Convergence theory of Trajectory Following Dynamic Programming (a.k.a SDDP & co)

Maël Forcier, Vincent Leclère

Journées SMAI-MODE

CERMICS

June 1st, 2022

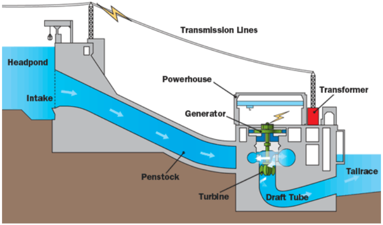
Motivations

- An hydroelectric stock

$$\boldsymbol{s}_t = \boldsymbol{s}_{t-1} - \boldsymbol{u}_t + \xi_t$$

where, at time t:

  - $\boldsymbol{s}_t$ is the amount of water
  - $\boldsymbol{u}_t$ is the water turbined
  - ξ_t is the inflow
  - $\boldsymbol{p}_t$ is the price



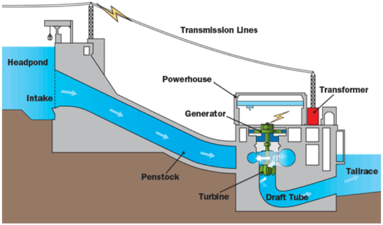
Motivations

- An hydroelectric stock

$$s_t = s_{t-1} - u_t + \xi_t$$

where, at time t:

- s_t is the amount of water
- u_t is the water turbined
- ξ_t is the inflow
- p_t is the price



$$\underset{(u_t)_{t=1:T}}{\text{Min}} \quad \mathbb{E}\left[\sum_{t=1}^{T} -p_t u_t + K(s_T)\right]$$

$$\begin{aligned}
\text{s.t.} \quad & s_0 = s_{init} && \text{(initial stock)} \\
& s_t = s_{t-1} - u_t + \xi_t && \text{(dynamic)} \\
& 0 \leq s_t \leq \bar{s}_t && \text{(state constraints)} \\
& \sigma(u_t) \subset \sigma(\xi_1, \dots, \xi_t) && \text{(information constraints)}
\end{aligned}$$

Contents

Introducing the value function

$$
\begin{aligned}
V_{t_0}(s) := \underset{(\boldsymbol{u}_t)_{t=t_0:T}}{\mathrm{Min}} \quad & \mathbb{E}\left[\sum_{t=t_0}^{T} \boldsymbol{p}_t \boldsymbol{u}_t + K(\boldsymbol{s}_T)\right] & \\
\text{s.t.} \quad & \boldsymbol{s}_{t_0} = s & \text{(Initial stock)} \\
& \boldsymbol{s}_t = \boldsymbol{s}_{t-1} - \boldsymbol{u}_t + \boldsymbol{\xi}_t & \text{(dynamic)} \\
& 0 \leq \boldsymbol{s}_t \leq \bar{s}_t & \text{(state constraints)} \\
& \sigma(\boldsymbol{u}_t) \subset \sigma(\boldsymbol{\xi}_1, \dots, \boldsymbol{\xi}_t) & \text{(information constraints)}
\end{aligned}
$$

- $V_{t_0}(s)$ is the optimal value of the problem starting at t_0 with stock s
- $V_0(s_{init})$ is the value of the original problem
- $\dfrac{dV_t(s)}{ds}$ is the marginal value of stock

Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\xi_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\xi_t}\left[\min_{u_t}\left\{ \underbrace{-p_t u_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - u_t + \xi_t)}_{\text{cost-to-go}} \right\}\right]$$

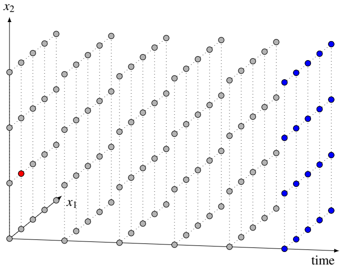
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

```
V_T ≡ K ; V_t ≡ 0
for t : T − 1 → 0 do
    for s ∈ S do
        for ξ ∈ Ξ do
            v̂ = min_{u∈U} −p_t u + V_{t+1}(s − u + ξ)
            V_t(s) += P(ξ_t = ξ) v̂
```



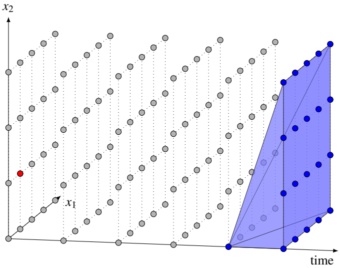
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



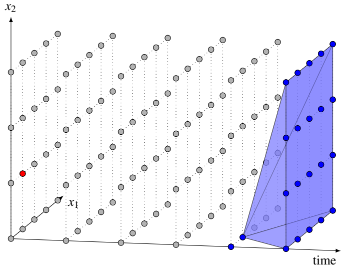
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



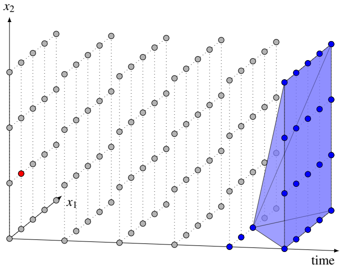
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



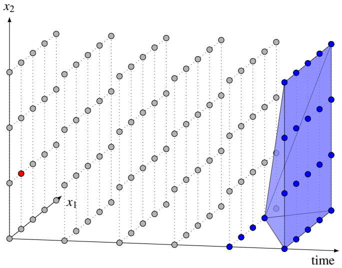
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



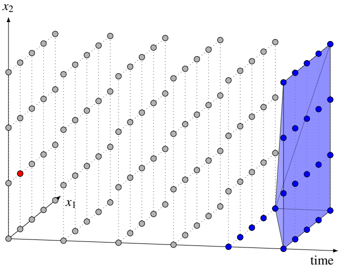
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



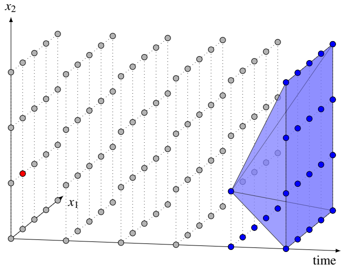
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



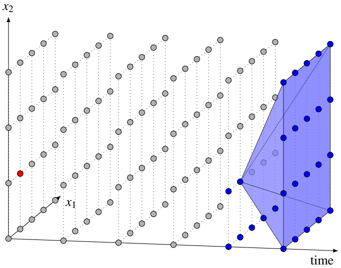
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

```
V_T ≡ K ; V_t ≡ 0
for t : T − 1 → 0 do
    for s ∈ S do
        for ξ ∈ Ξ do
            v̂ = min_{u∈U} −p_t u + V_{t+1}(s − u + ξ)
            V_t(s) += P(ξ_t = ξ) v̂
```



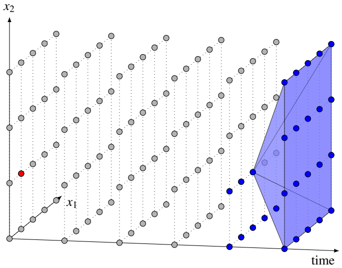
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



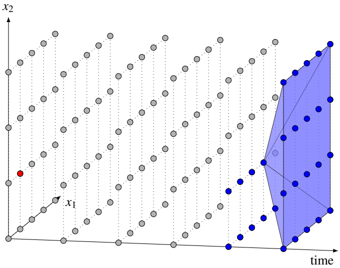
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



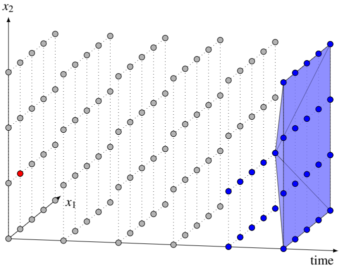
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\xi_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\xi_t}\left[\min_{u_t}\left\{ \underbrace{-p_t u_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - u_t + \xi_t)}_{\text{cost-to-go}} \right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\xi_t = \xi)\hat{v}$



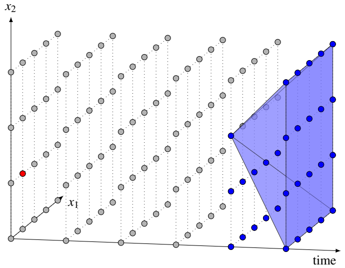
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



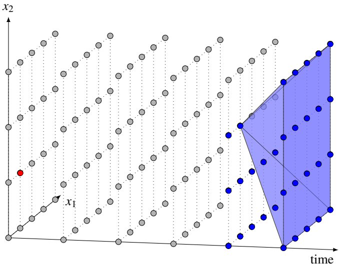
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



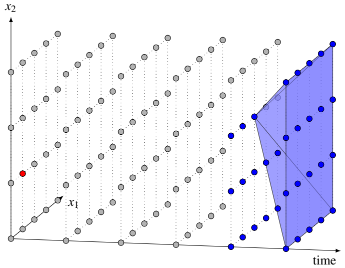
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

```
V_T ≡ K ; V_t ≡ 0
for t : T − 1 → 0 do
    for s ∈ S do
        for ξ ∈ Ξ do
            v̂ = min_{u∈U} −p_t u + V_{t+1}(s − u + ξ)
            V_t(s) += P(ξ_t = ξ) v̂
```



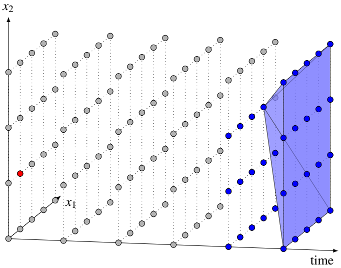
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



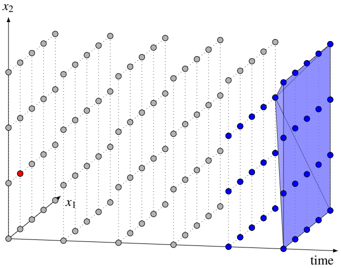
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



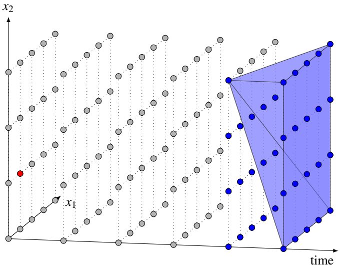
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\xi_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\xi_t}\Big[\min_{u_t}\Big\{\underbrace{-p_t u_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - u_t + \xi_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\xi_t = \xi)\hat{v}$



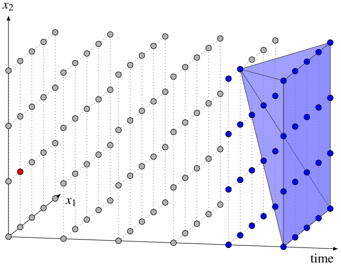
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



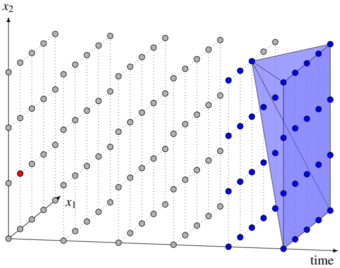
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



