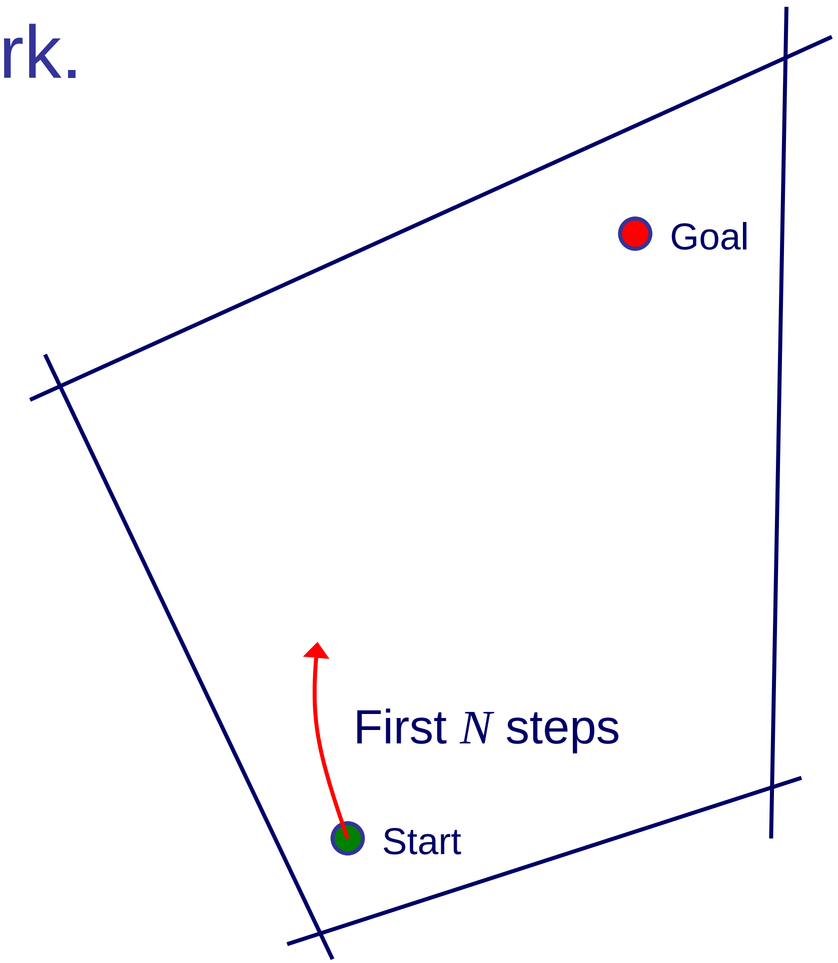
pSulu RH

VERY roughly speaking…

pSulu RH **=** **Iterative Risk Allocation** **+** **Receding horizon control**

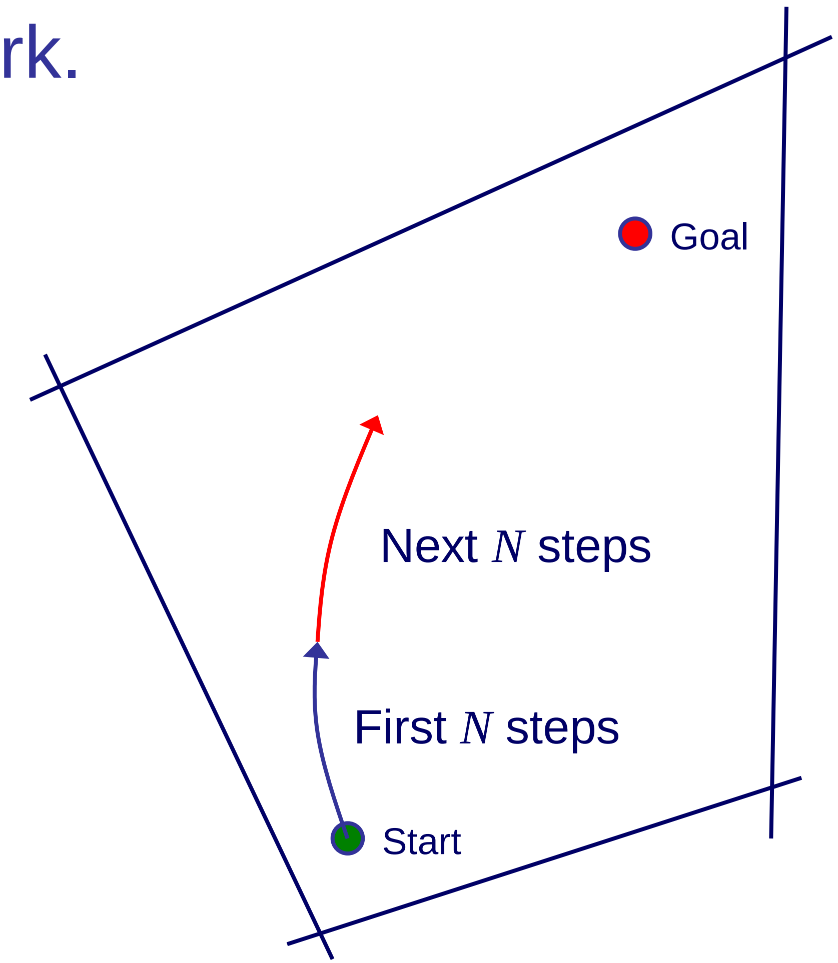
Receding Horizon Control

- Patchwork.



