Joint Chance-Constrained Dynamic Programming

Masahiro Ono

Keio University

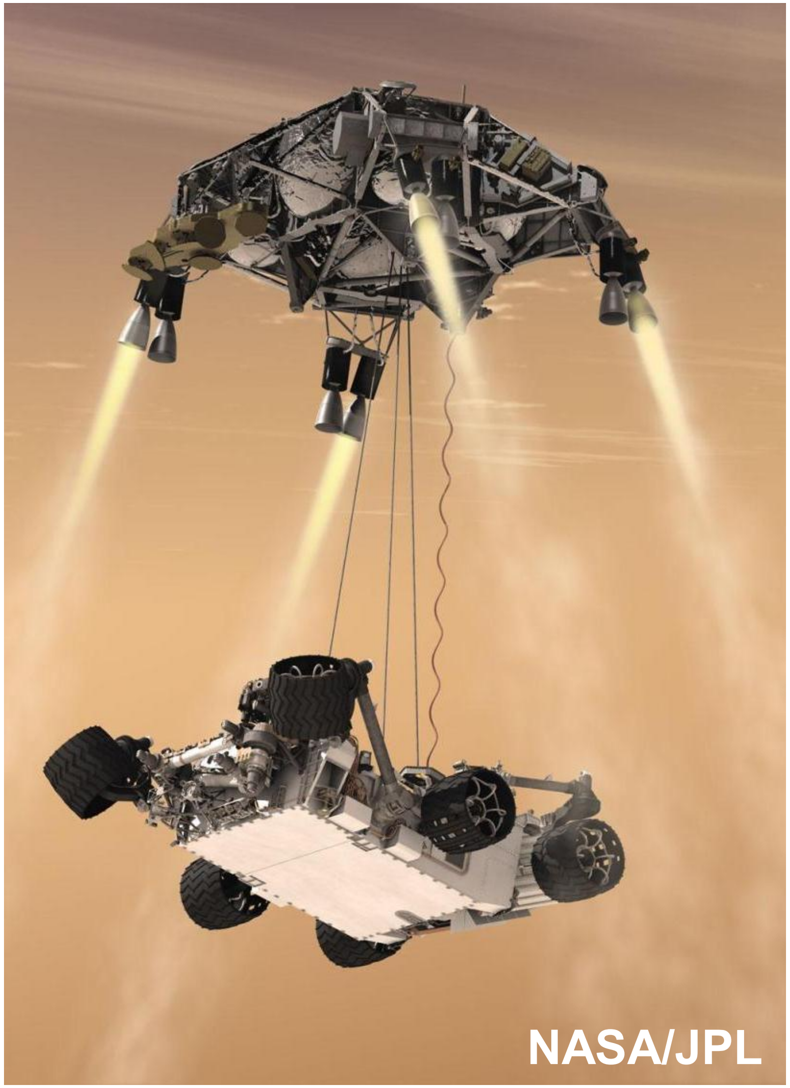
CALAMVS GLADIO FORTIOR 1858

Yoshiaki Kuwata

J. (Bob) Balaram

JPL

Motivation: Planetary Mission Planning



Future Mars entry, descent & landing requires to:

- Optimize lander's control law/landing site
- Manage risk of landing failure
- Minimize rover's driving distance to science targets

Intro: Chance-Constrained MPC

$$\min_{u_{\tau:\tau+N-1}} \mathbb{E}\left\{ g_N(x_N) + \sum_{k=\tau}^{\tau+N-1} g_k(x_k, u_k) \right\}$$

$$\text{s.t.} \qquad \Pr\left\{ \bigwedge_{k=\tau}^{\tau+N-1} h(x_k) \preceq 0 \right\} \geq 1 - \Delta$$

Risk bound

$$\bigwedge_{k=\tau}^{\tau+N-1} x_{k+1} = f(x_k, u_k, w_k)$$

- Significant body of work in Chance-constrained MPC
- Applications
  - Building control
  - Power grid control
  - Path planning

Chance-constrained *Dynamic Programming*

$$\min_{\boldsymbol{\mu}} \mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k)) \right\}$$

$$\text{s.t.} \qquad \Pr\left\{ \bigwedge_{k=1}^{N} x_k \in \mathcal{X}_k \;\middle|\; x_0 \right\} \geq 1 - \Delta$$

Risk bound

$$\bigwedge_{k=0}^{N-1} x_{k+1} = f(x_k, u_k, w_k)$$

DP can deal with

- nonlinear dynamics
- nonconvex feasible reagion
- discrete state space
- non-Gaussian probability distribution
- state-dependent disturbance

State constraint: $x_k \in \mathcal{X}_k$

Control constraint: $u_k \in \mathcal{U}_k(x_k)$

Control policy: $\mu_k : \mathcal{X} \mapsto \mathcal{U}(x_k)$

Chance-constrained *Dynamic Programming*

$$\min_{\boxed{\mu}} \mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k)) \right\}$$

$$\text{s.t.} \qquad \Pr\left\{ \bigwedge_{k=1}^{N} x_k \in \mathcal{X}_k \,\middle|\, x_0 \right\} \geq 1 - \Delta$$

Risk bound

$$\bigwedge_{k=0}^{N-1} x_{k+1} = f(x_k, u_k, w_k)$$

DP optimizes control policy (i.e., feedback control law)

State constraint: $x_k \in \mathcal{X}_k$

Control constraint: $u_k \in \mathcal{U}_k(x_k)$

Control policy: $\mu_k : \mathcal{X} \mapsto \mathcal{U}(x_k)$

CCDP: Challenges and Approaches

$$\min_{\boldsymbol{\mu}} \mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k)) \right\}$$

$$\text{s.t.} \quad \mathbb{E}\left\{ c_N(x_N) + \sum_{k=0}^{N-1} c_k(x_k, \mu_k(x_k)) \right\} \leq \Delta$$

Challenge 1: Standard DP cannot solve a constrained optimization

Constrained DP/MDP (Rossman 1977, Beulter & Ross 1985, Williams et al. 2008) can be applied only to constraints in the same form as the objective function

State constraint: $x_k \in \mathcal{X}_k$

Control constraint: $u_k \in \mathcal{U}_k(x_k)$

Control policy: $\mu_k : \mathcal{X} \mapsto \mathcal{U}(x_k)$

CCDP: Challenges and Approaches

$$\min_{\boldsymbol{\mu}} \mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k)) \right\}$$

$$\text{s.t.} \qquad \Pr\left\{ \bigwedge_{k=1}^{N} x_k \in \mathcal{X}_k \;\middle|\; x_0 \right\} \geq 1 - \Delta$$

Risk bound

Challenge 1: Standard DP cannot solve a constrained optimization
Approach 1: Solve the dual optimization problem
Challenge 2: Cannot obtain exact solution (mainly due to duality gap)
Approach 2: Obtain error bound

State constraint: $x_k \in \mathcal{X}_k$

Control constraint: $u_k \in \mathcal{U}_k(x_k)$

Control policy: $\mu_k : \mathcal{X} \mapsto \mathcal{U}(x_k)$

CCDP: Dual Reformulation

$$\min_{\boldsymbol{\mu}} \mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k)) \right\}$$

$$\text{s.t.} \qquad \Pr\left\{ \bigwedge_{k=1}^{N} x_k \in \mathcal{X}_k \;\middle|\; x_0 \right\} \geq 1 - \Delta$$

Indicator function

$$I_k(x_k) = \begin{cases} 1 & (x_k \notin \mathcal{X}_k) \\ 0 & (\text{Otherwise}) \end{cases}$$

CCDP: Dual Reformulation

$$\min_{\boldsymbol{\mu}} \mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k)) \right\}$$

$$\text{s.t.} \qquad \Pr\left\{ \bigwedge_{k=1}^{N} x_k \in \mathcal{X}_k \;\middle|\; x_0 \right\} = 1 - \Pr\left\{ \bigvee_{k=1}^{N} x_k \notin \mathcal{X}_k \mid x_0 \right\}$$

De Morgan's law

$$\geq 1 - \sum_{k=1}^{N} \Pr\left\{ x_k \notin \mathcal{X}_k \mid x_0 \right\}$$

Boole's ineq.

$$= 1 - \sum_{k=1}^{N} \mathbb{E}\{ I_k(x_k) \mid x_0 \}$$

$$= 1 - \mathbb{E}\left\{ \sum_{k=1}^{N} I_k(x_k) \mid x_0 \right\}$$

Indicator function

$$I_k(x_k) = \begin{cases} 1 & (x_k \notin \mathcal{X}_k) \\ 0 & (\text{Otherwise}) \end{cases}$$

Risk at kth time step

$$\Pr\left\{ x_k \notin \mathcal{X}_k \mid x_0 \right\}$$

n time step

CCDP: Dual Reformulation

$$\min_{\boldsymbol{\mu}} \mathbb{E}\left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k)) \right\}$$

$$\text{s.t.} \qquad \mathbb{E}\left\{ \sum_{k=1}^{N} I_k(x_k) \,\middle|\, x_0 \right\} \leq \Delta$$

Risk bound

Indicator function

$$I_k(x_k) = \begin{cases} 1 & (x_k \notin \mathcal{X}_k) \\ 0 & (\text{Otherwise}) \end{cases}$$

Lagrangian

$$L_k^\lambda(x_k, u_k) = \begin{cases} g_0(x_0, u_0) & (k = 0) \\ g_k(x_k, u_k) + \lambda I_k(x_k) & (k = 1 \cdots N-1) \\ g_N(x_N) + \lambda I_N(x_N) & (k = N). \end{cases}$$

λ: Dual variable (Lagrange multiplier)

CCDP: Dual Reformulation

Dual optimization problem

$$\max_{\lambda \geq 0} \min_{\boldsymbol{\mu}} \mathbb{E}\left\{ \sum_{k=0}^{N} L_k^{\lambda}(x_k, \mu_k(x_k)) \right\} - \lambda \Delta$$

Indicator function

$$I_k(x_k) = \begin{cases} 1 & (x_k \notin \mathcal{X}_k) \\ 0 & (\text{Otherwise}) \end{cases}$$