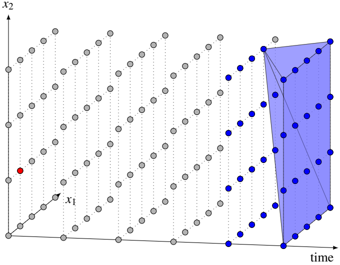
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



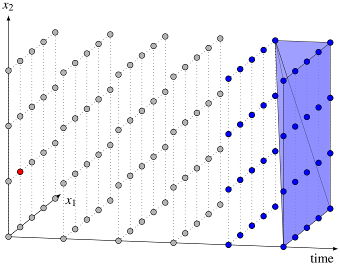
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



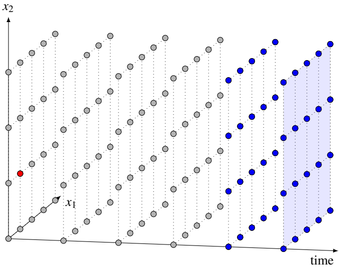
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



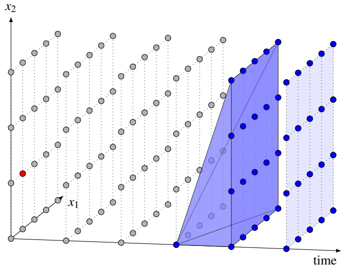
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



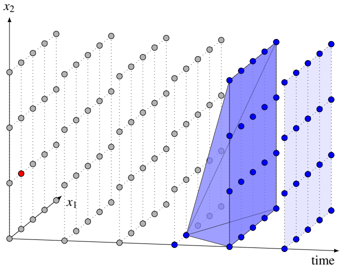
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



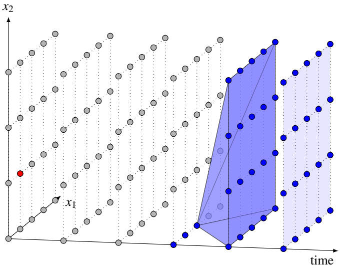
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



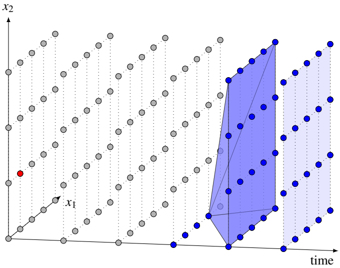
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



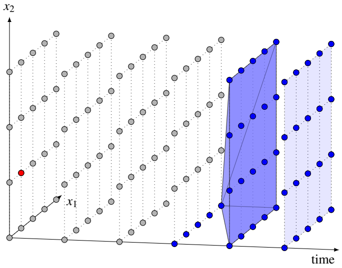
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



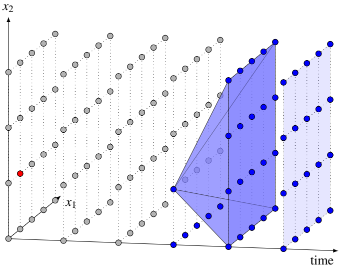
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t \boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}} \right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



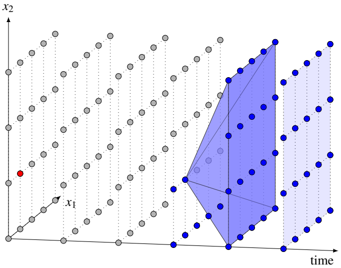
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



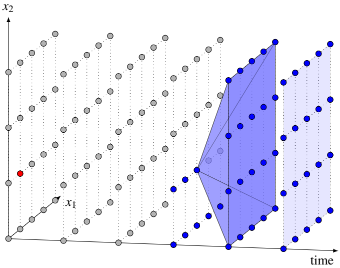
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



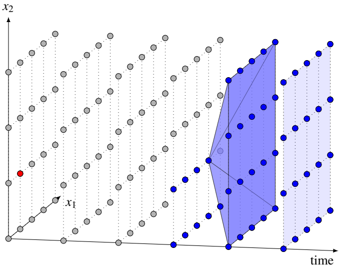
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



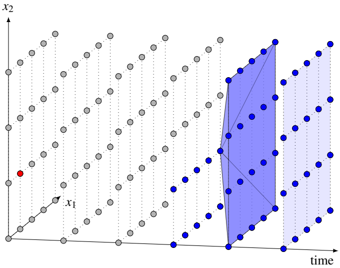
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}} \right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



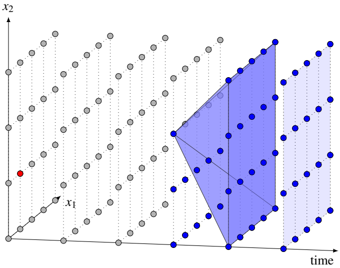
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



Dynamic Programming

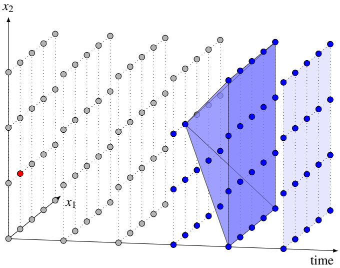
Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

```
V_T ≡ K ; V_t ≡ 0
for t : T − 1 → 0 do
    for s ∈ S do
        for ξ ∈ Ξ do
            v̂ = min_{u∈𝒰} −p_t u + V_{t+1}(s − u + ξ)
            V_t(s) += ℙ(ξ_t = ξ) v̂
```

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



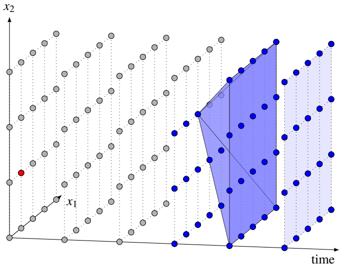
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t \in [T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t \boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u \in \mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



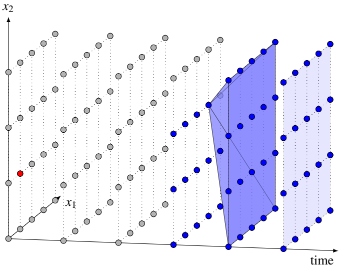
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



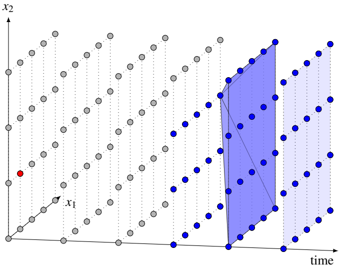
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



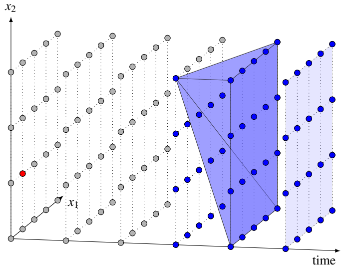
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



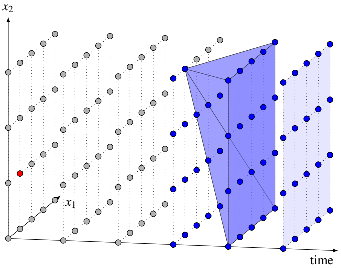
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



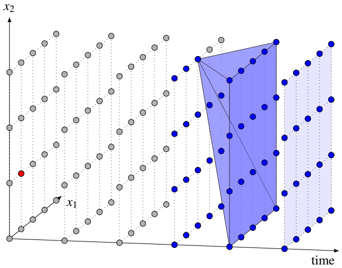
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



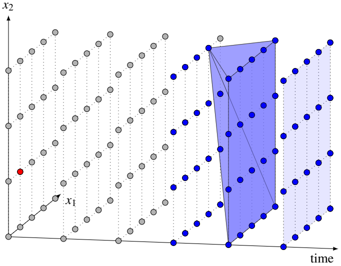
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



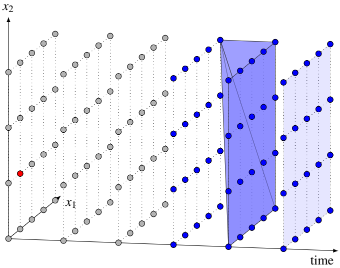
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



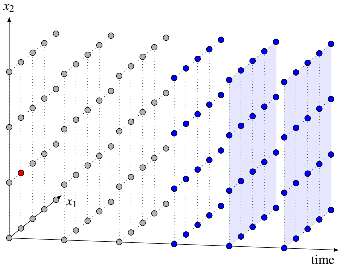
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}} \right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\xi_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\xi_t}\left[\min_{u_t}\left\{\underbrace{-p_t u_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - u_t + \xi_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$

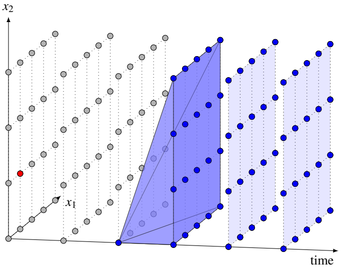
for $t : T-1 \to 0$ **do**

  for $s \in S$ **do**

    for $\xi \in \Xi$ **do**

      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$

      $V_t(s) \mathrel{+}= \mathbb{P}(\xi_t = \xi)\hat{v}$



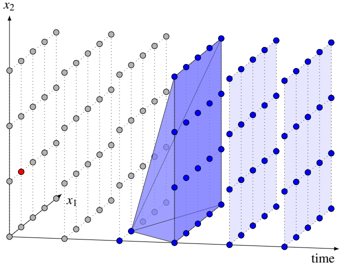
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t \boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}} \right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u \in \mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



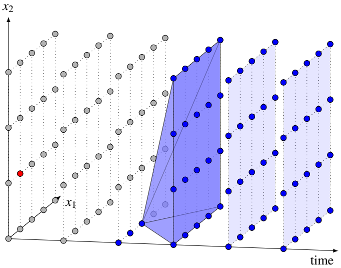
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



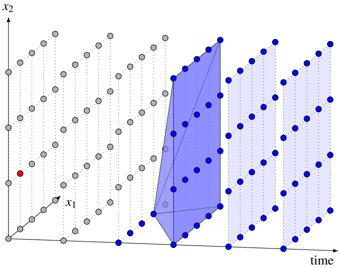
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



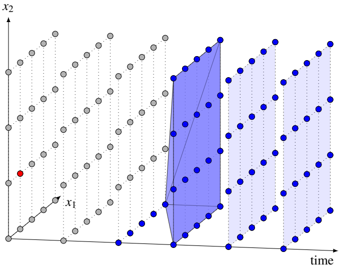
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



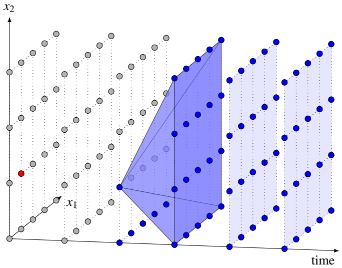
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$

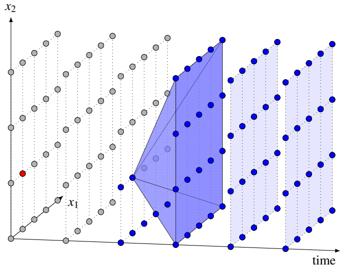
for $t : T-1 \to 0$ **do**

  for $s \in S$ **do**

    for $\xi \in \Xi$ **do**

      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$

      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



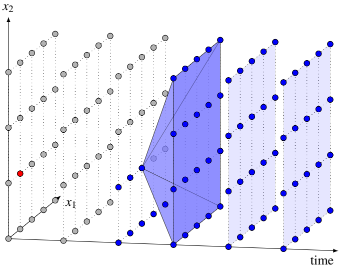
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