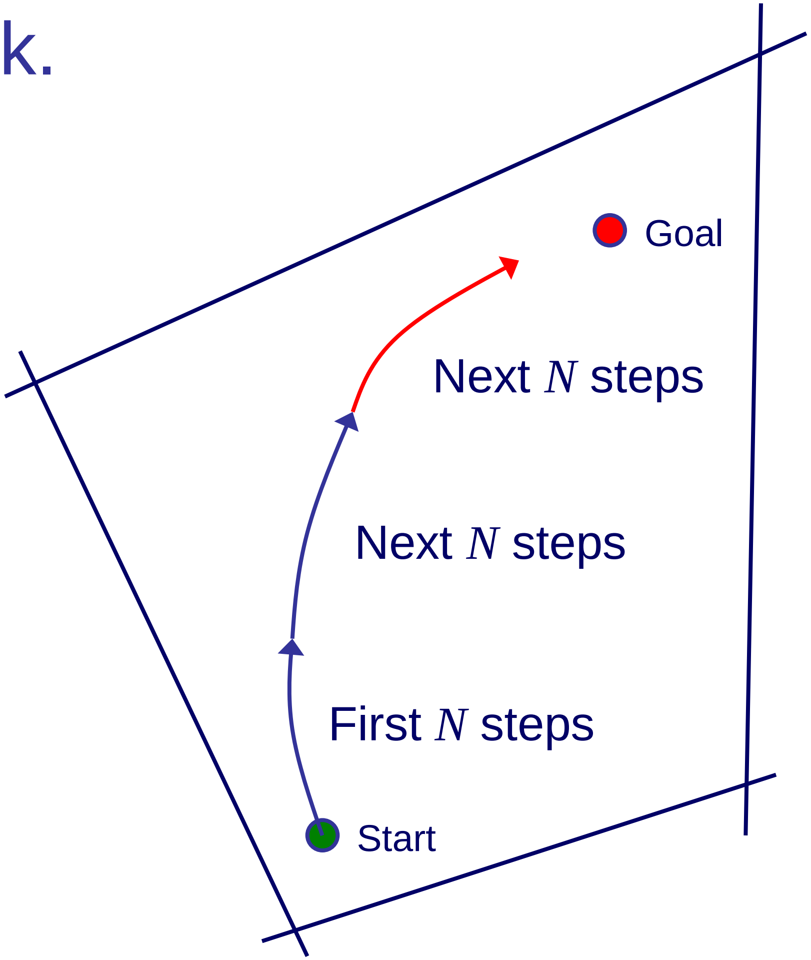
Receding Horizon Control

- Patchwork.



Receding Horizon Control

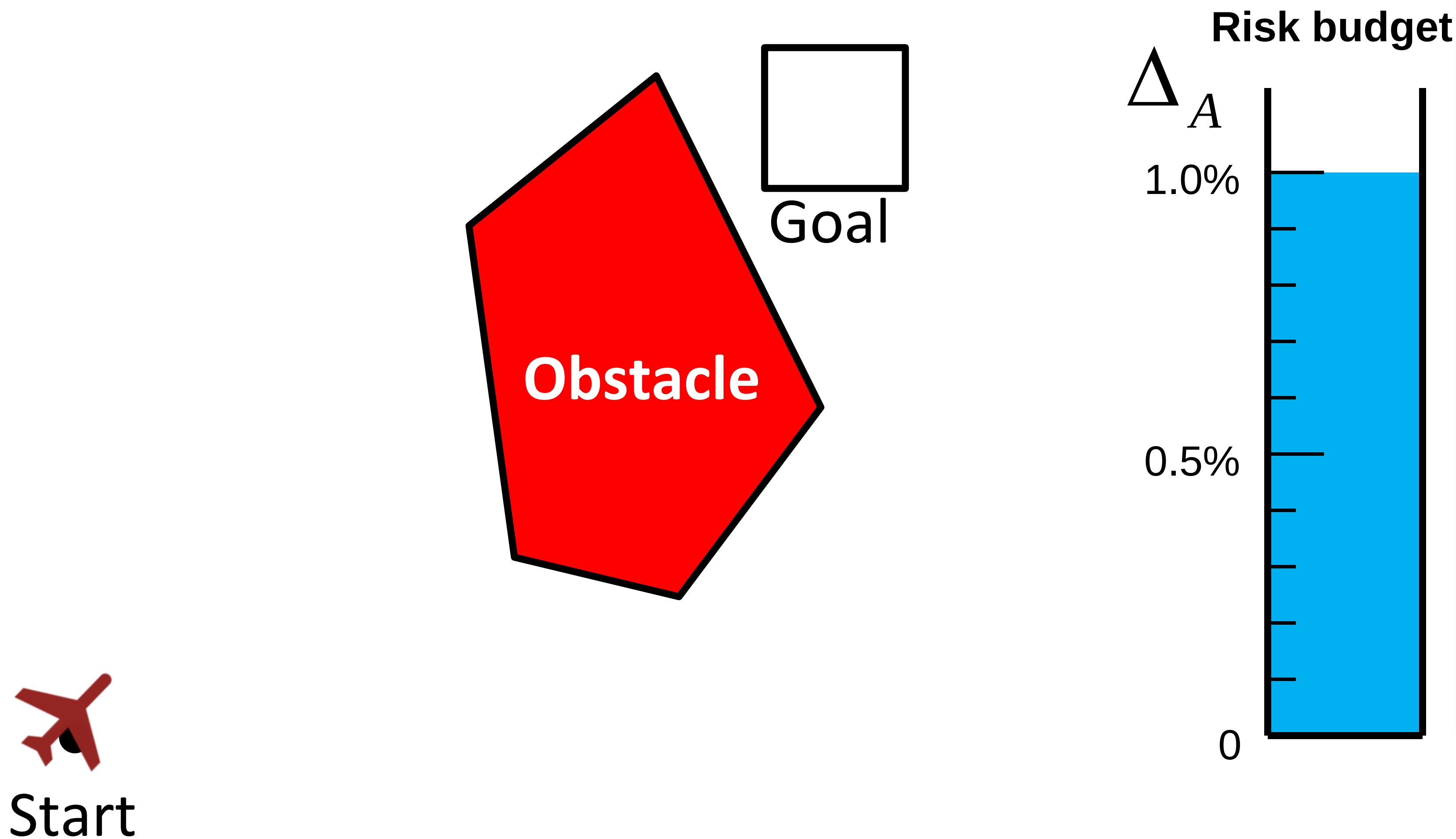
- Patchwork.