Lagrangian

$$L_k^{\lambda}(x_k, u_k) = \begin{cases} g_0(x_0, u_0) & (k = 0) \\ g_k(x_k, u_k) + \lambda I_k(x_k) & (k = 1 \cdots N-1) \\ g_N(x_N) + \lambda I_N(x_N) & (k = N). \end{cases}$$

λ: Dual variable (Lagrange multiplier)

CCDP: Dual Reformulation

Dual optimization problem

$$\max_{\lambda \geq 0} \min_{\boldsymbol{\mu}} \underline{\mathbb{E}\left\{\sum_{k=0}^{N} L_k^{\lambda}(x_k, \mu_k(x_k))\right\}} - \lambda\Delta$$

Good news 1: Control policy (primal var) can be optimized by standard DP

CCDP: Dual Reformulation

Dual optimization problem

$$\max_{\lambda \geq 0} \boxed{\min_{\boldsymbol{\mu}} \mathbb{E}\left\{\sum_{k=0}^{N} L_k^{\lambda}(x_k, \mu_k(x_k))\right\} - \lambda \Delta} = q(\lambda)$$

Good news 1: Control policy (primal var) can be optimized by standard DP

CCDP: Dual Solution

Dual optimization problem

$$\max_{\lambda \geq 0} \; q(\lambda)$$

Good news 2: Dual optimization can be solved by zero-finding

- Condition for optimality: $0 \in \partial q(\lambda)$ (∂q : subgradient)

$$\underbrace{r_0^\lambda(x_0) - \Delta}_{\text{Excess risk}} \in \partial q(\lambda)$$

Risk-to-go function

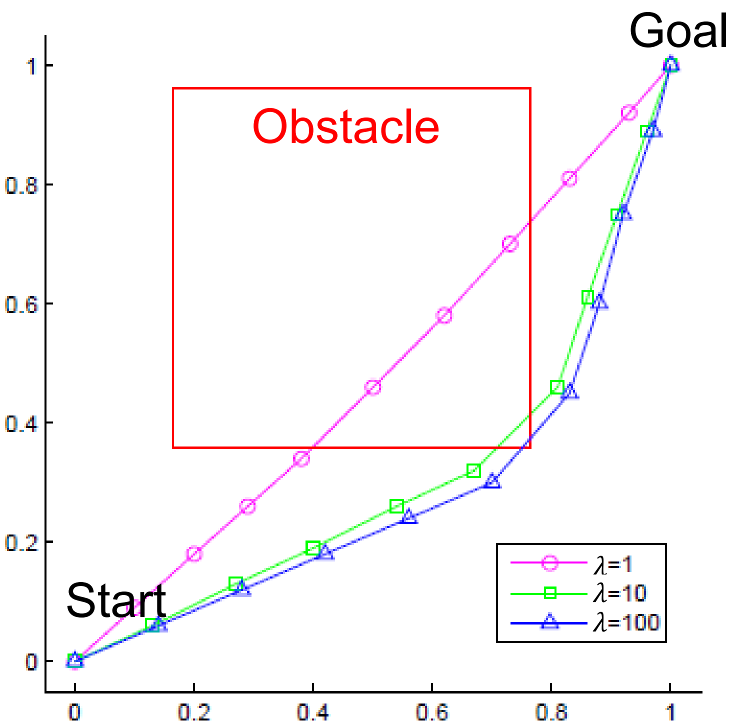
$$r_0^\lambda(x_0) := \mathbb{E}\left\{ \sum_{k=1}^{N} I_k(x_k) \mid x_0, \boldsymbol{\mu}^\lambda \right\}$$

Probability of failure given the optimal control policy $\boldsymbol{\mu}^\lambda$

Intuition: Risk Management via Penalty

λ: Penalty of failure

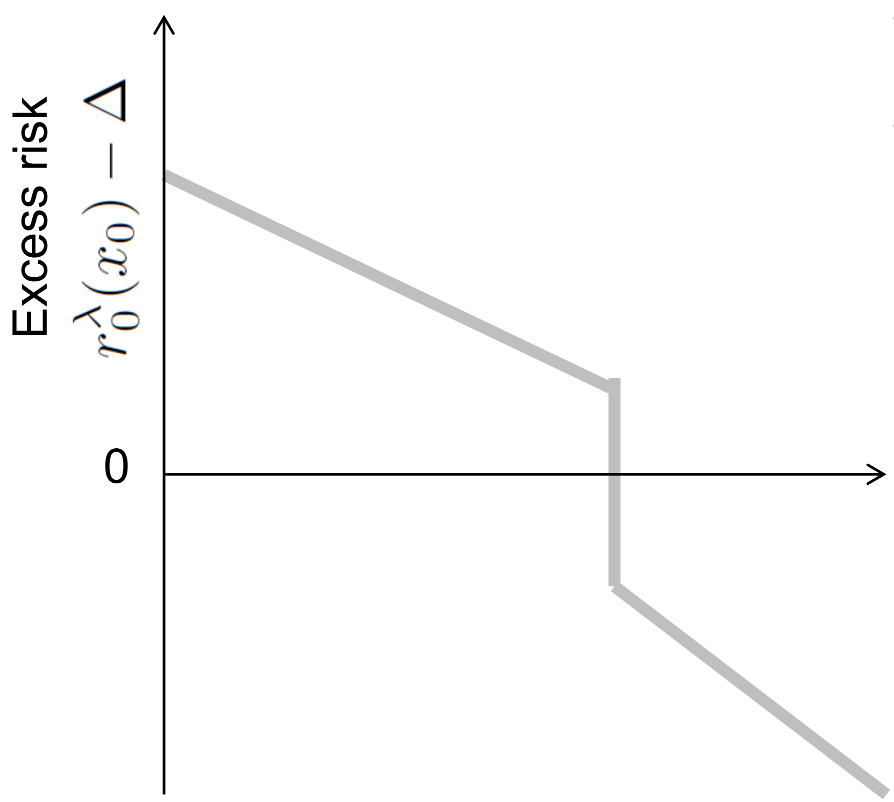
$$\max_{\lambda \geq 0} \min_{\boldsymbol{\mu}} \mathbb{E}\left\{ \sum_{k=0}^{N} \left(g_k(x_k, u_k) + \underbrace{\lambda I_k(x_k)}_{\text{Penalty cost of failure}} \right) \right\} - \lambda\Delta$$



Penalty λ	Prob. of failure
1	100.0%
10	37.5%
100	3.1%

Interpretation: CCDP finds the level of penalty with which the risk of failure is Δ.

Algorithm

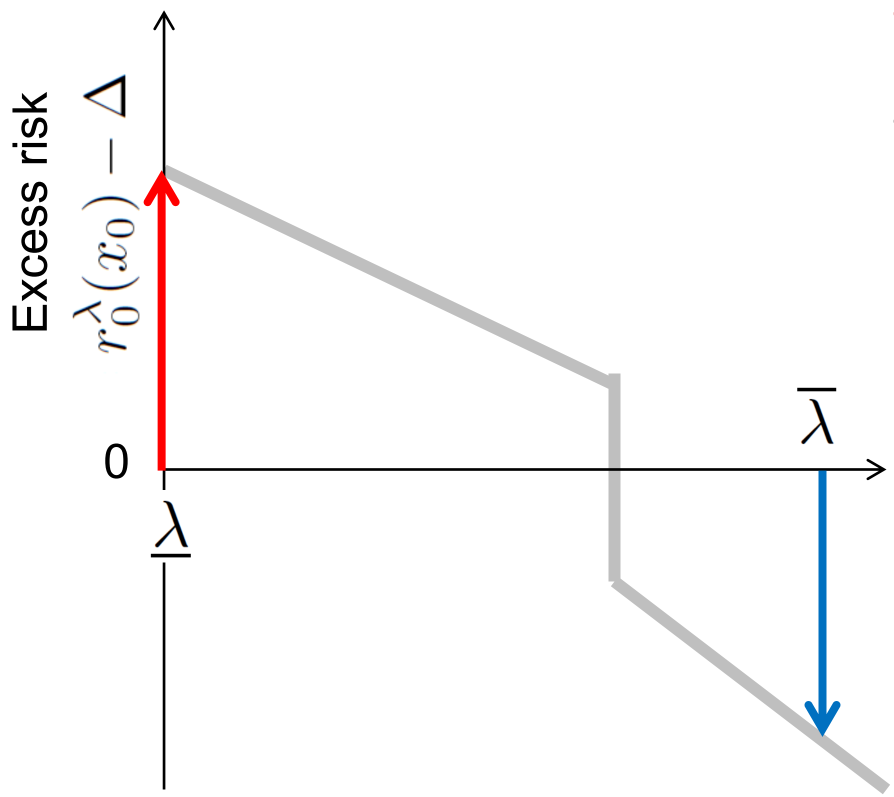


- Initialize $[\underline{\lambda} \;\; \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda} \;\; \overline{\lambda})$
  - Compute optimal control policy μ^{λ} by solving a DP
  - Evaluate risk r^{λ}
  - If $r^{\lambda} - \Delta > 0$