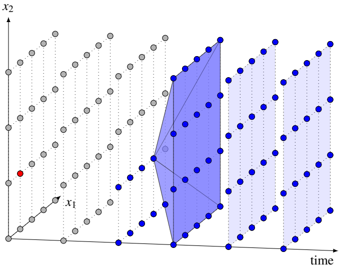
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



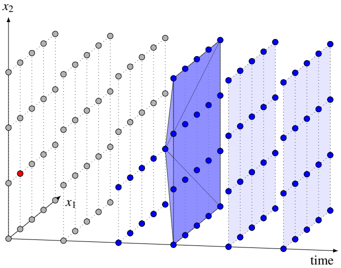
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



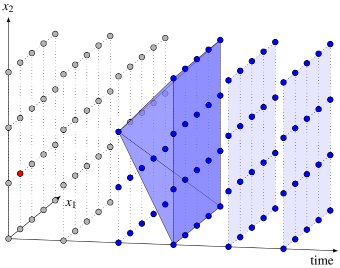
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



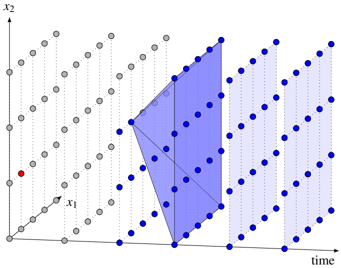
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$

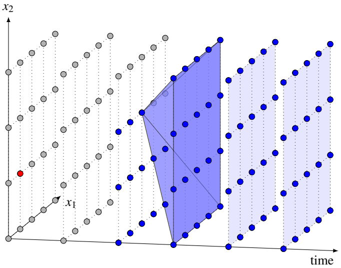
for $t : T-1 \to 0$ **do**

  for $s \in S$ **do**

    for $\xi \in \Xi$ **do**

      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$

      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



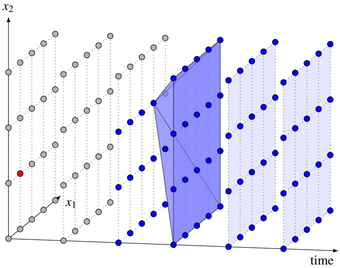
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



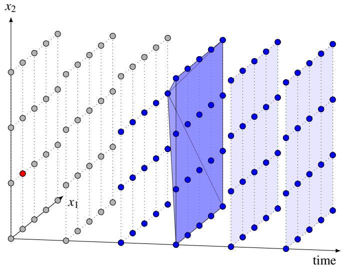
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



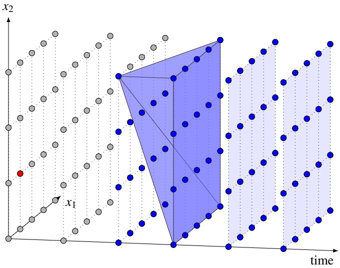
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

```
V_T ≡ K ; V_t ≡ 0
for t : T − 1 → 0 do
    for s ∈ S do
        for ξ ∈ Ξ do
            v̂ = min_{u∈U} −p_t u + V_{t+1}(s − u + ξ)
            V_t(s) += P(ξ_t = ξ) v̂
```



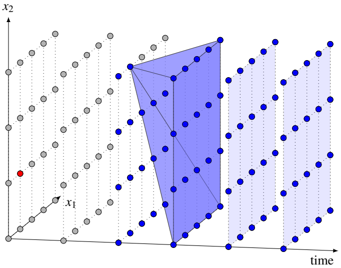
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



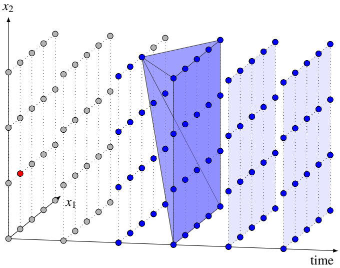
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



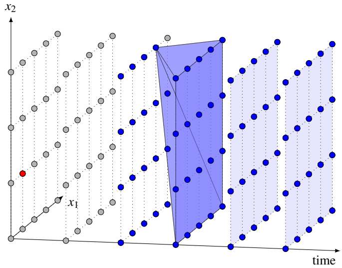
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$

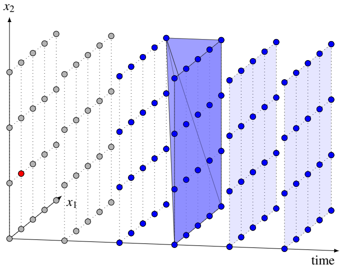
for $t : T-1 \to 0$ **do**

  for $s \in S$ **do**

    for $\xi \in \Xi$ **do**

      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$

      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



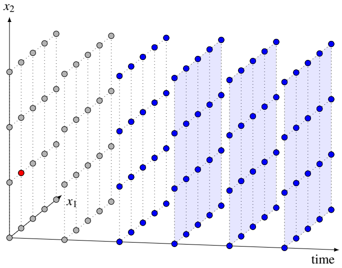
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t \boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}} \right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$

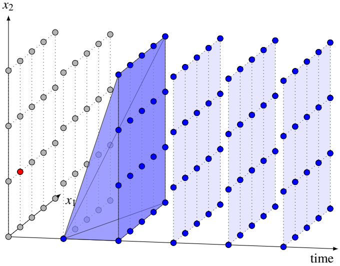
for $t : T-1 \to 0$ **do**

  for $s \in S$ **do**

    for $\xi \in \Xi$ **do**

      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$

      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



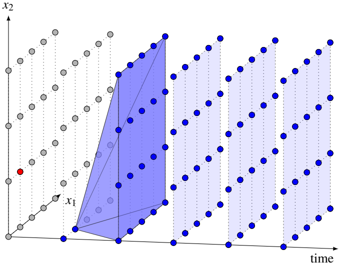
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



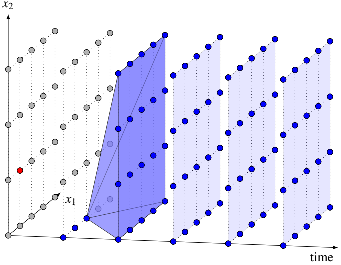
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$

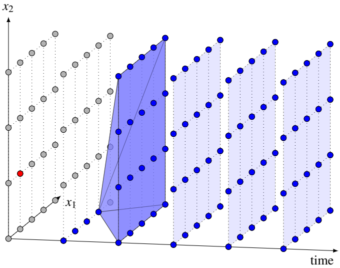
for $t : T-1 \to 0$ **do**

  for $s \in S$ **do**

    for $\xi \in \Xi$ **do**

      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$

      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



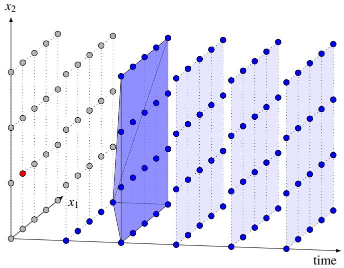
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min\limits_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



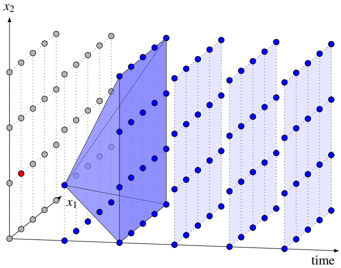
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



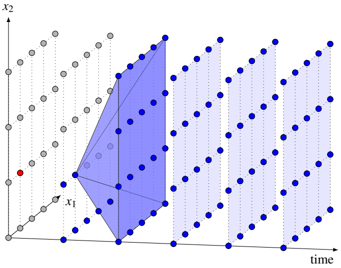
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\xi_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\xi_t}\left[\min_{u_t}\left\{\underbrace{-p_t u_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - u_t + \xi_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\xi_t = \xi)\hat{v}$



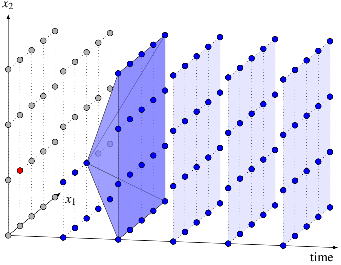
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\Big\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



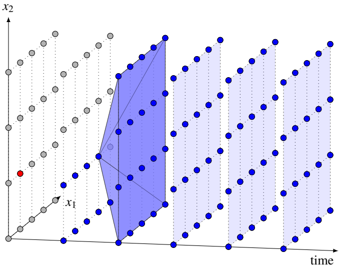
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



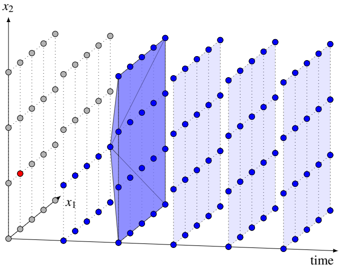
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}} \right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



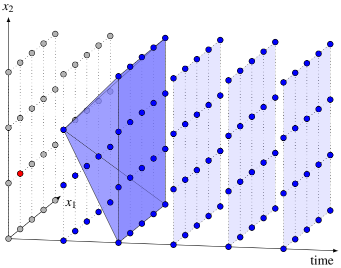
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



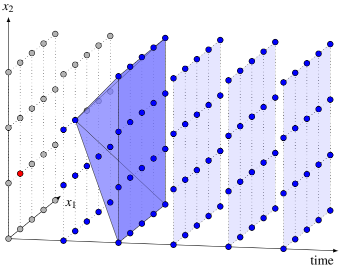
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\xi_t)_{t \in [T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\xi_t}\left[\min_{\boldsymbol{u}_t} \left\{ \underbrace{-\boldsymbol{p}_t \boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \xi_t)}_{\text{cost-to-go}} \right\} \right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u \in \mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\xi_t = \xi)\hat{v}$



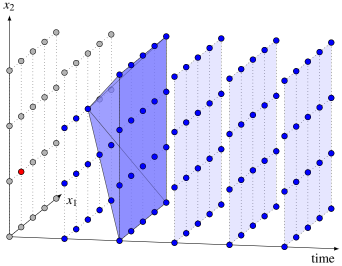
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



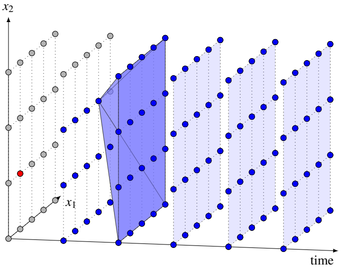
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



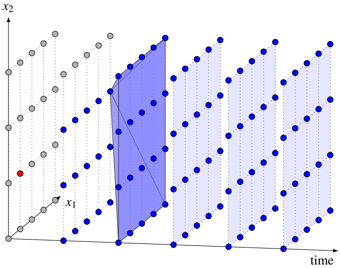
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



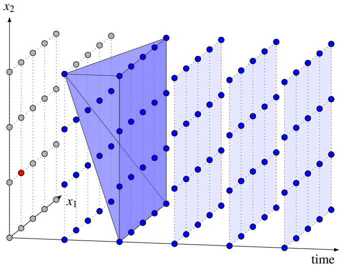
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t \boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}} \right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$