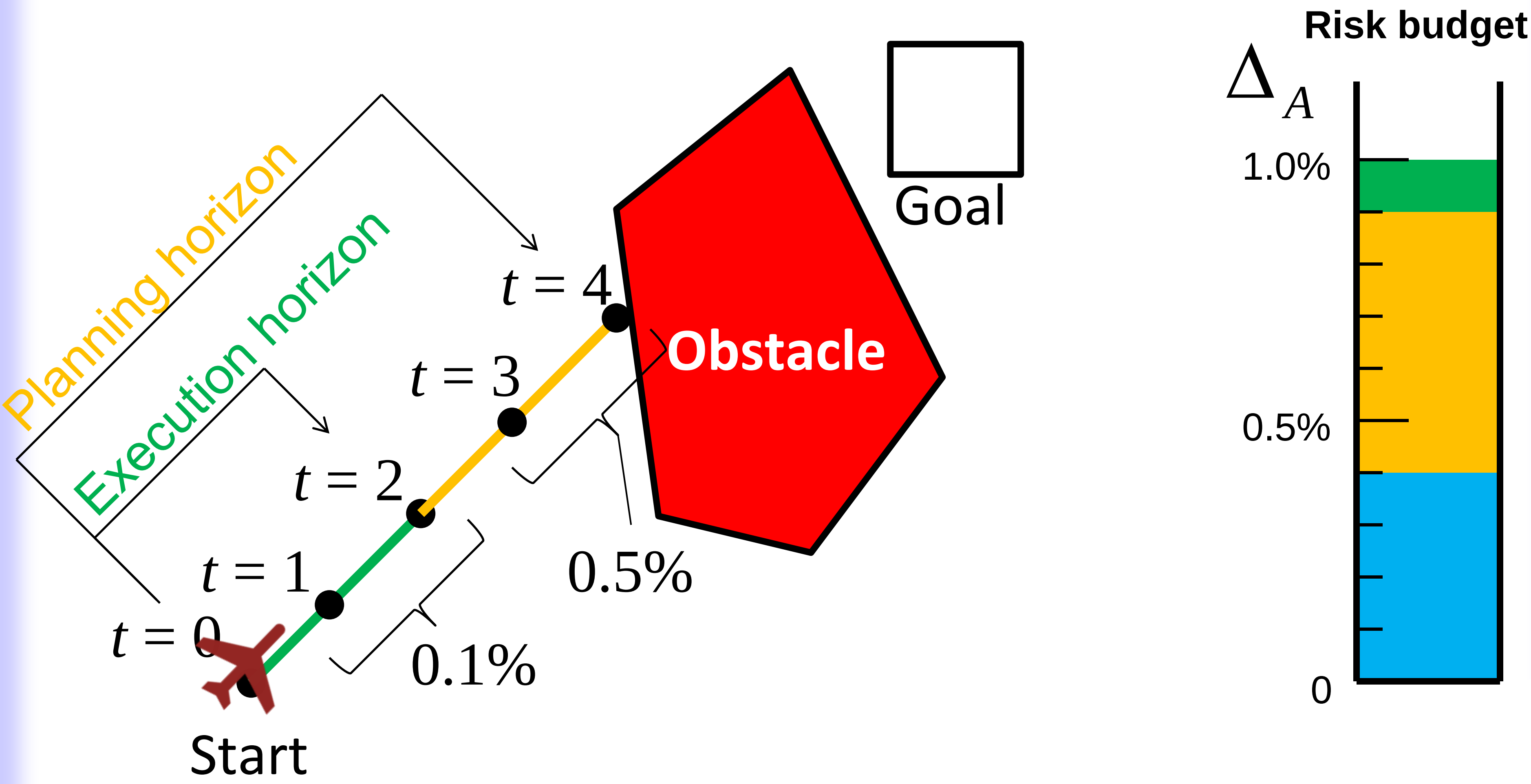
Risk Budgeting

Model-based Embedded & Robotic Systems

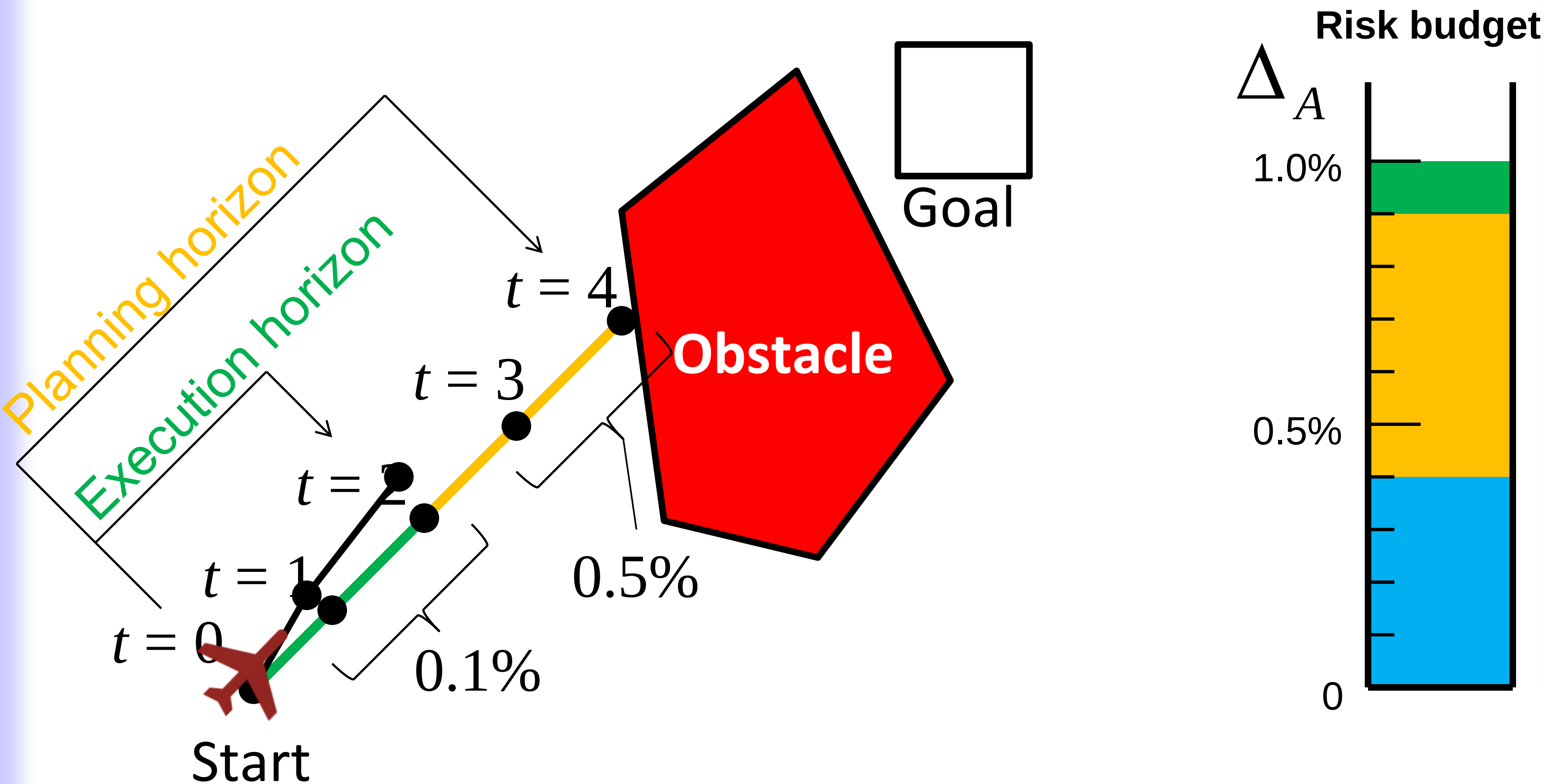
$\Delta = 1\%$