    $\underline{\lambda} \leftarrow \lambda$
  - Else

    $\overline{\lambda} \leftarrow \lambda$
- end while

Algorithm

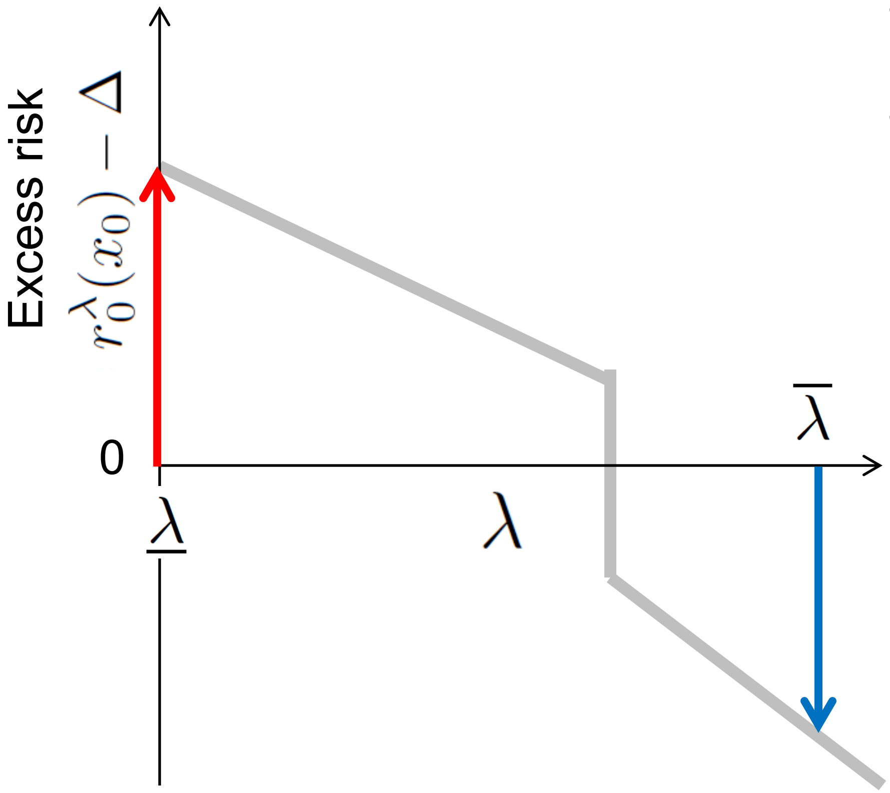


- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$
  - Compute optimal control policy μ^λ by solving a DP
  - Evaluate risk r^λ
  - If $r^\lambda - \Delta > 0$

    $\underline{\lambda} \leftarrow \lambda$
  - Else

    $\overline{\lambda} \leftarrow \lambda$
- end while

Algorithm

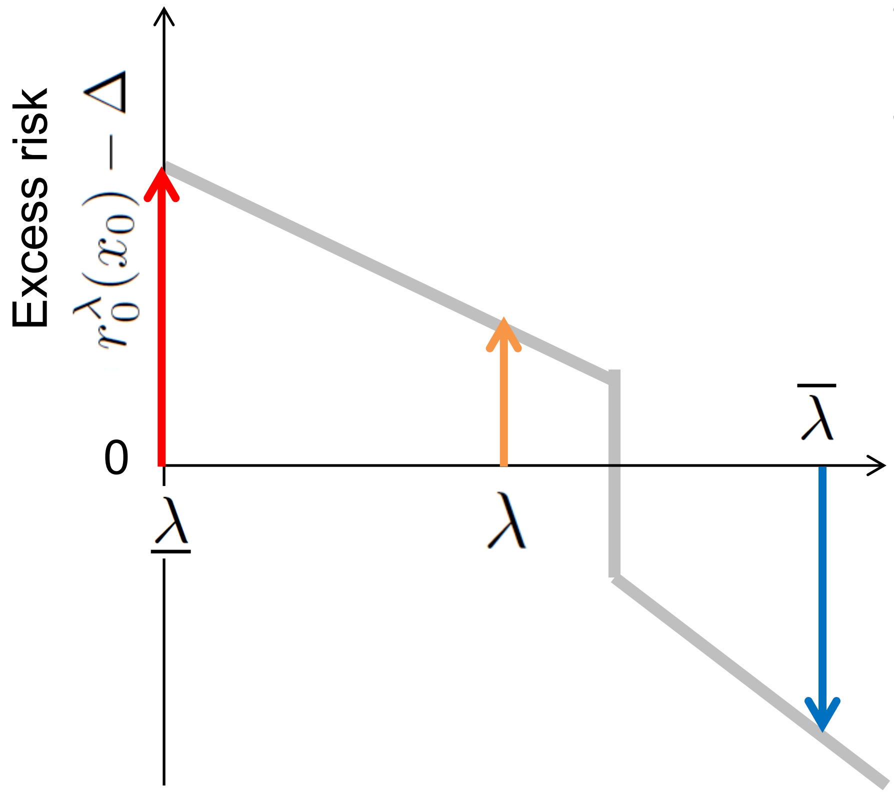


- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$
  - Compute optimal control policy μ^{λ} by solving a DP
  - Evaluate risk r^{λ}
  - If $r^{\lambda} - \Delta > 0$

    $\underline{\lambda} \leftarrow \lambda$
  - Else

    $\overline{\lambda} \leftarrow \lambda$
- end while

Algorithm

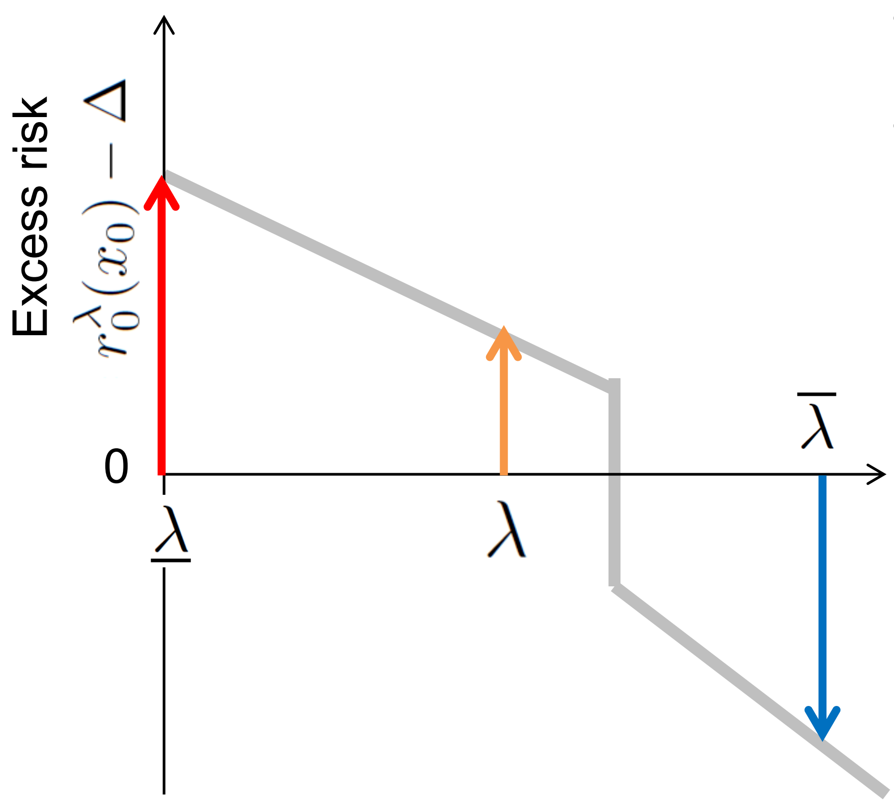


- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$
  - Compute optimal control policy μ^λ by solving a DP
  - Evaluate risk r^λ
  - If $r^\lambda - \Delta > 0$

    $\underline{\lambda} \leftarrow \lambda$
  - Else

    $\overline{\lambda} \leftarrow \lambda$
- end while

Algorithm

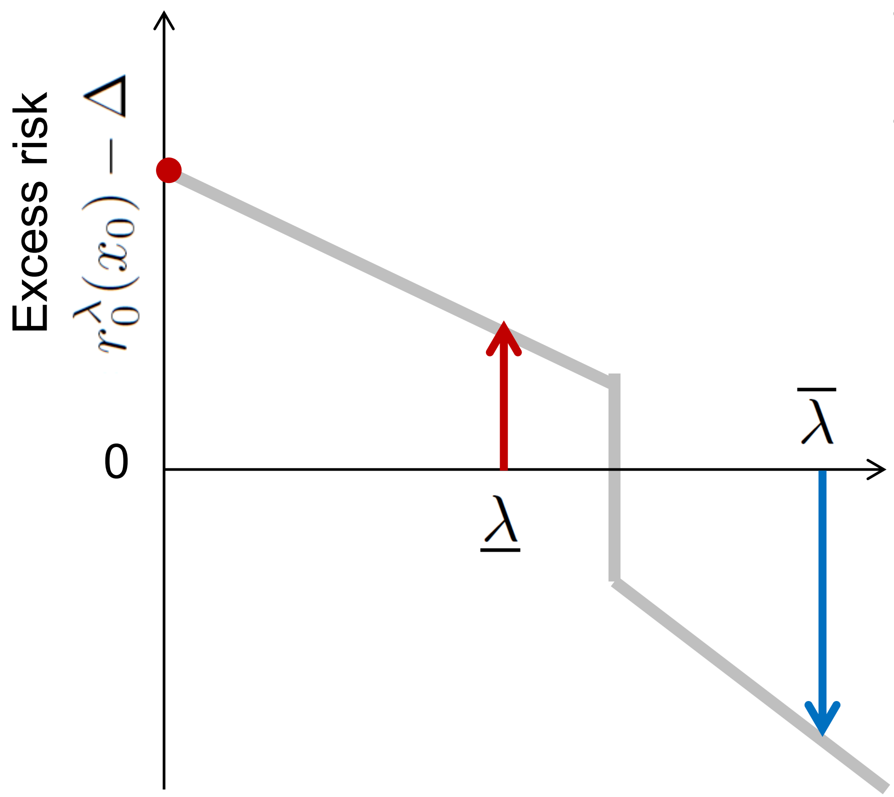


- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$
  - Compute optimal control policy μ^λ by solving a DP
  - Evaluate risk r^λ
  - If $r^\lambda - \Delta > 0$