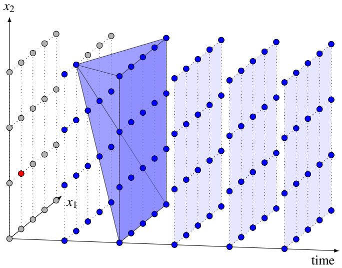
for $t : T-1 \to 0$ **do**

  for $s \in S$ **do**

    for $\xi \in \Xi$ **do**

      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$

      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



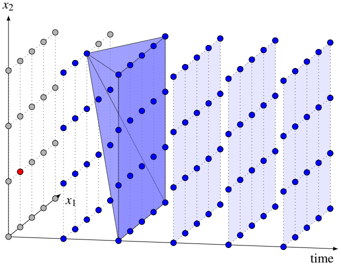
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



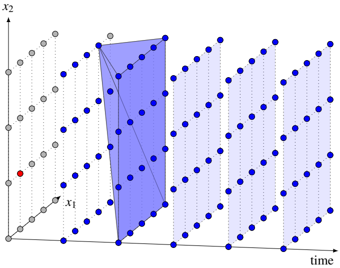
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{ \underbrace{-\boldsymbol{p}_t \boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - \boldsymbol{u}_t + \boldsymbol{\xi}_t)}_{\text{cost-to-go}} \right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u \in \mathcal{U}} -p_t u + V_{t+1}(s - u + \xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



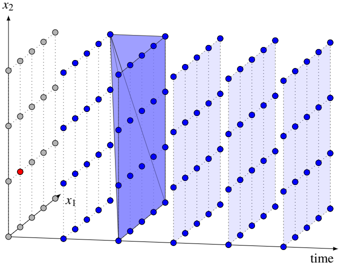
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\left[\min_{\boldsymbol{u}_t}\left\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\right\}\right]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



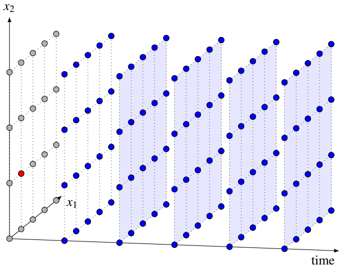
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\boldsymbol{\xi}_t}\Big[\min_{\boldsymbol{u}_t}\Big\{\underbrace{-\boldsymbol{p}_t\boldsymbol{u}_t}_{\text{current cost}} + \underbrace{V_{t+1}(s-\boldsymbol{u}_t+\boldsymbol{\xi}_t)}_{\text{cost-to-go}}\Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
    for $s \in S$ **do**
        for $\xi \in \Xi$ **do**
            $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
            $V_t(s) \mathrel{+}= \mathbb{P}(\boldsymbol{\xi}_t = \xi)\hat{v}$



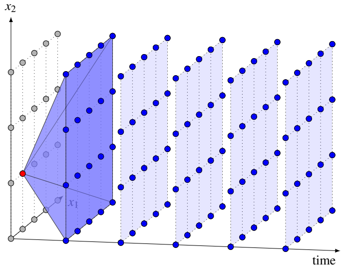
Dynamic Programming

Under a crucial stagewise independence assumption (*i.e.* $(\xi_t)_{t\in[T]}$ is a sequence of independent random variables), we have the Dynamic Programming equation

$$V_t(s) = \mathbb{E}_{\xi_t}\Big[\min_{u_t}\Big\{ \underbrace{-p_t u_t}_{\text{current cost}} + \underbrace{V_{t+1}(s - u_t + \xi_t)}_{\text{cost-to-go}} \Big\}\Big]$$

Algorithm 1: Discretized Stochastic Dynamic Programming

$V_T \equiv K$; $V_t \equiv 0$
for $t : T-1 \to 0$ **do**
  for $s \in S$ **do**
    for $\xi \in \Xi$ **do**
      $\hat{v} = \min_{u\in\mathcal{U}} -p_t u + V_{t+1}(s-u+\xi)$
      $V_t(s) \mathrel{+}= \mathbb{P}(\xi_t = \xi)\hat{v}$

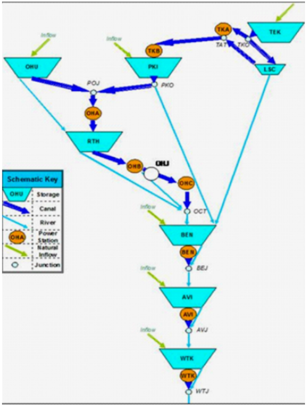


From Dynamic Programming to SDDP

- DP is a flexible tool, hampered by the curses of dimensionality
- Numerical illustration (7 dams):
  - $T = 52$ weeks
  - $|S| = 100^7$ possible states
  - $|U| = 10^7$ possible controls
  - $|\xi_t| = 10$ (10^{52} scenarios)

➥ $\approx$ 2 days on today's fastest super-computer (3.10^6 years for 10 dams)

➥ Can be solved[1] in $\approx$ 1 minute ($\approx$ 3 minutes for 10 dams)



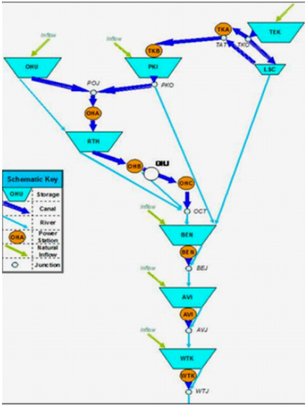
[1] Approximately, depending on the problem and precision required...

From Dynamic Programming to SDDP

- DP is a flexible tool, hampered by the curses of dimensionality
- Numerical illustration (7 dams):
  - $T = 52$ weeks
  - $|S| = 100^7$ possible states
  - $|U| = 10^7$ possible controls
  - $|\xi_t| = 10$ (10^{52} scenarios)

➥ $\approx$ 2 days on today's fastest super-computer (3.10^6 years for 10 dams)

➥ Can be solved[1] in $\approx$ 1 minute ($\approx$ 3 minutes for 10 dams)



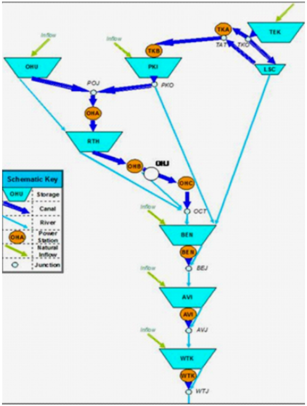
[1] Approximately, depending on the problem and precision required...

From Dynamic Programming to SDDP

- DP is a flexible tool, hampered by the curses of dimensionality
- Numerical illustration (7 dams):
  - $T = 52$ weeks
  - $|S| = 100^7$ possible states
  - $|U| = 10^7$ possible controls
  - $|\xi_t| = 10$ (10^{52} scenarios)

➥ $\approx$ 2 days on today's fastest super-computer (3.10^6 years for 10 dams)

➥ Can be solved[1] in $\approx$ 1 minute ($\approx$ 3 minutes for 10 dams)



[1] Approximately, depending on the problem and precision required...

How can we be so much faster ?

- Structural assumptions:
  - convexity
  - continuous state
  - ➥ duality tools
- Sampling instead of exhaustive computation
- Iteratively refining value function estimation at "the right places" only

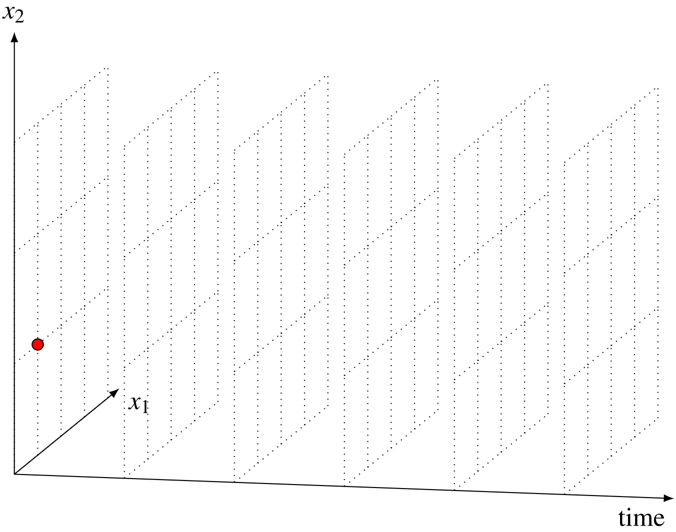
➥ Stochastic Dual Dynamic Programming (SDDP) which
  - has been around for 30 years
  - is widely used in the energy community
  - has lots of extensions and variants
  - some convergence results, mainly asymptotic

How can we be so much faster ?

- Structural assumptions:
  - convexity
  - continuous state
  - ➥ duality tools
- Sampling instead of exhaustive computation
- Iteratively refining value function estimation at "the right places" only

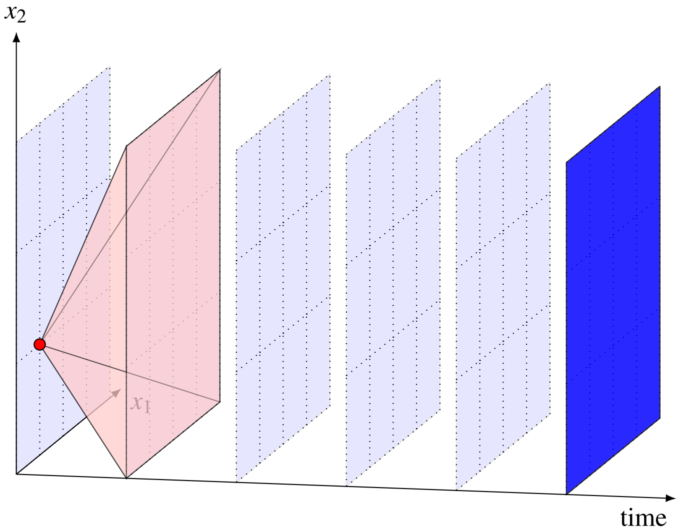
➥ Stochastic Dual Dynamic Programming (SDDP) which
  - has been around for 30 years
  - is widely used in the energy community
  - has lots of extensions and variants
  - some convergence results, mainly asymptotic

Trajectory Following Dynamic Programming



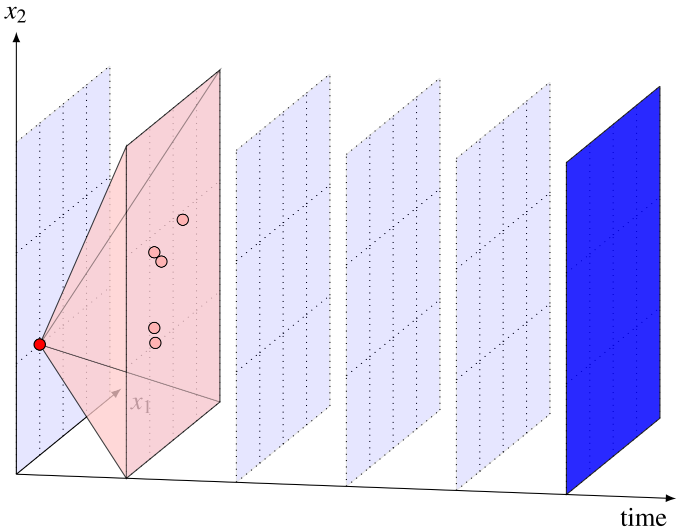
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