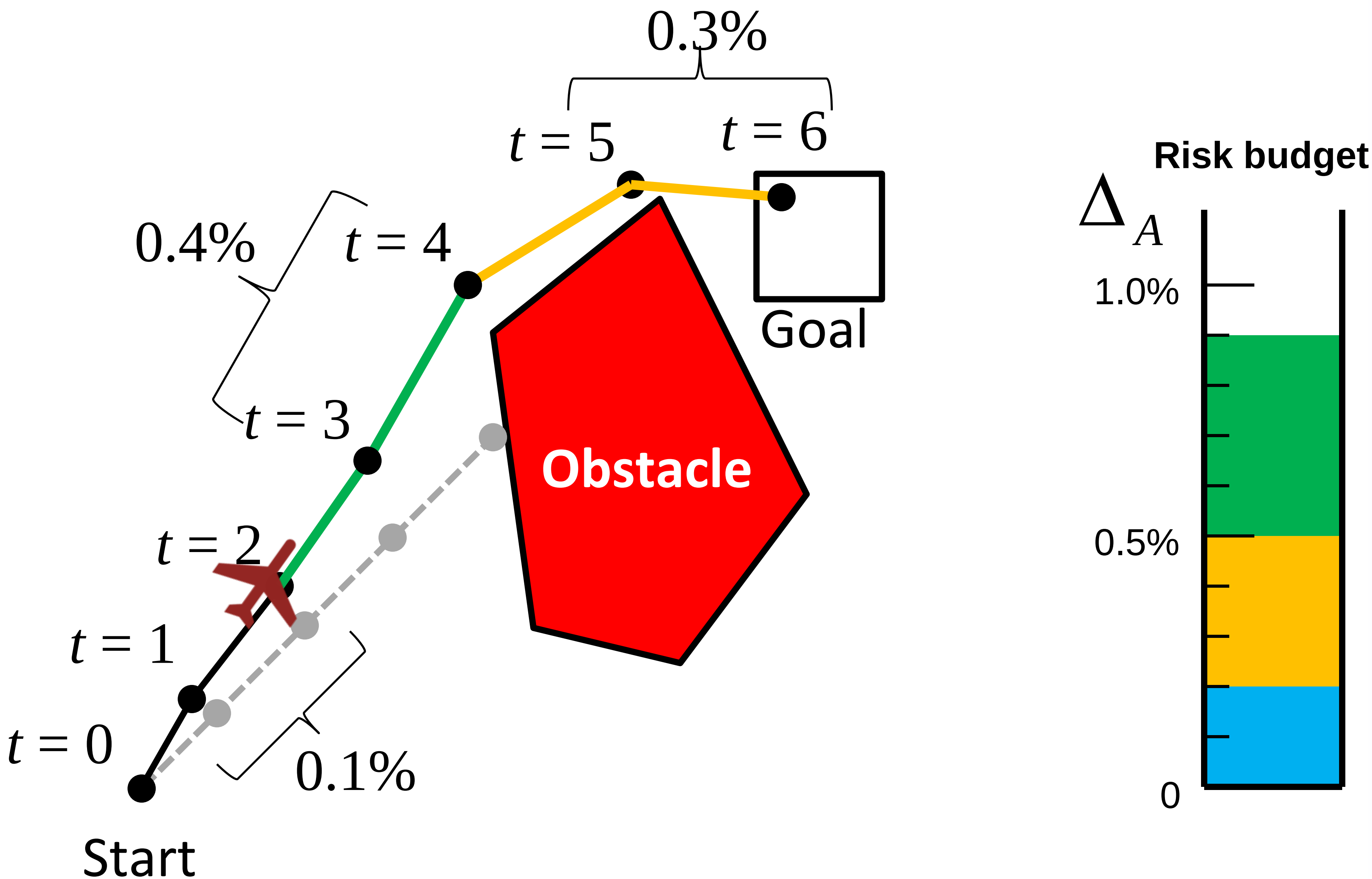
Risk Budgeting



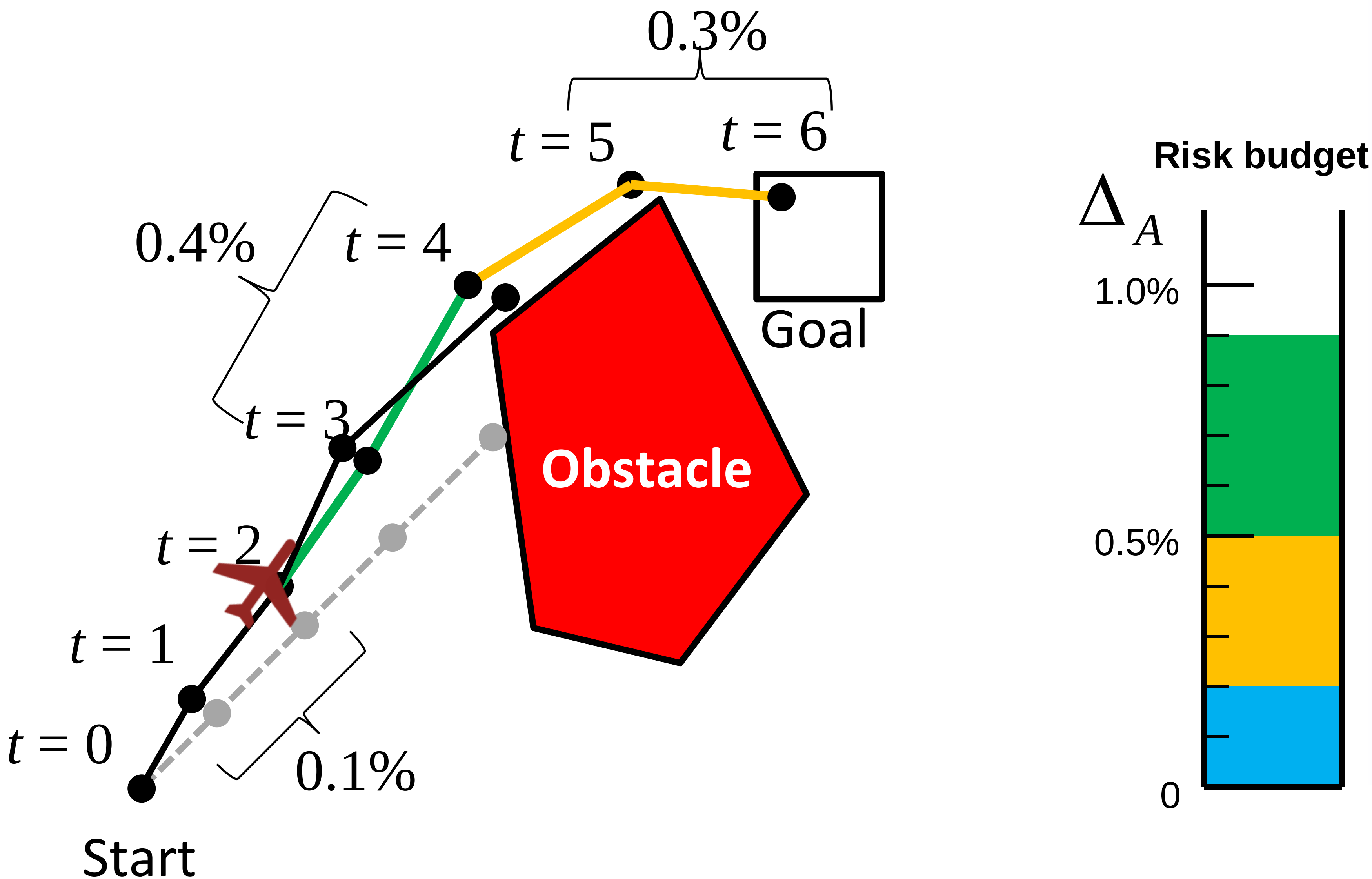
Risk Budgeting



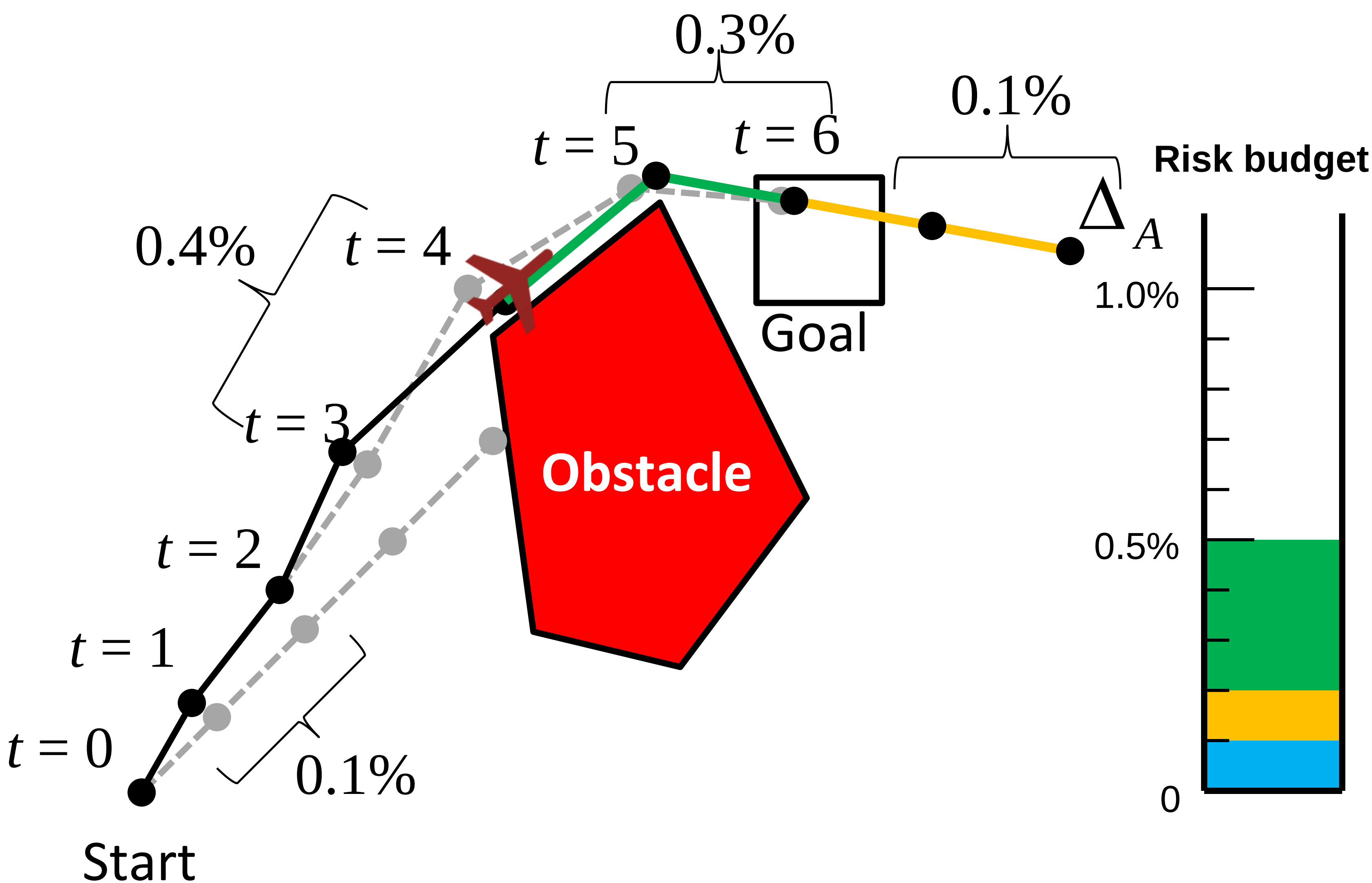
Risk Budgeting



Risk Budgeting