    $\underline{\lambda} \leftarrow \lambda$
  - Else

    $\overline{\lambda} \leftarrow \lambda$
- end while

Algorithm

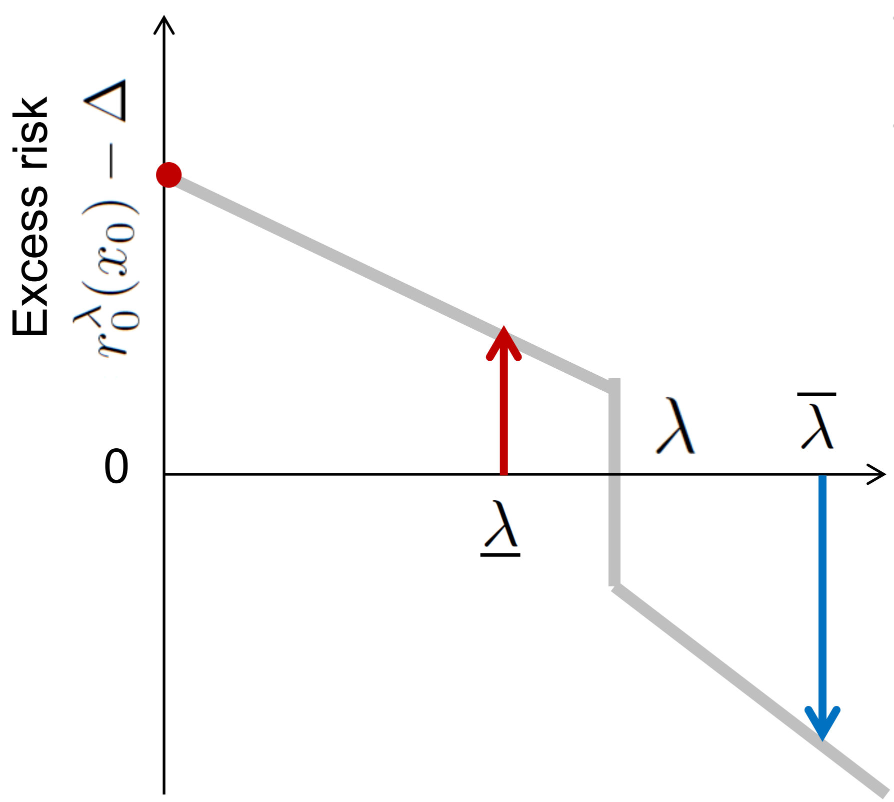


- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$
  - Compute optimal control policy μ^λ by solving a DP
  - Evaluate risk r^λ
  - If $r^\lambda - \Delta > 0$

    $\underline{\lambda} \leftarrow \lambda$
  - Else

    $\overline{\lambda} \leftarrow \lambda$
- end while

Algorithm

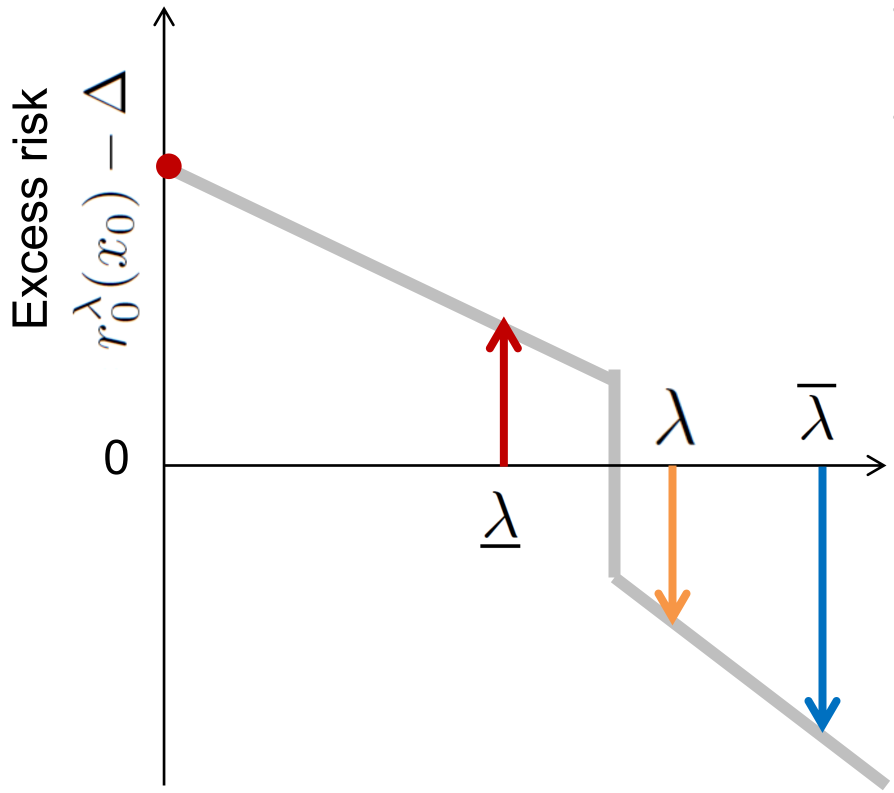


- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$
  - Compute optimal control policy μ^λ by solving a DP
  - Evaluate risk r^λ
  - If $r^\lambda - \Delta > 0$

    $\underline{\lambda} \leftarrow \lambda$
  - Else

    $\overline{\lambda} \leftarrow \lambda$
- end while

Algorithm

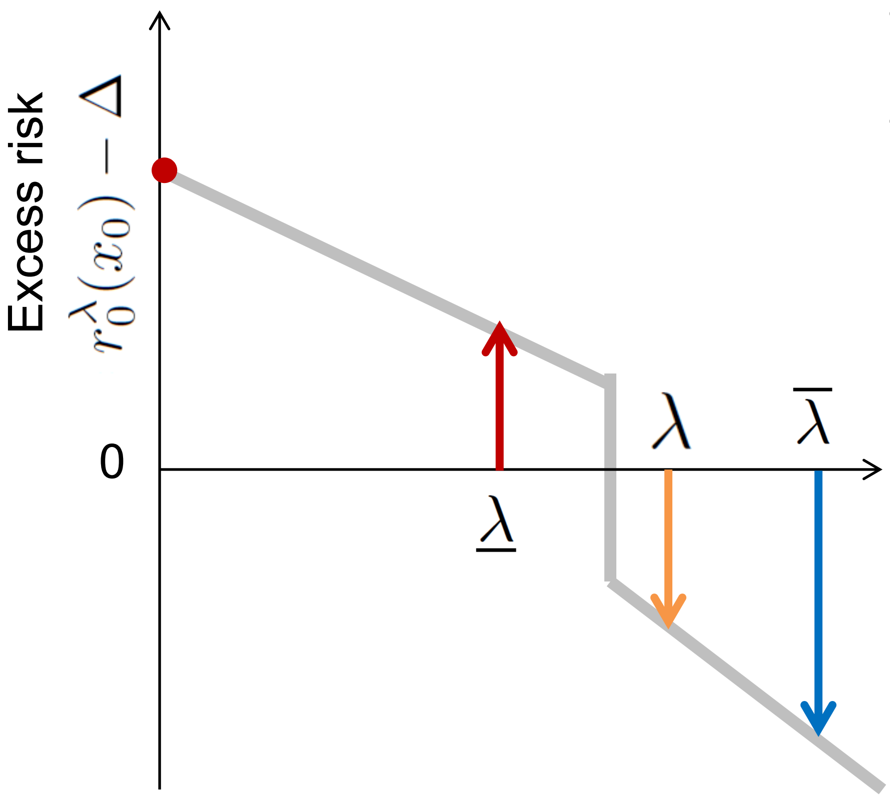


- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$
  - Compute optimal control policy μ^λ by solving a DP
  - Evaluate risk r^λ
  - If $r^\lambda - \Delta > 0$

    $\underline{\lambda} \leftarrow \lambda$
  - Else

    $\overline{\lambda} \leftarrow \lambda$
- end while

Algorithm

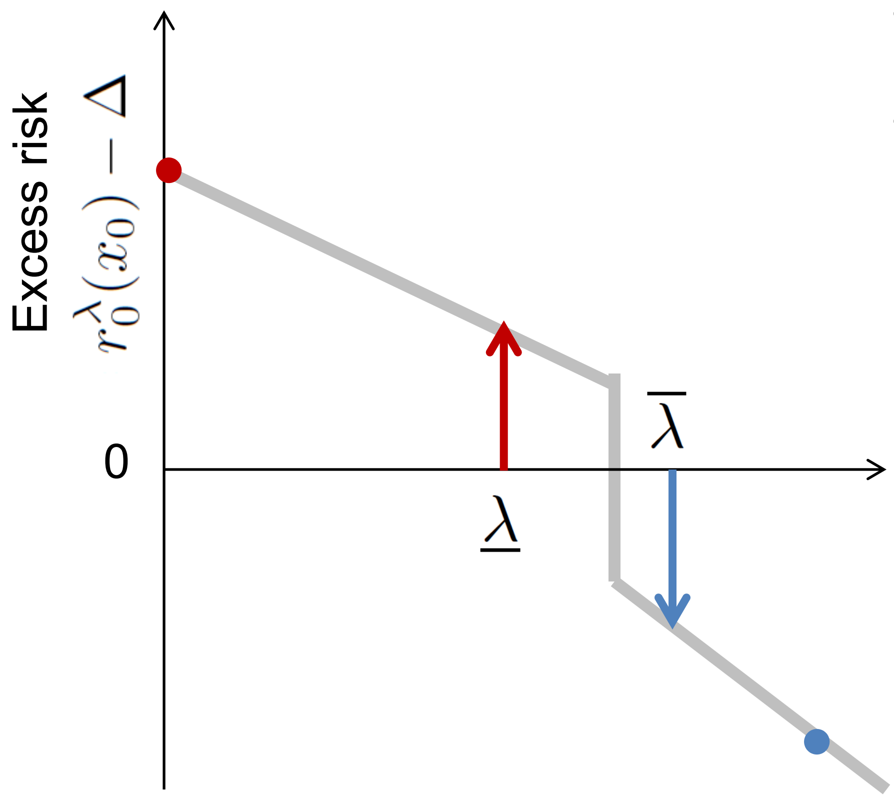


- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$
  - Compute optimal control policy μ^{λ} by solving a DP
  - Evaluate risk r^{λ}
  - If $r^{\lambda} - \Delta > 0$