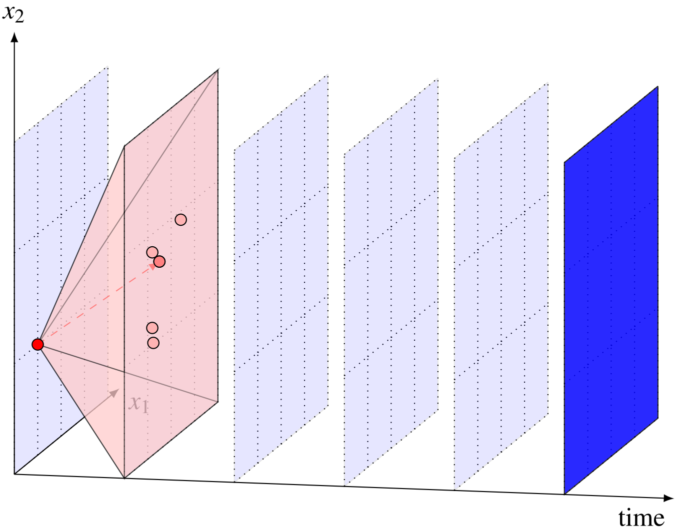
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



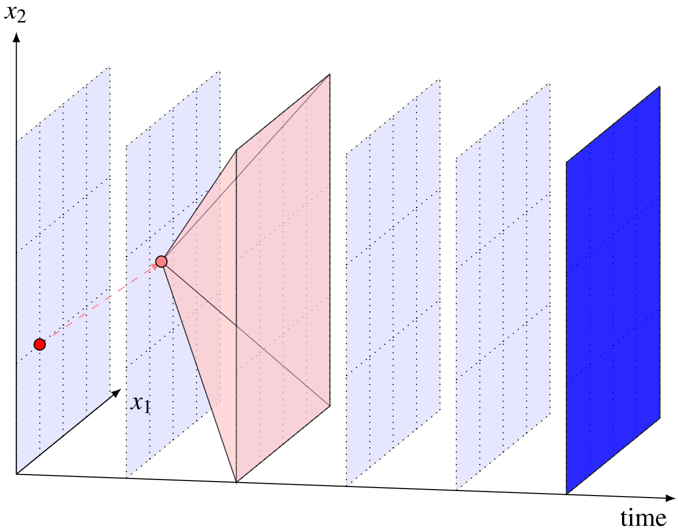
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



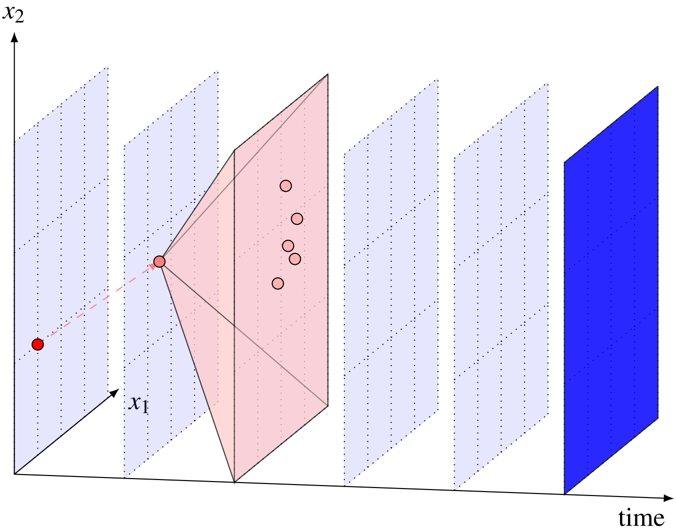
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



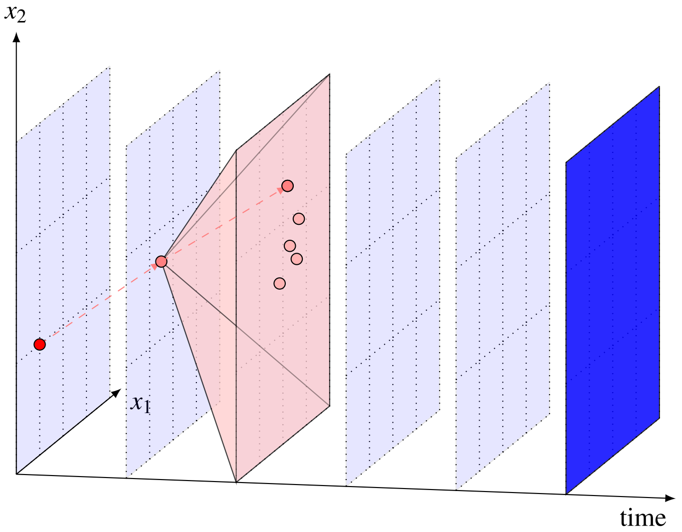
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



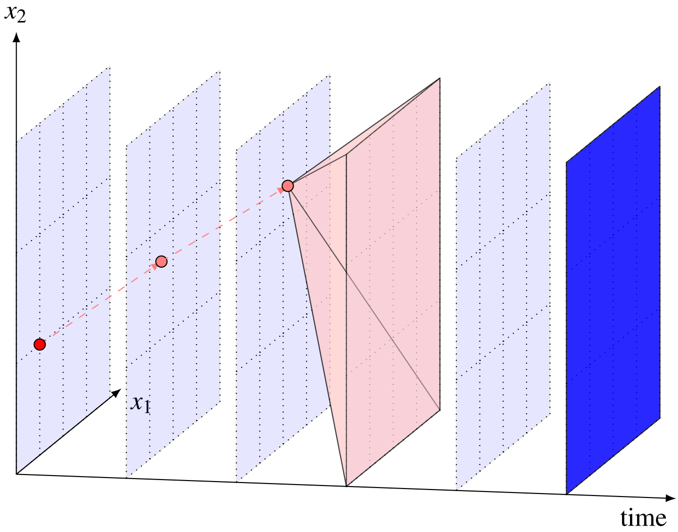
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



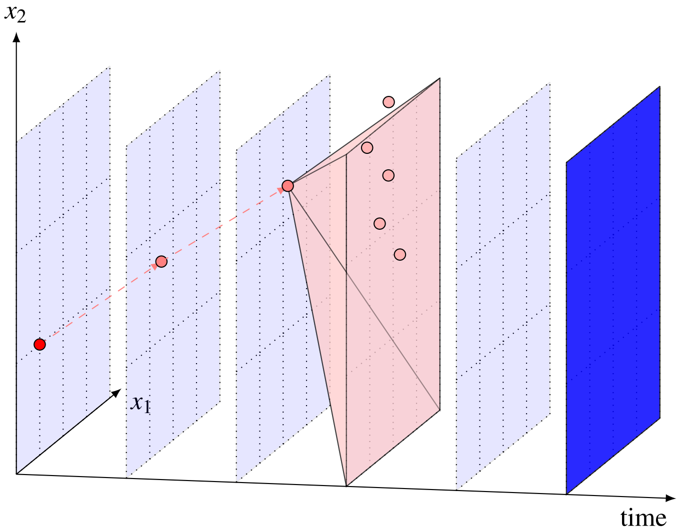
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



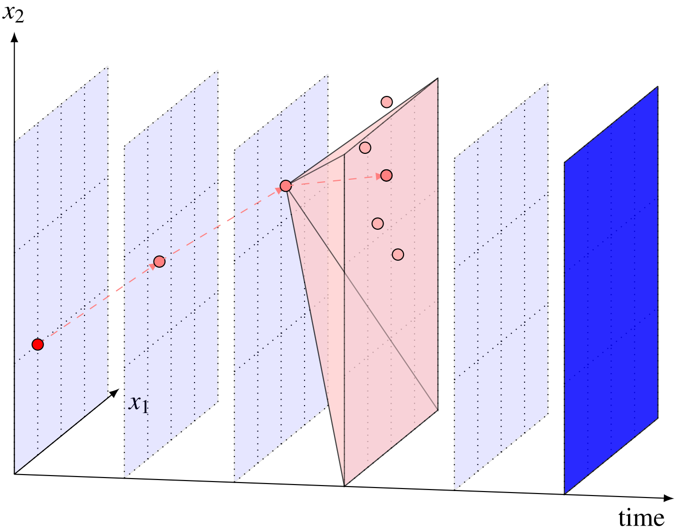
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



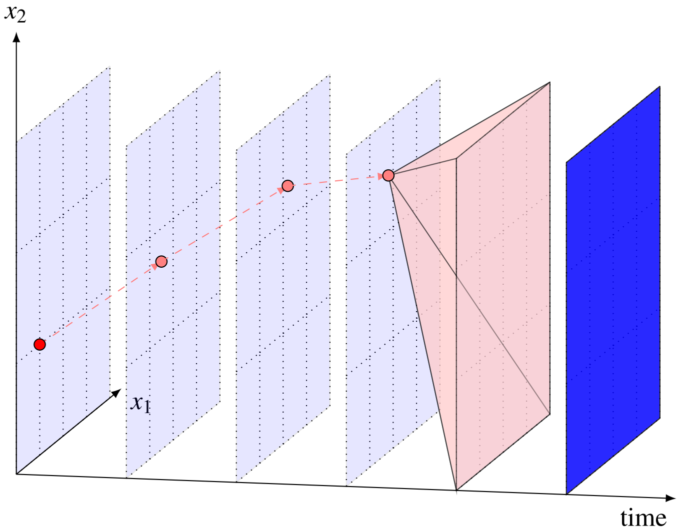
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



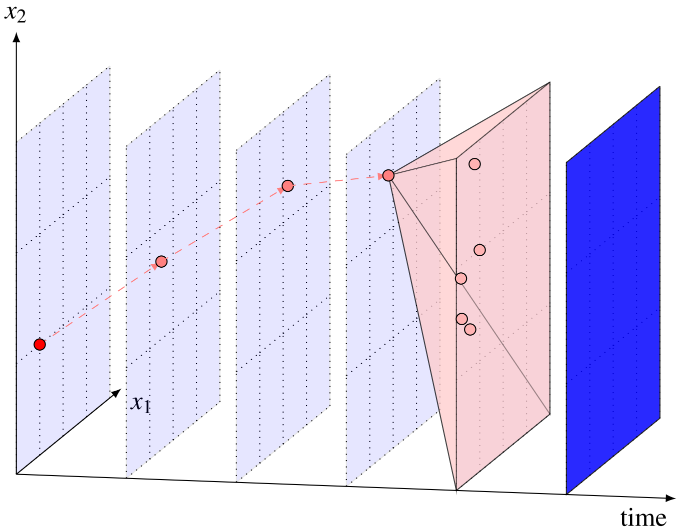
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