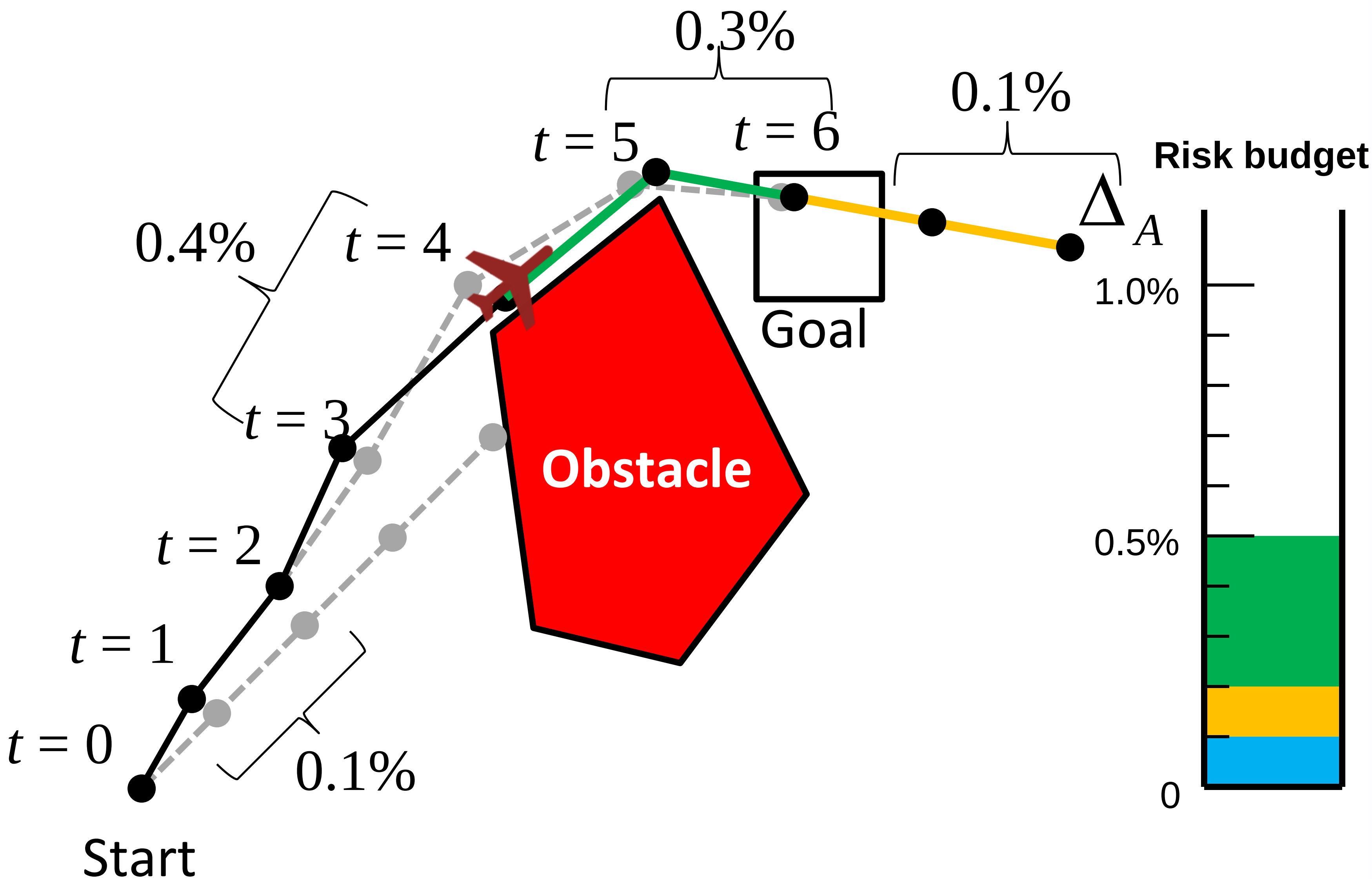


Risk Budgeting



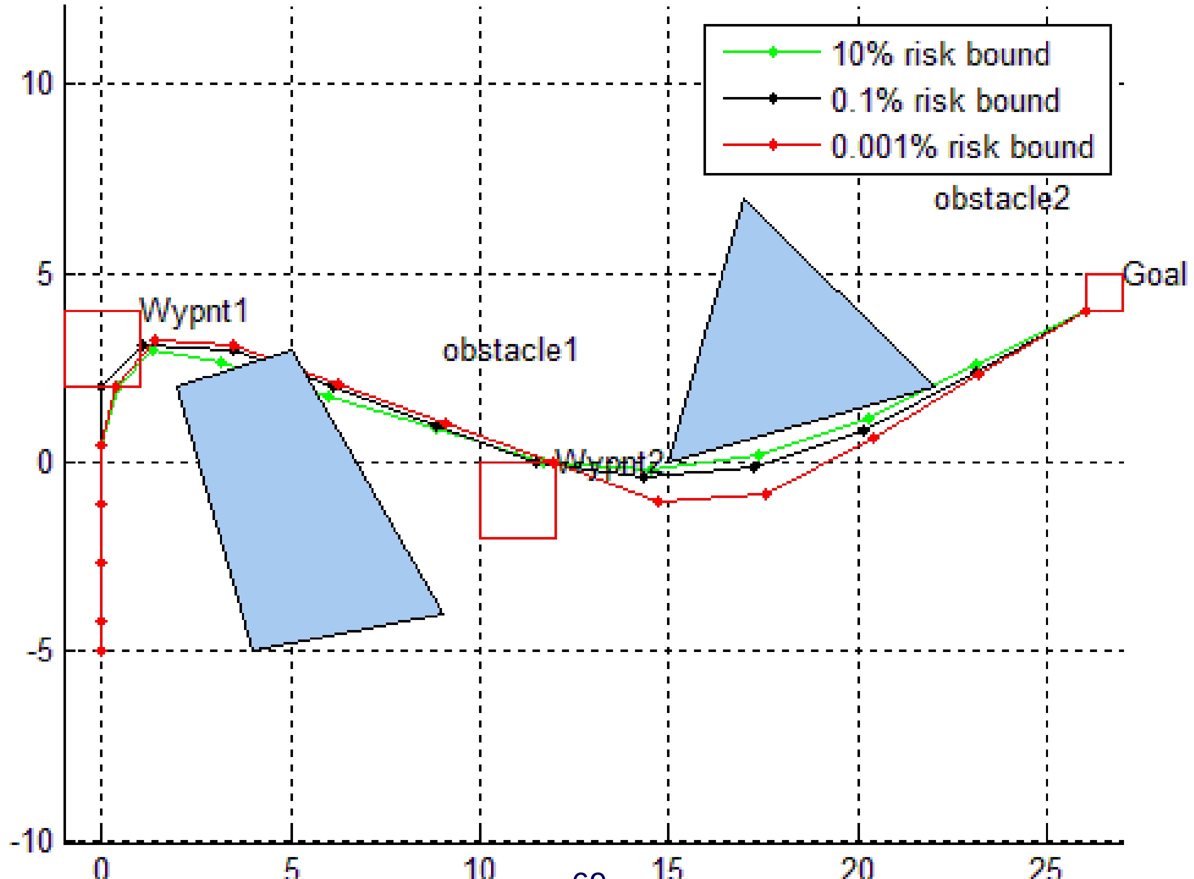
Risk Budgeting

Model-based Embedded & Robotic Systems