    $\underline{\lambda} \leftarrow \lambda$
  - Else

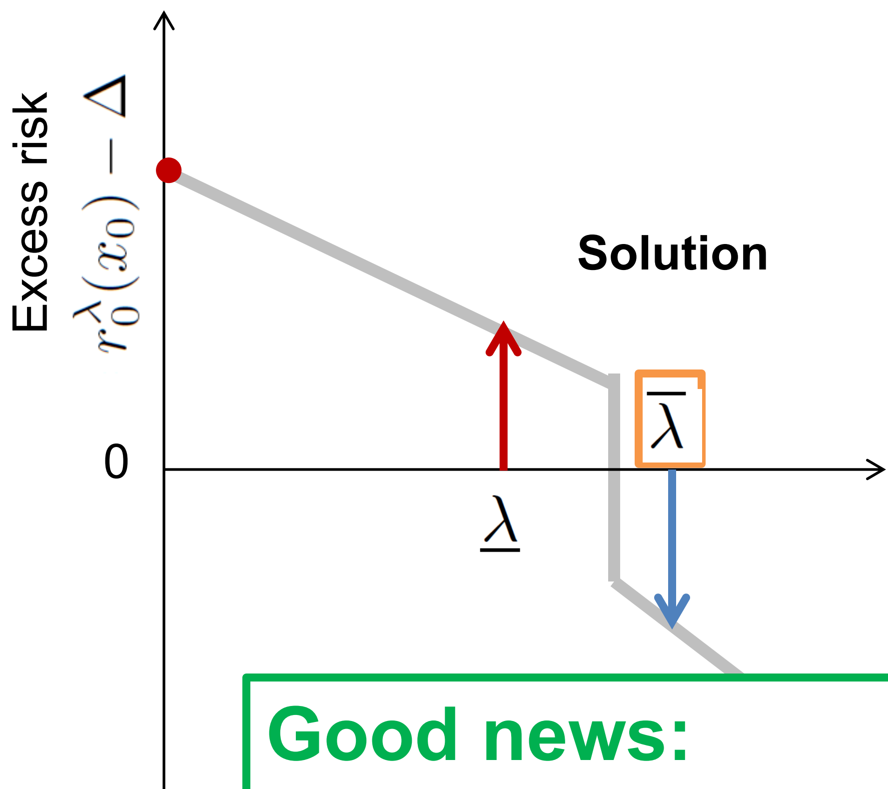
    $\overline{\lambda} \leftarrow \lambda$
- end while

Algorithm



- Initialize $[\underline{\lambda} \ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda} \ \overline{\lambda})$
  - Compute optimal control policy μ^{λ} by solving a DP
  - Evaluate risk r^{λ}
  - If $r^{\lambda} - \Delta > 0$
    $\underline{\lambda} \leftarrow \lambda$
  - Else
    $\overline{\lambda} \leftarrow \lambda$
- end while

Algorithm

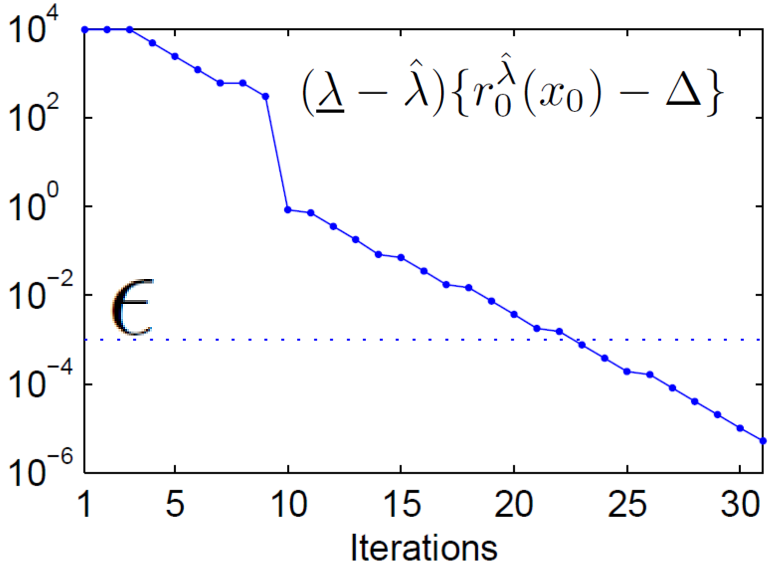


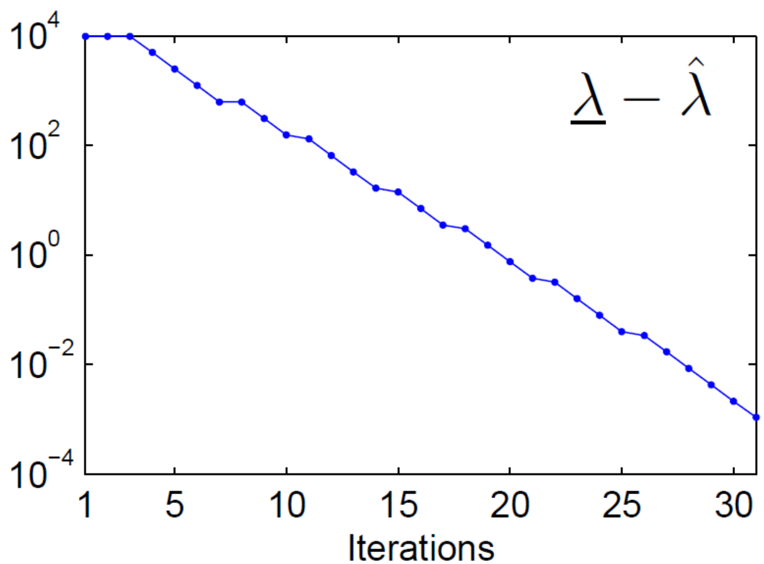
- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$
  - Compute optimal control policy μ^λ by solving a DP
  - Evaluate risk r^λ
  - If $r^\lambda - \Delta > 0$

    $\lambda \leftarrow \lambda$

Good news:
- Solution is on the "safe side"
- Guaranteed to satisfy the risk constraint

Algorithm





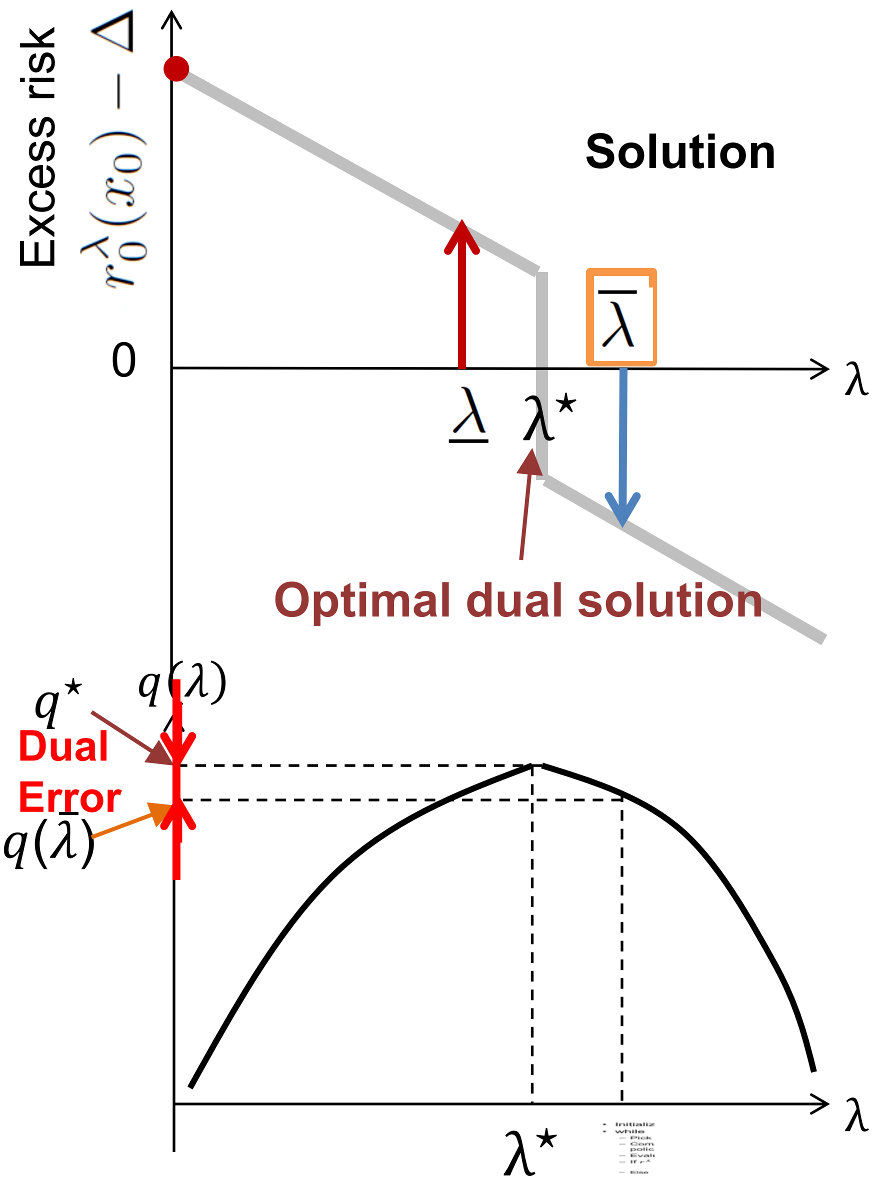
E. Margaritifer, 2k x 2k, $\Delta = 10^{-3}$, $\epsilon = 10^{-3}$

- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda}-\overline{\lambda})\{r_0^{\overline{\lambda}}(x_0)-\Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$

Good news:

- Dual optimization converges exponentially
- Typically requires 10-20 iterations
- # of iterations does not grow with problem size
  - No exponential blowup