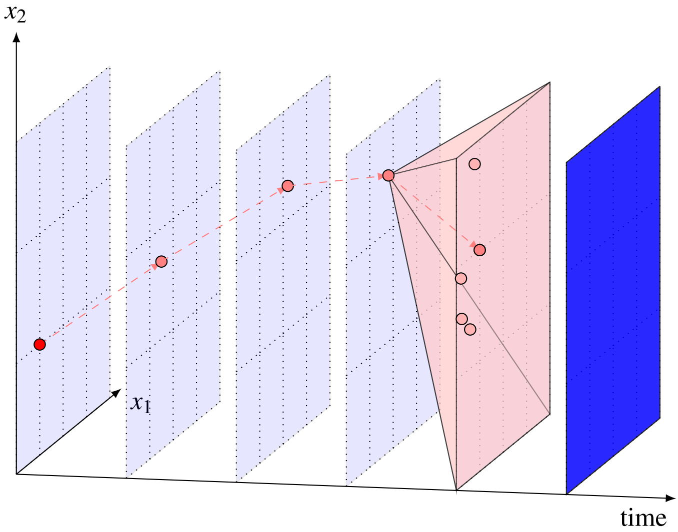
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



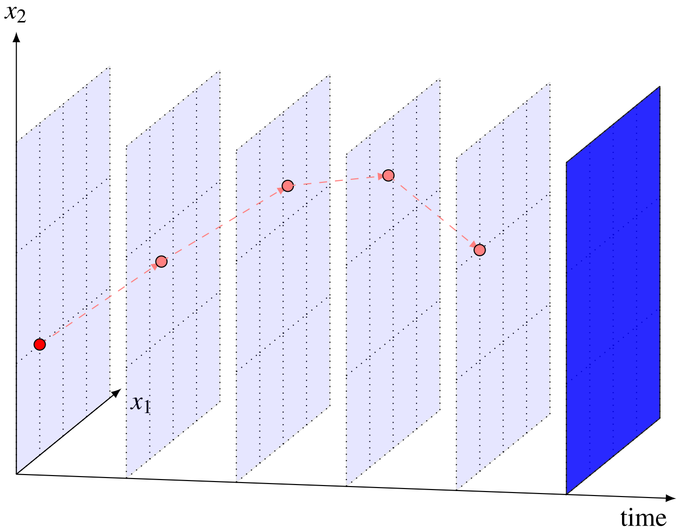
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



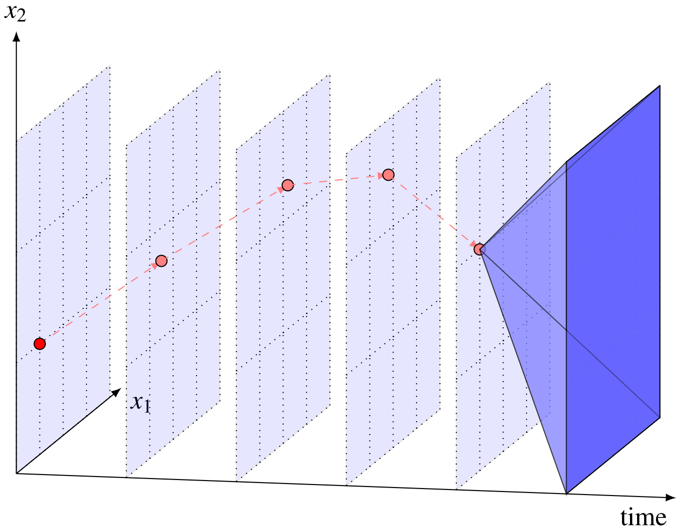
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



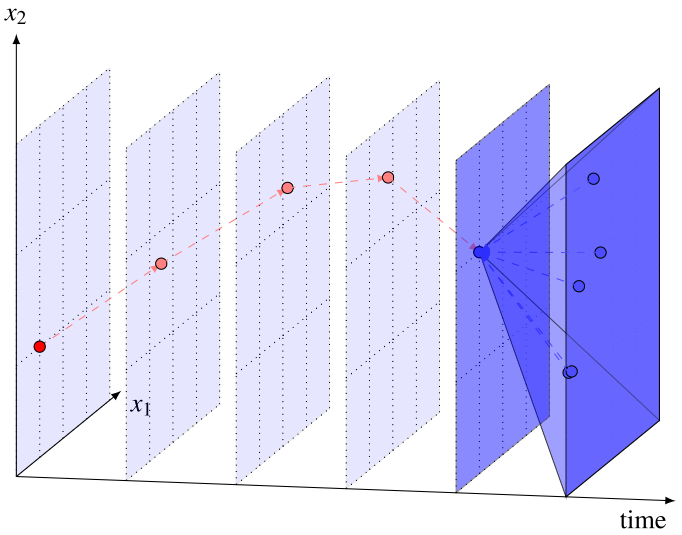
First forward pass : computing trajectory

Trajectory Following Dynamic Programming



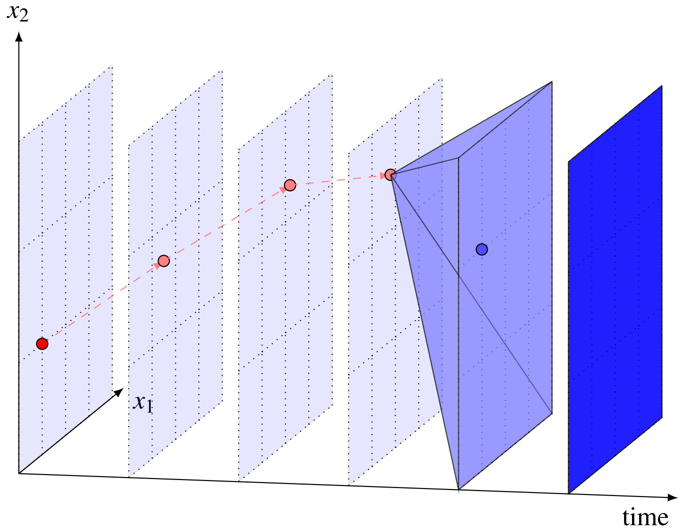
First backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



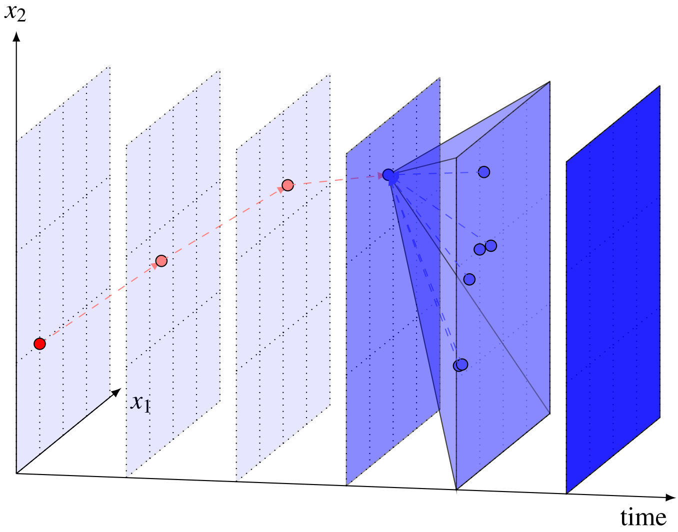
First backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



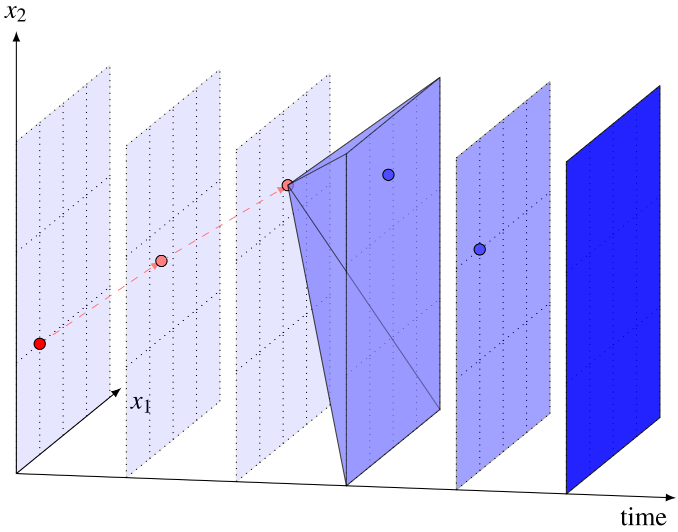
First backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



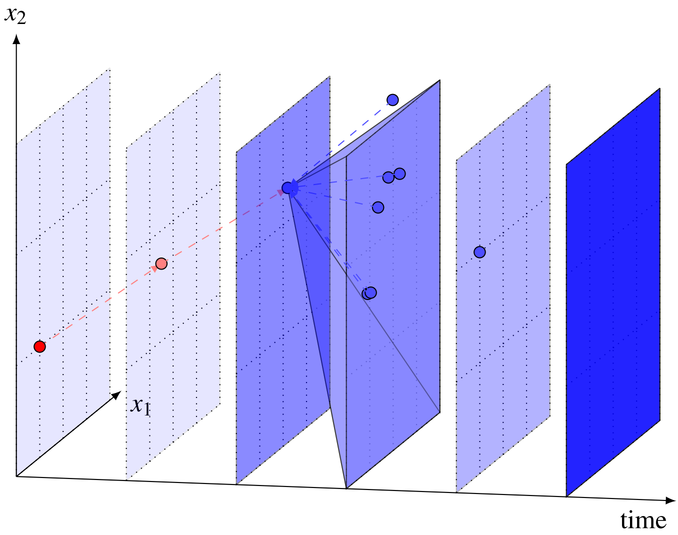
First backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



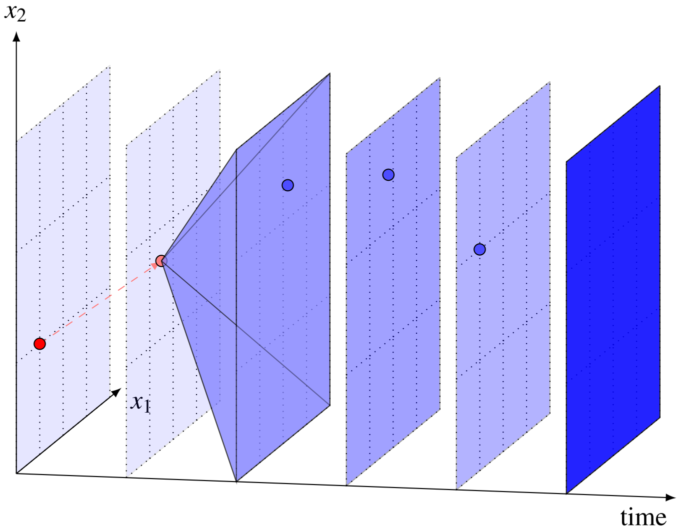
First backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