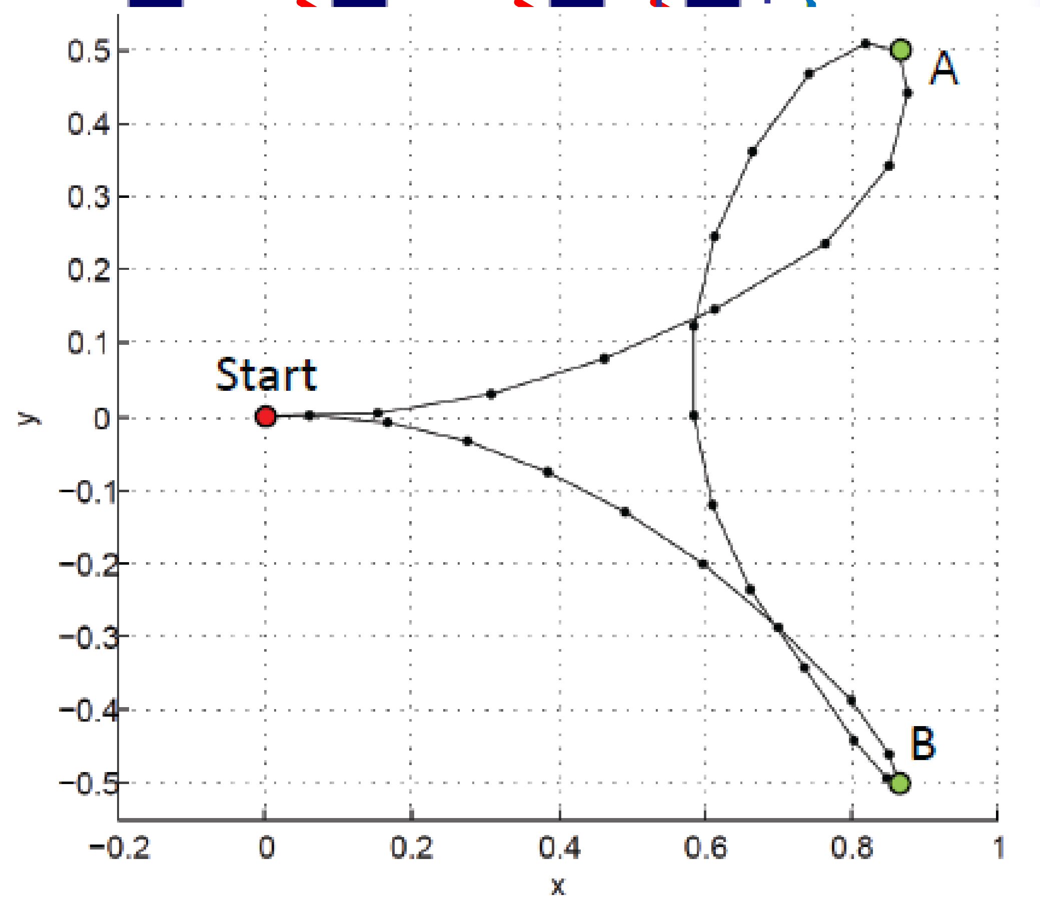
Result: p-Sulu

- Risk-performance trade-off
  - More risk ⇔ shorter path
  - Less risk ⇔ longer path



p-Sulu Application to Space Rendezvous

HTV unmanned resupply vehicle



Challenges:

- Risk of collision
- Complicated rendezvous procedure
- Unintuitive dynamics (follows Clohessy-Wiltshire eq.)