Bound on Solution Error

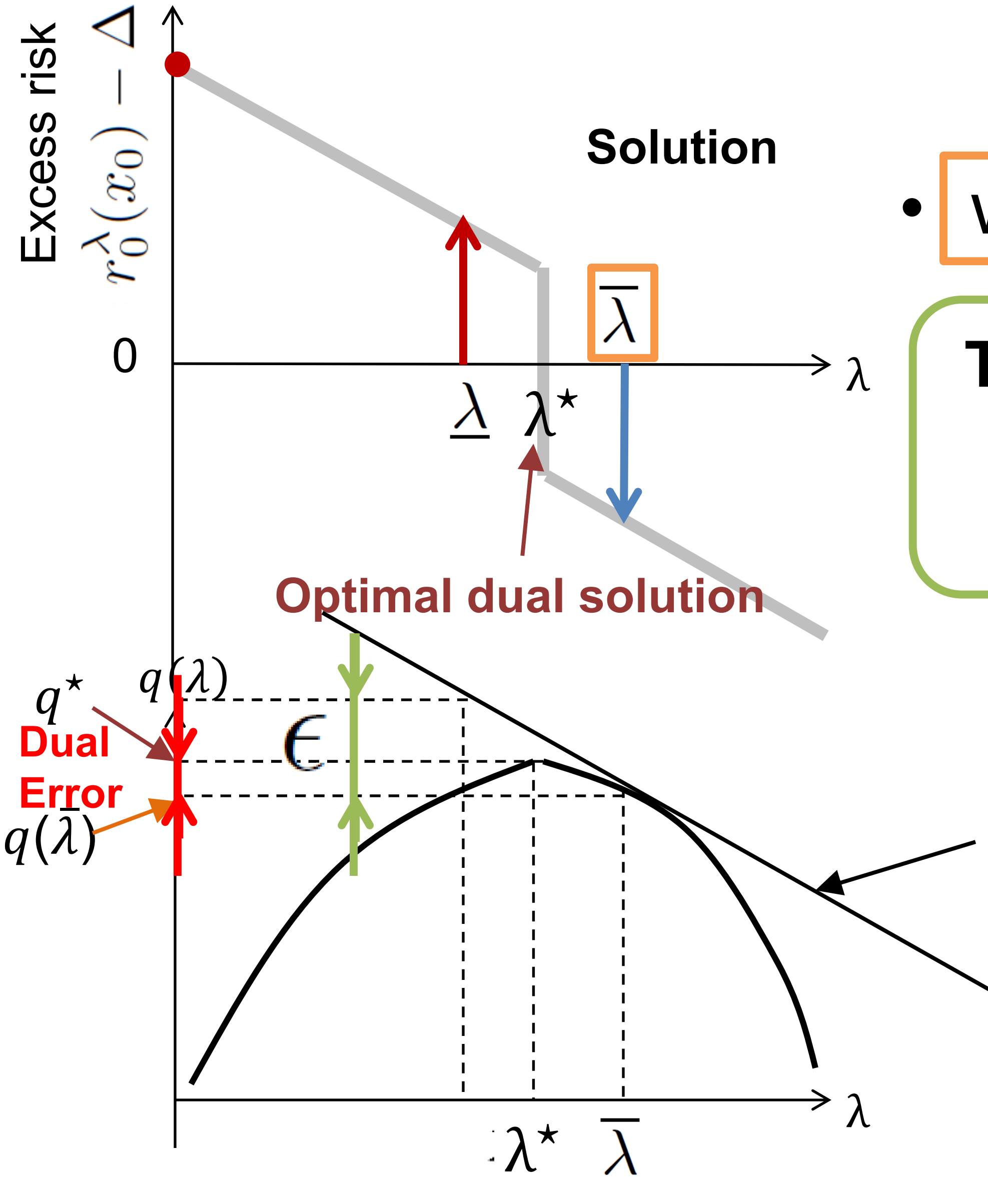


- Initialize $[\underline{\lambda}\ \overline{\lambda}]$
- while $(\underline{\lambda} - \overline{\lambda})\{r_0^{\overline{\lambda}}(x_0) - \Delta\} > \epsilon$
  - Pick $\lambda \in (\underline{\lambda}\ \overline{\lambda})$
  - Compute optimal control policy μ^λ by solving a DP
  - Evaluate risk r^λ
  - If $r^\lambda - \Delta > 0$

    $\underline{\lambda} \leftarrow \lambda$
  - Else

    $\overline{\lambda} \leftarrow \lambda$
- end while

Bound on Solution Error



- while $(\underline{\lambda} - \bar{\lambda})\{r_0^{\bar{\lambda}}(x_0) - \Delta\} > \epsilon$

Theorem 1

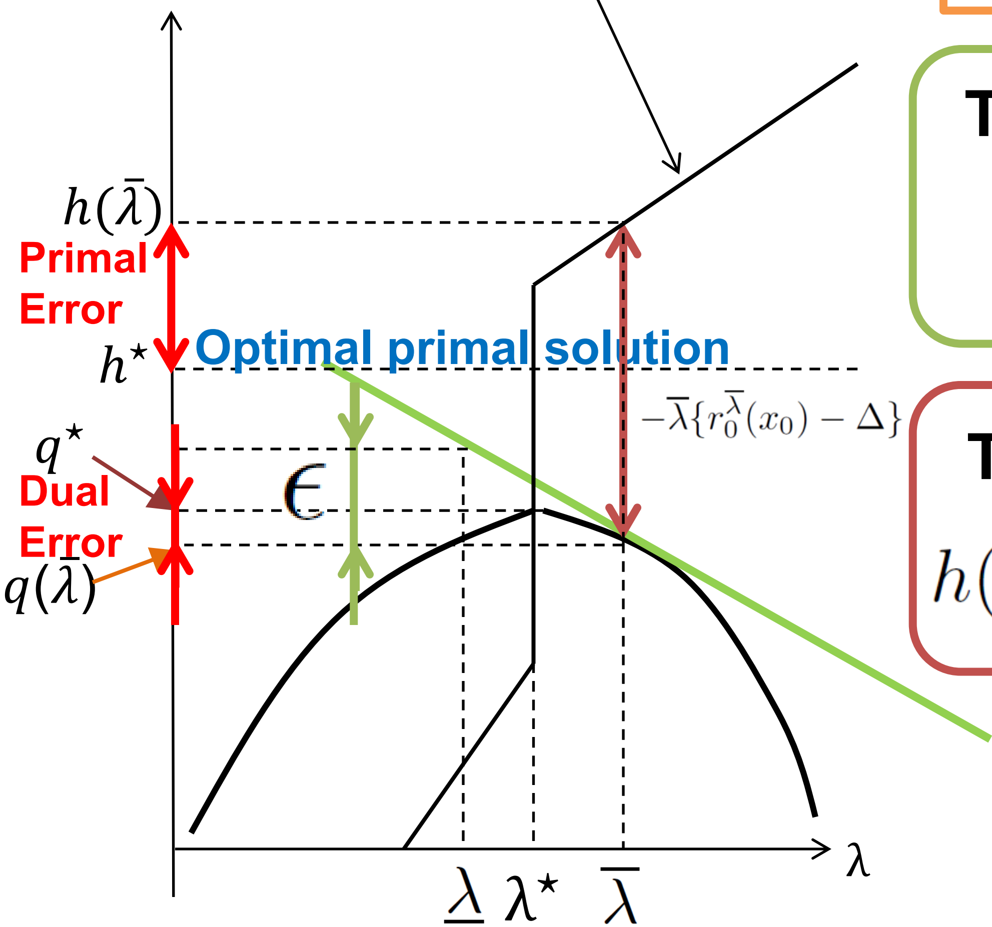
$$q^\star - q(\bar{\lambda}) \leq \epsilon$$

Slope: $\underline{\lambda} - \bar{\lambda}$

Bound on Solution Error

Cost with the optimal policy given λ

$$h(\lambda) = q(\lambda) - \lambda\{r_0^{\lambda}(x_0) - \Delta\}$$



- while $(\underline{\lambda} - \bar{\lambda})\{r_0^{\bar{\lambda}}(x_0) - \Delta\} > \epsilon$

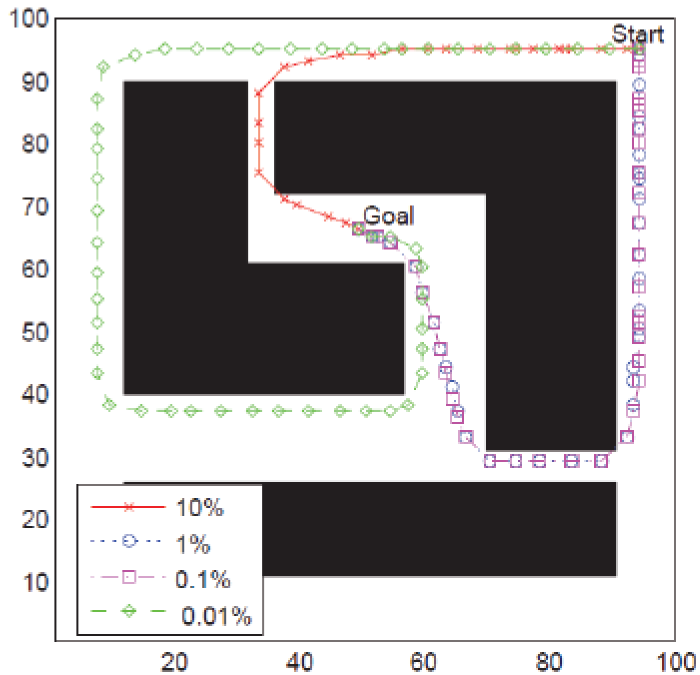
Theorem 1

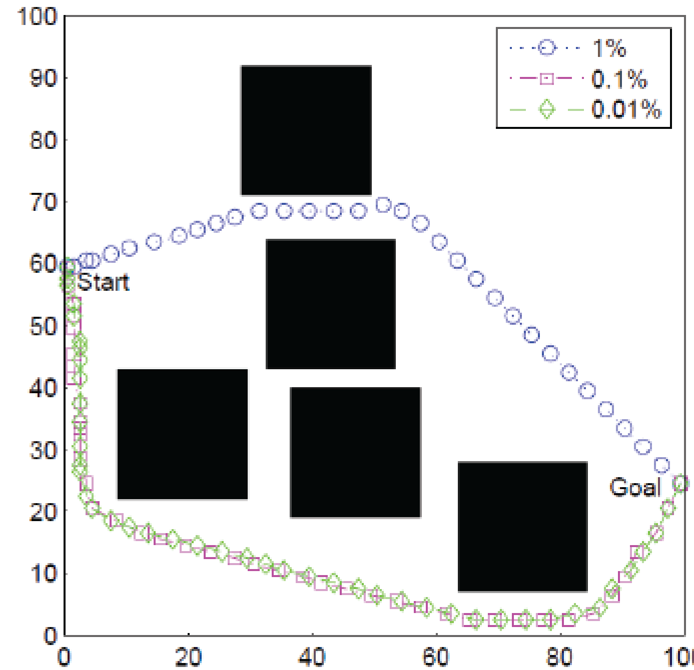
$$q^\star - q(\bar{\lambda}) \leq \epsilon$$