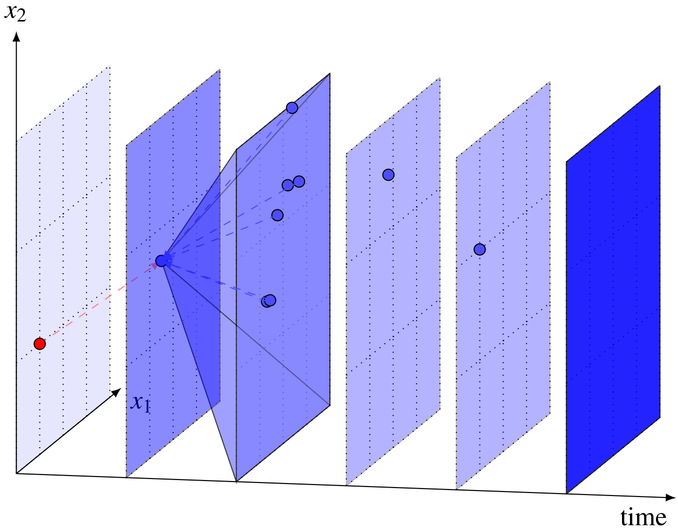
First backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



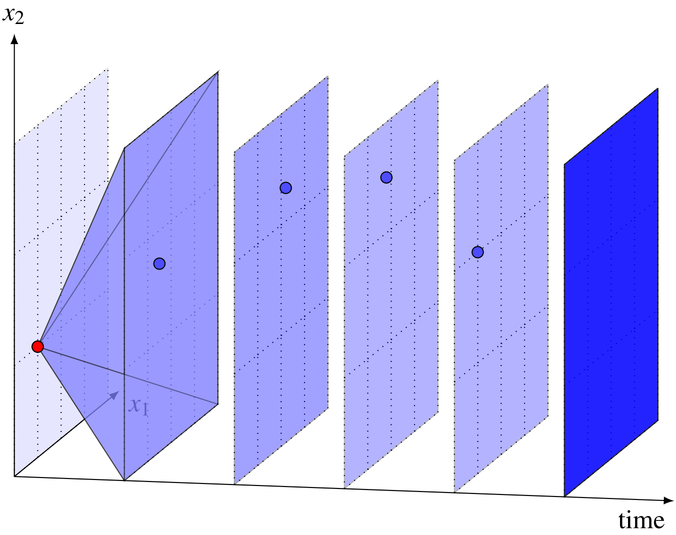
First backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



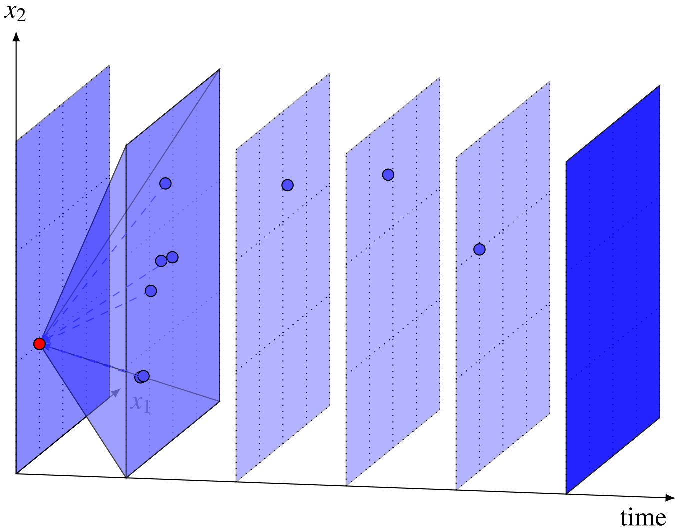
First backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



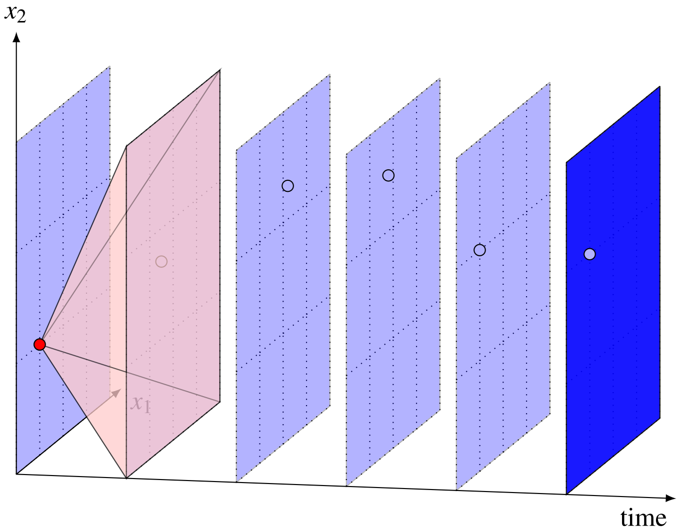
First backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



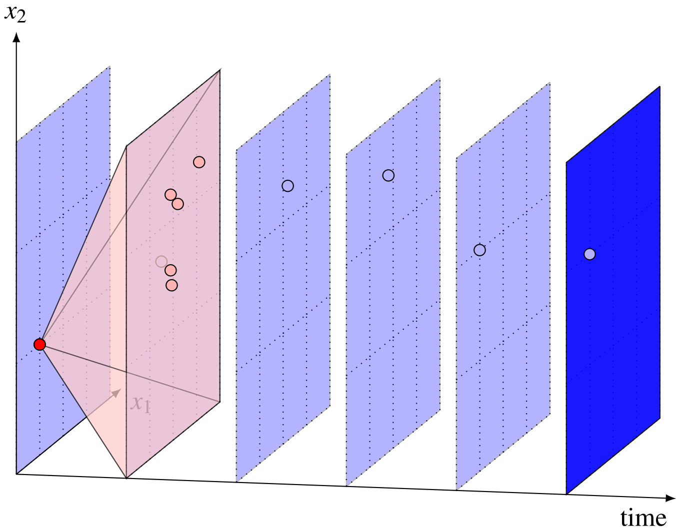
First backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



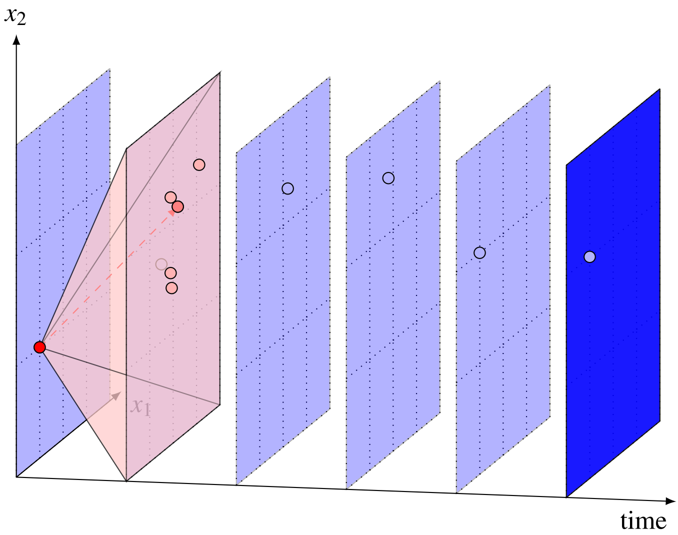
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



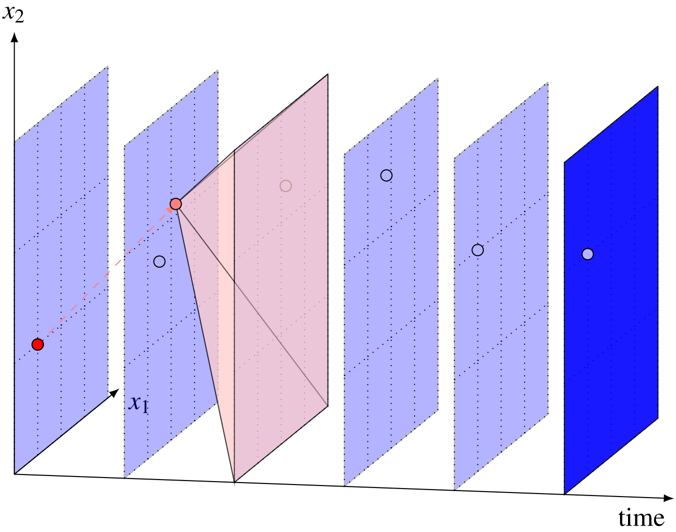
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



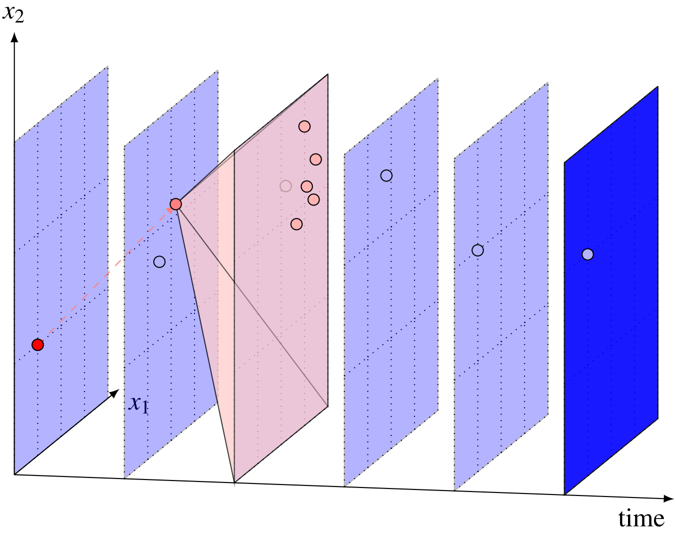
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



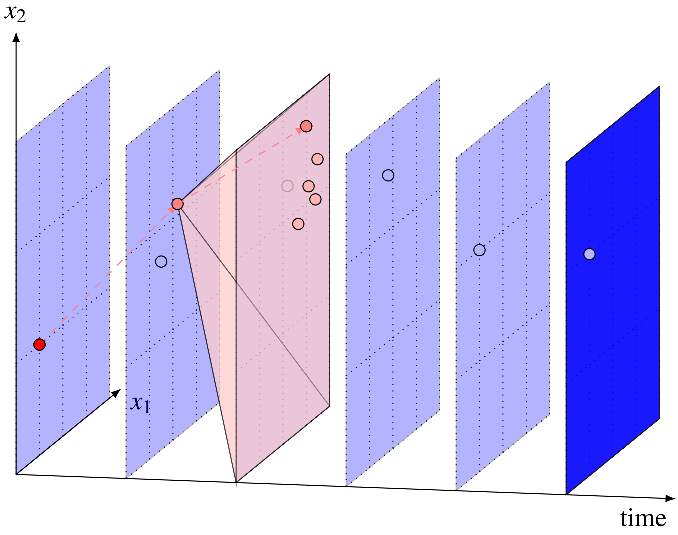
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



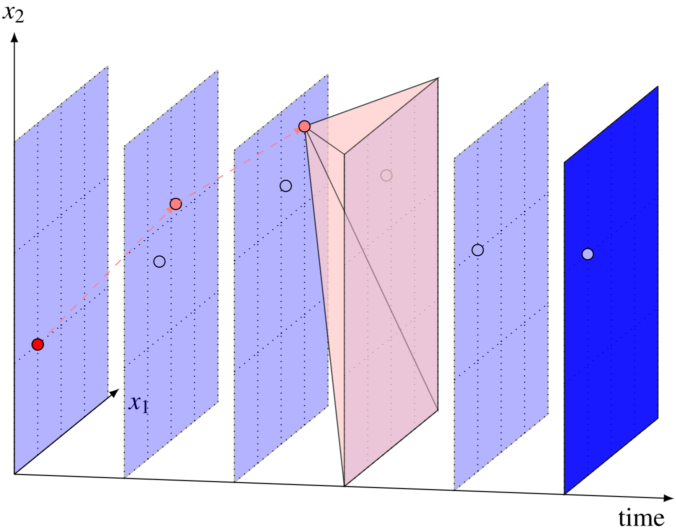
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