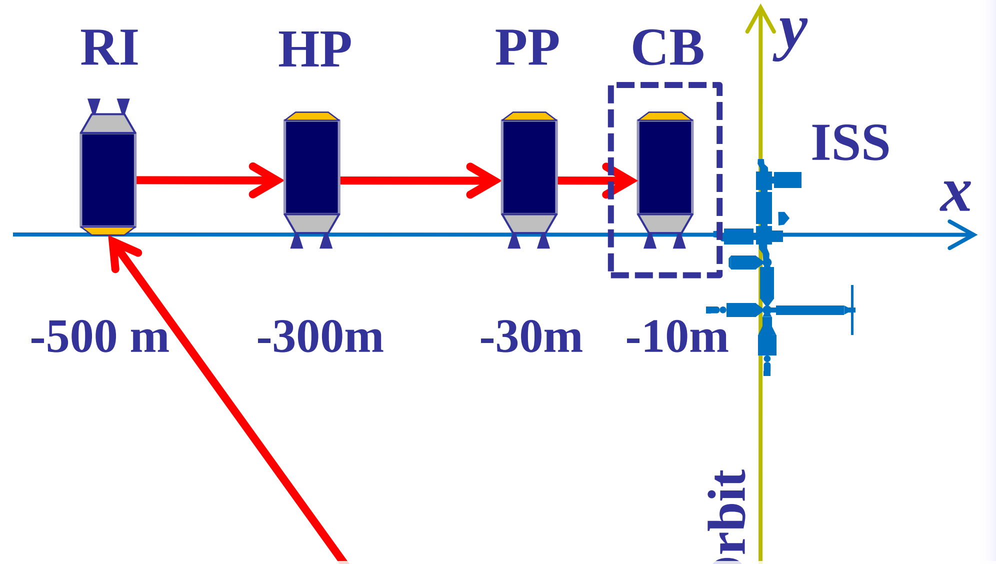
CB: Capture Box

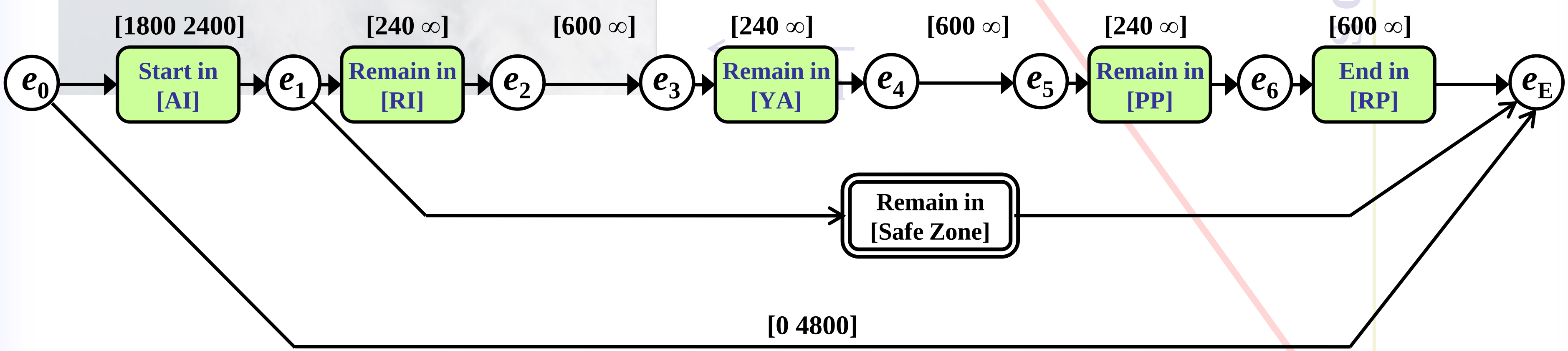
-5000 m

HTV rendezvous planning problem

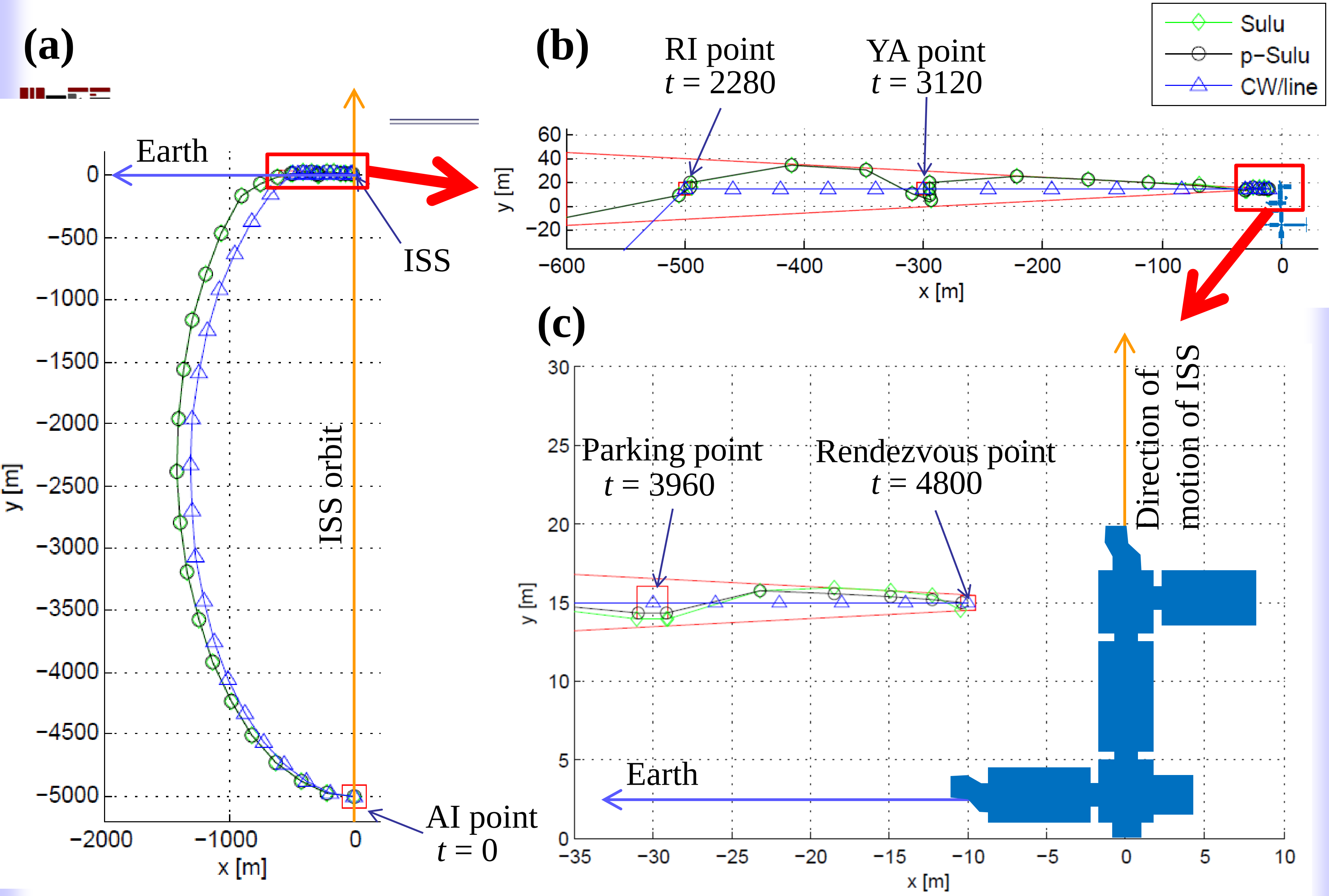
Model-based Embedded & Robotic Systems

HTV unmanned resupply vehicle





Chance constraints: (green box) $\cdots \Delta_1 = 0.5\%$; (double-bordered box) $\cdots \Delta_2 = 0.5\%$



AI: Approach Initiation, RI: R-bar Initiation, YA: Yaw-around

HTV rendezvous planning : Result

Algorithm		Sulu	**p-Sulu**	Nominal
c_1 (Navigation)	Risk bound Δ_1	0.005		
	Probability of failure $P_{fail,1}$	0.92	0.0024	$< 10^{-6}$
c_2 (Goals)	Risk bound Δ_2	0.005		
	Probability of failure $P_{fail,2}$	1.0	0.0029	$< 10^{-6}$
Cost function value (Delta V) $J^{\star}$ (m/s)		7.30	7.32	8.73
Computation time (s)		3.9	11.4	0.09

11.9 kg saving of fuel, compared to the nominal plan