Theorem 2

$$h(\bar{\lambda}) - h^\star \leq -\bar{\lambda}\{r_0^{\bar{\lambda}}(x_0) - \Delta\}$$

Application 1: Path Planning





Dynamics

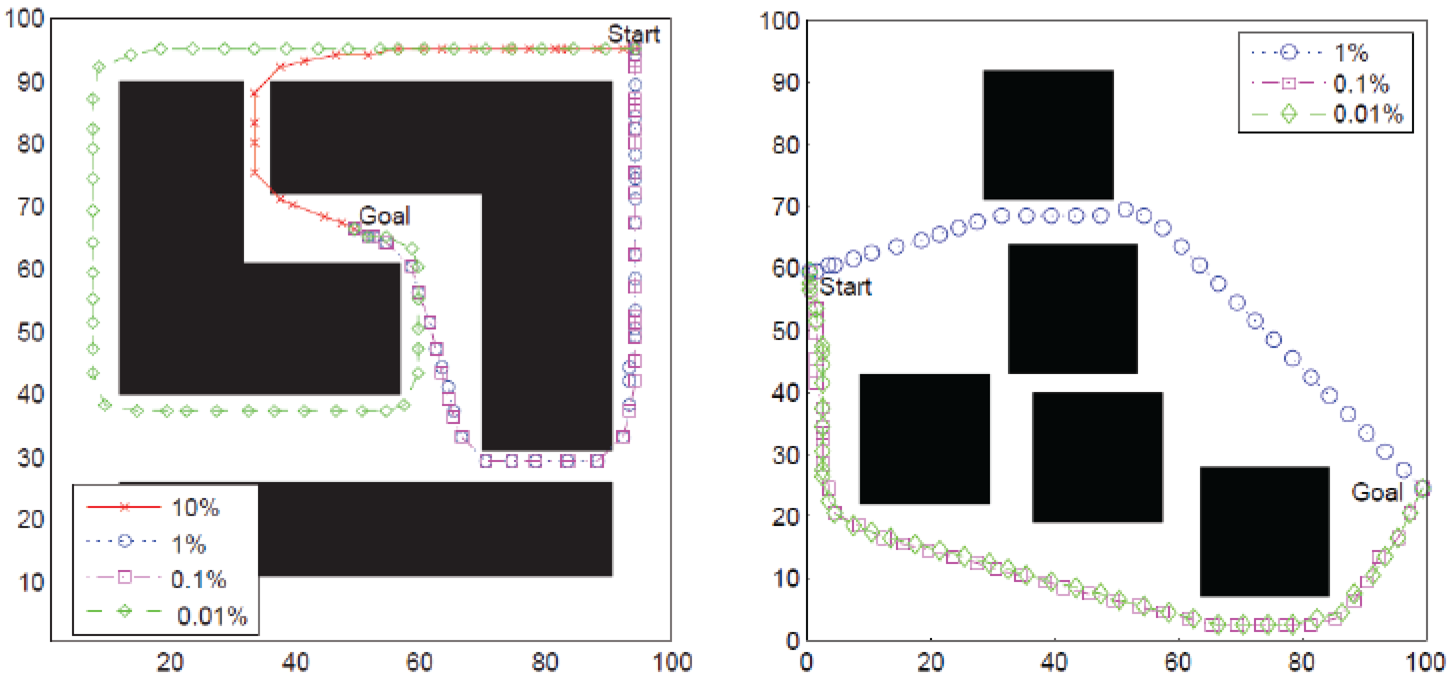
$$x_{k+1} = x_k + u_k + w_k$$

$$\|u_k\|_2 \leq d_k, \quad w_k \sim \mathcal{N}(0, \Sigma_k)$$

State space: 100 x 100 grid

Cost: expected path length from start to goal

Application 1: Path Planning

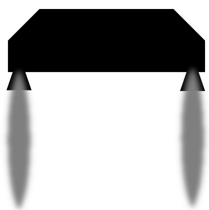


Risk bound	Cost	Length of nominal path	Computation time [sec]
$\Delta = 1\%$	0.88551 ± 0.0176	82.58 ± 25.10	17.2 ± 5.4
$\Delta = 0.1\%$	0.88555 ± 0.0175	86.74 ± 27.94	14.9 ± 4.6
$\Delta = 0.01\%$	0.88556 ± 0.0175	87.21 ± 28.11	12.2 ± 4.7

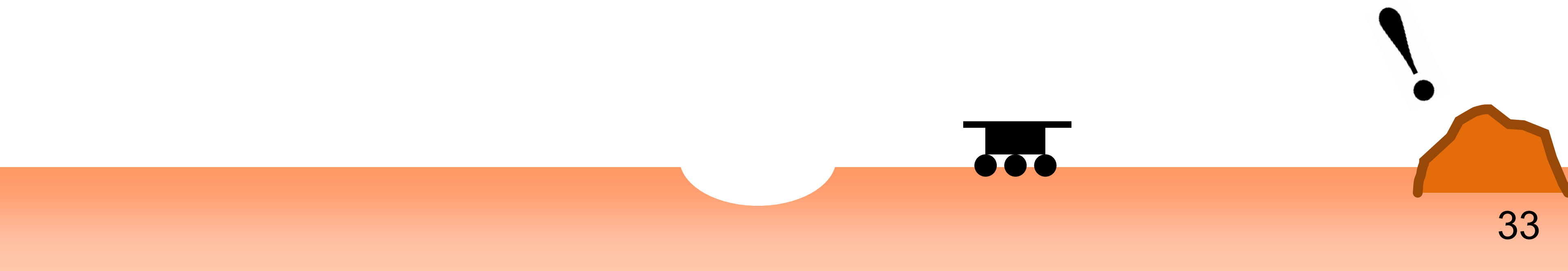
Application 2: Mars Entry, Descent & Landing

Landing site selection ($k = 0$) → **PDG** ($k = 1$) → **HDA** ($k = 2$) → **Surface mobility** ($k = 3$)

Powered **d**escent **g**uidance



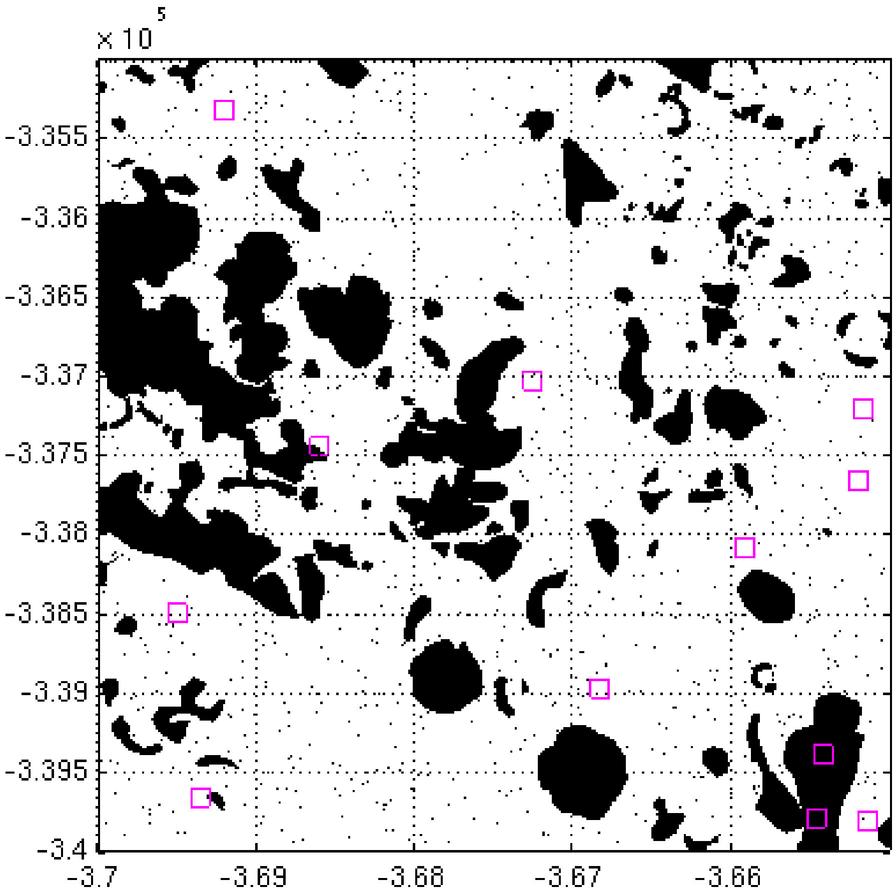
Hazard **d**etection & **a**voidance



Mars EDL Problem Formulation

Landing site selection → PDG → HDA → Landing

$k = 0$ | $k = 1$ | $k = 2$ | $k = 3$



Map obtained from Mars Reconnaissance Orbiter

Dynamics

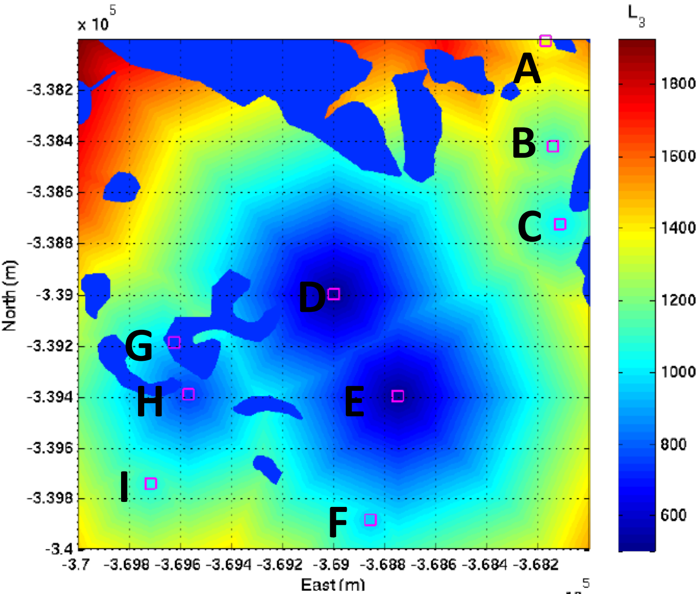
$$x_{k+1} = x_k + u_k + w_k$$

$$\|u_k\|_2 \le d_k, \quad w_k \sim \mathcal{N}(0, \Sigma_k)$$

State space: 2000 x 2000 grid

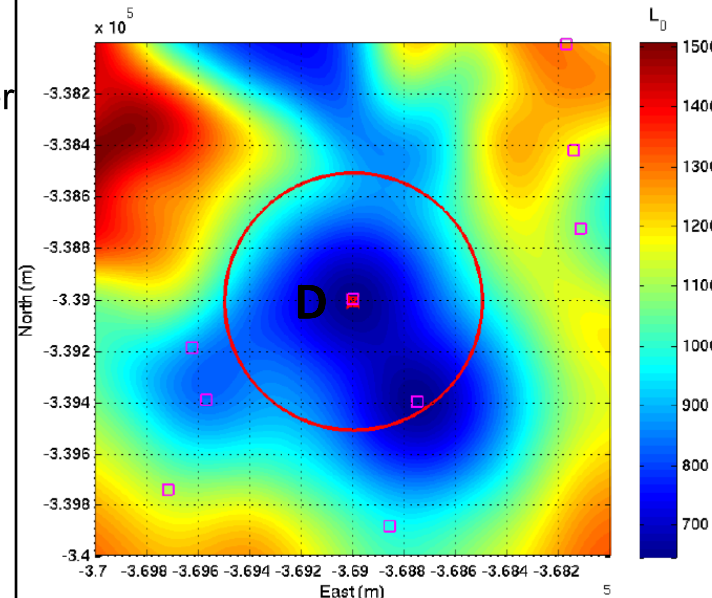
Cost: expected driving distance to visit N science target after landing

Simulation on E. Margaritifer



E. Margaritifer
2 km x 2km
1 m grid

With PDG
and HDA



$\Delta = 1\%$

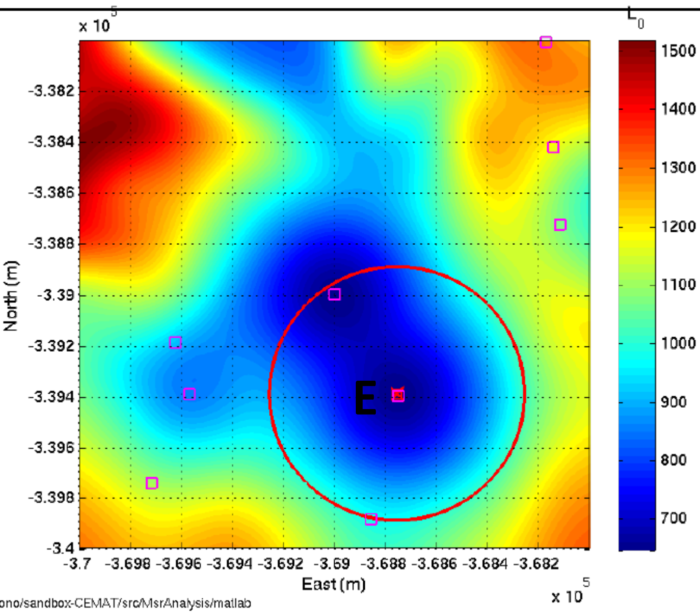
Expected risk

0.990%

Expected
Driving distance

637.81 m

$\bar{\lambda} = 725.16$



$\Delta = 0.1\%$

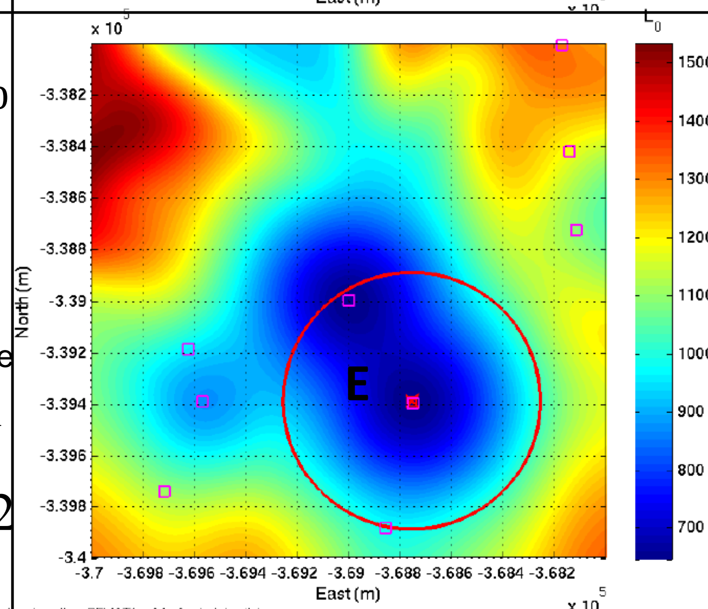
Expected risk

0.0160%

Expected
Driving distance

645.49 m

$\bar{\lambda} = 801.12$



$\Delta = 0.01\%$

Expected risk

0.0094%

Expected
Driving distance

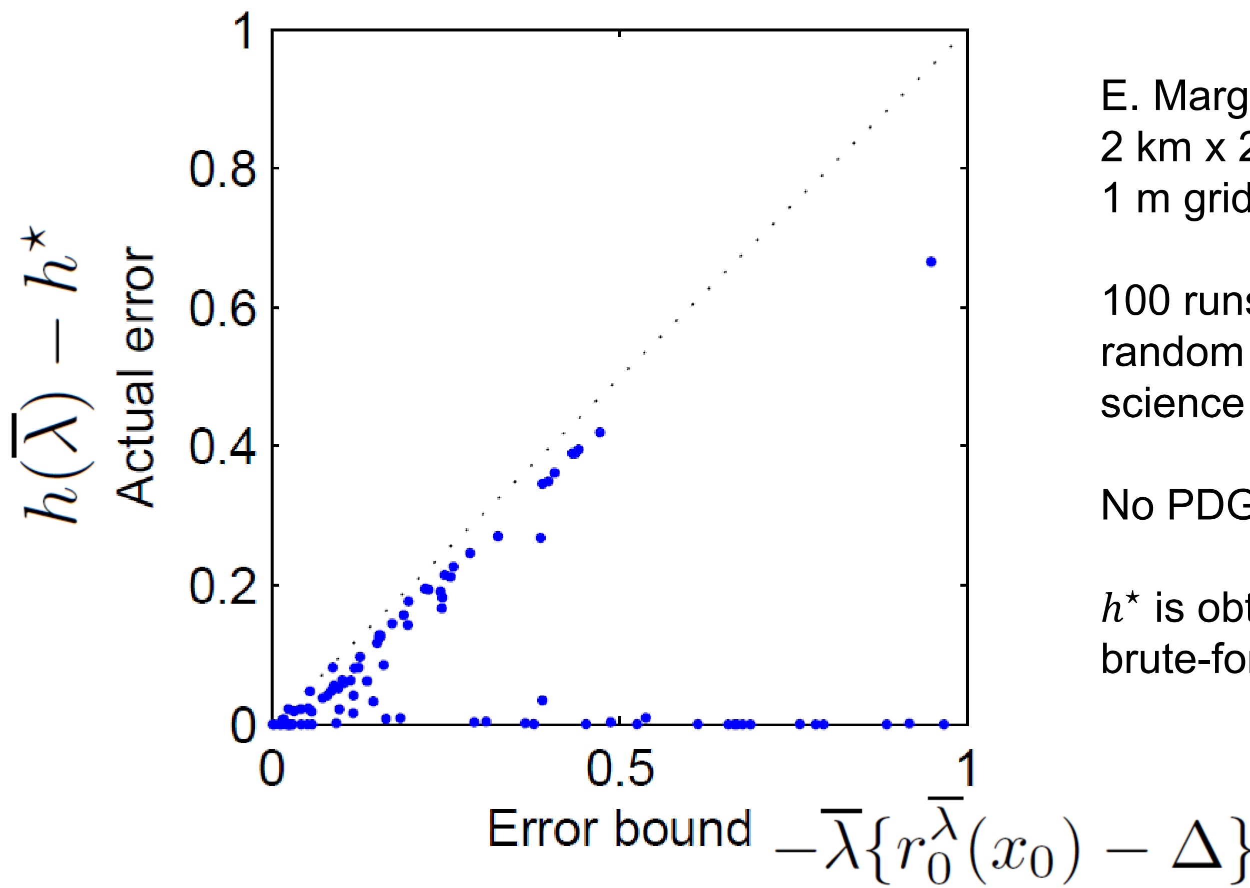
645.55 m

$\bar{\lambda} = 912.94$

Bound on Solution Error

Theorem 2

$$h(\overline{\lambda}) - h^\star \leq -\overline{\lambda}\{r_0^{\overline{\lambda}}(x_0) - \Delta\}$$



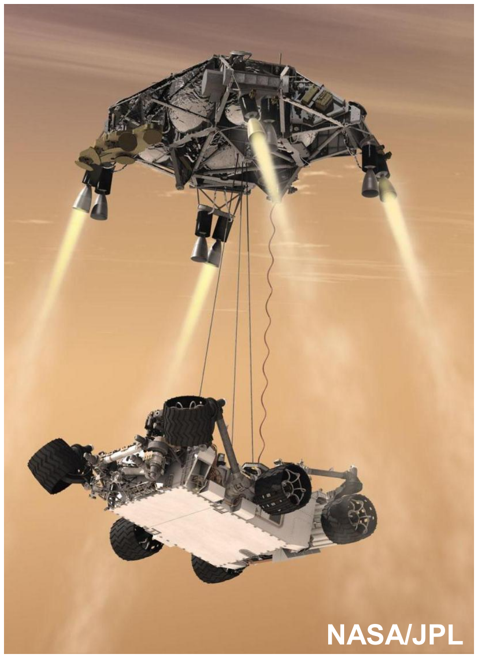
E. Margaritifer
2 km x 2km
1 m grid

100 runs with random location of science targets

No PDG, no HAD

$h^\star$ is obtained by a brute-force method

Conclusion



- Developed a CCDP algorithm that can
  - Make optimal sequential decisions under uncertainty
  - Minimize expected cost within a user-specified risk bound
  - Provide bounds on the solution error
- Demonstrated the CCDP algorithm on
  - Path planning problem
  - Future Mars entry, descent, and landing scenario

Questions