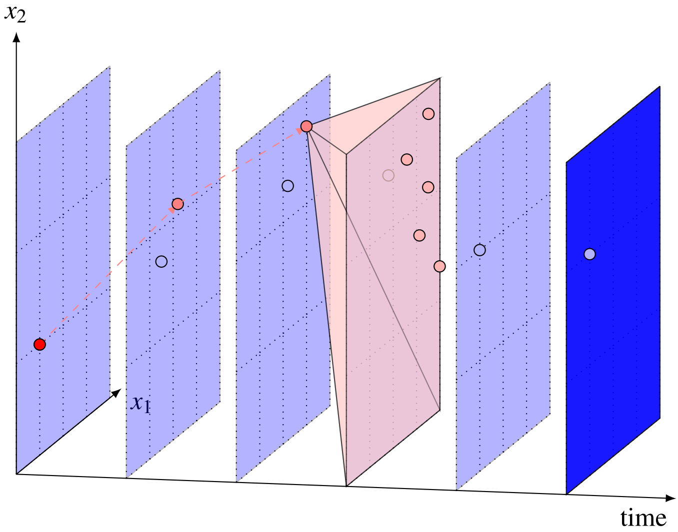
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



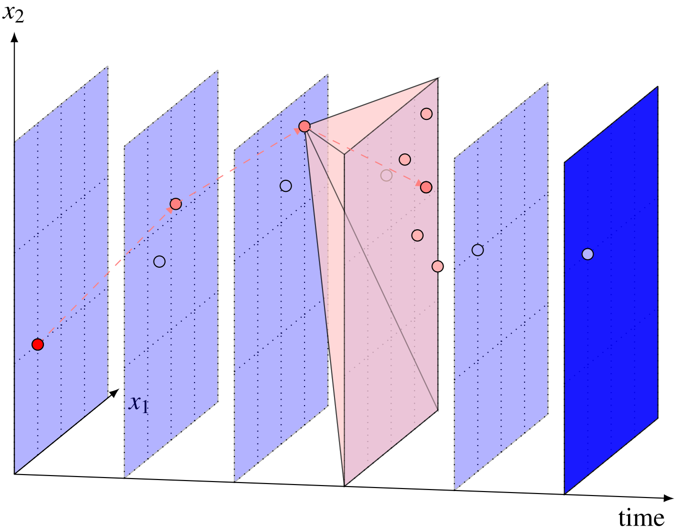
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



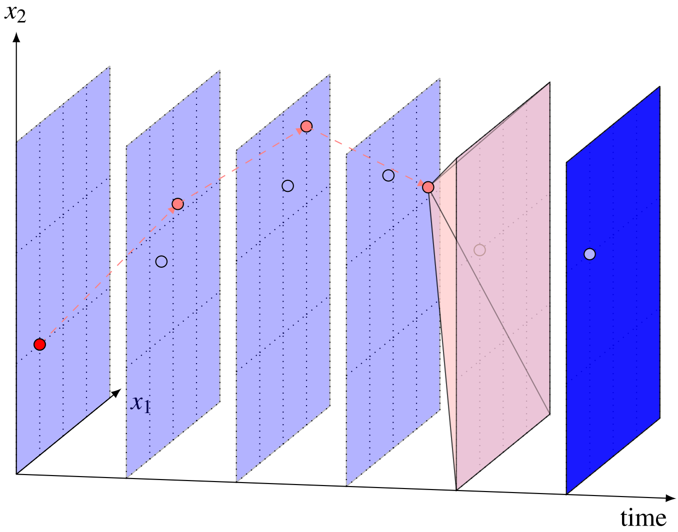
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



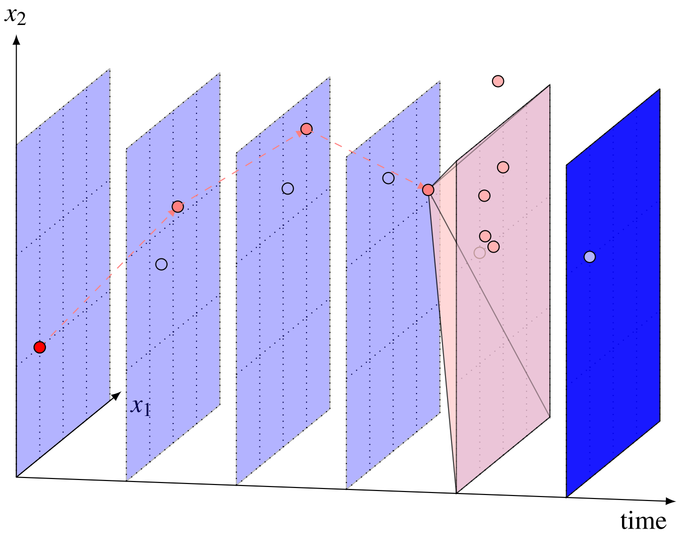
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



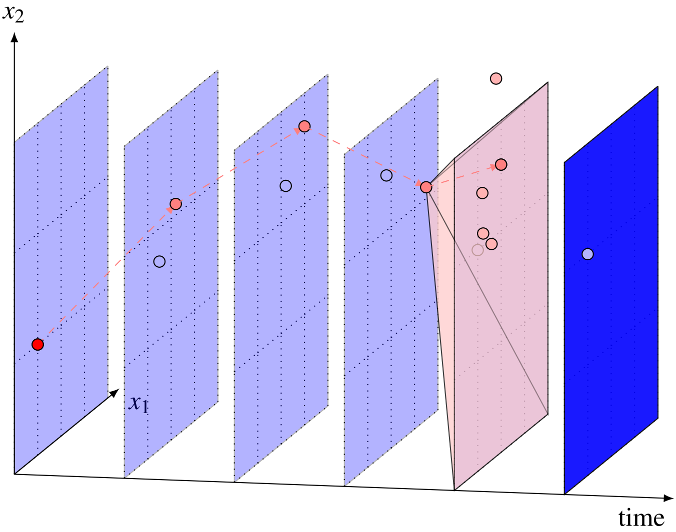
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



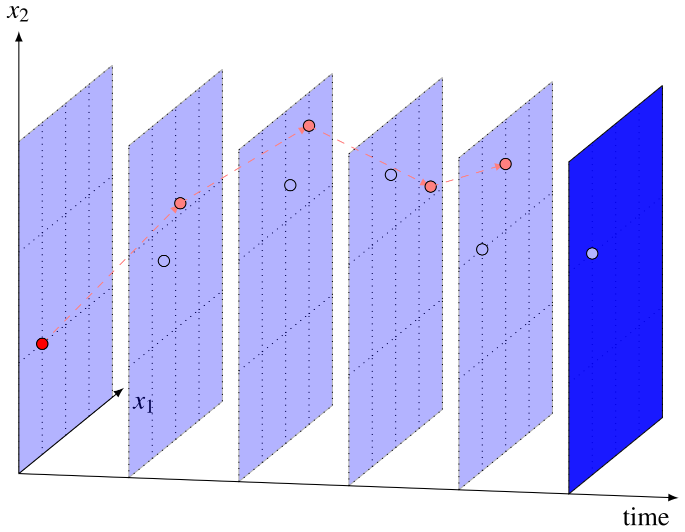
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



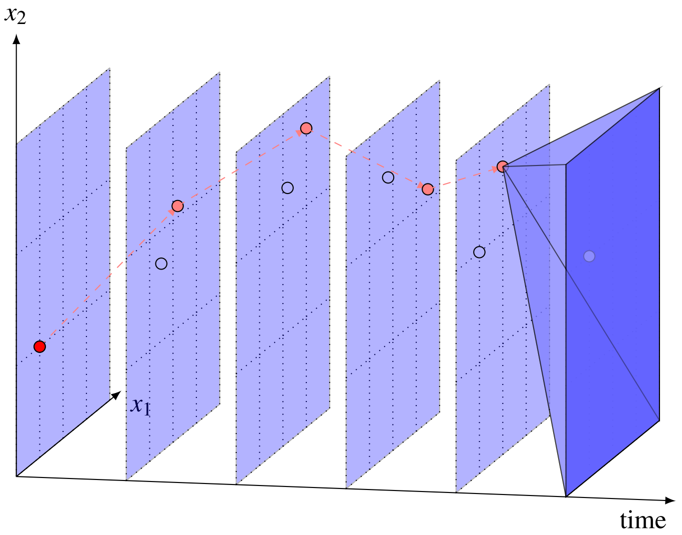
second forward pass : computing trajectory

Trajectory Following Dynamic Programming



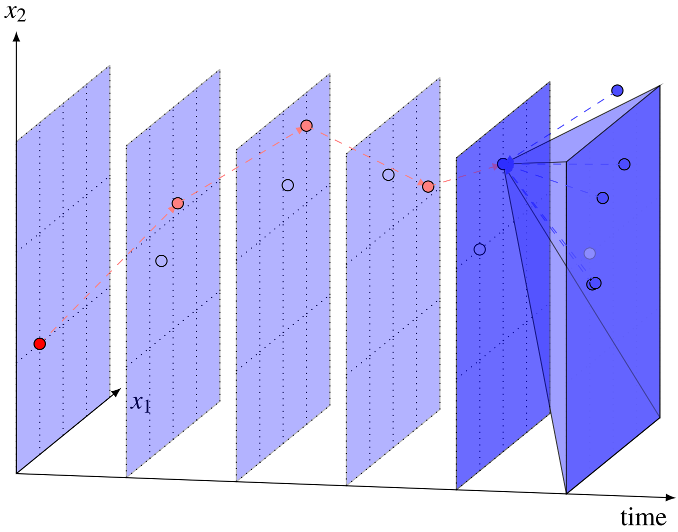
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



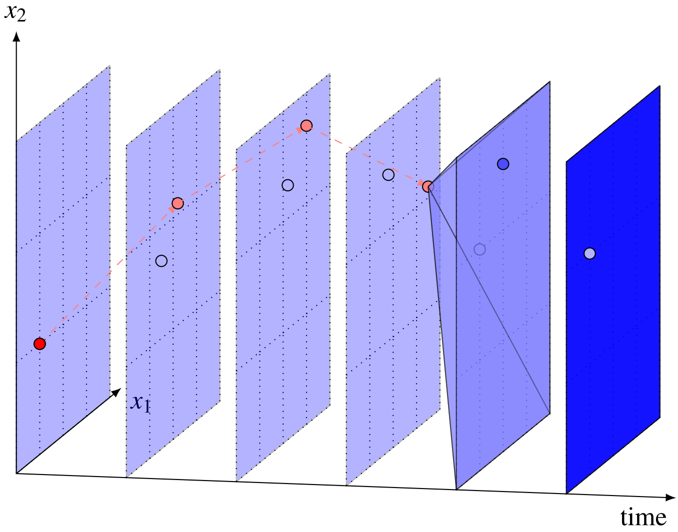
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



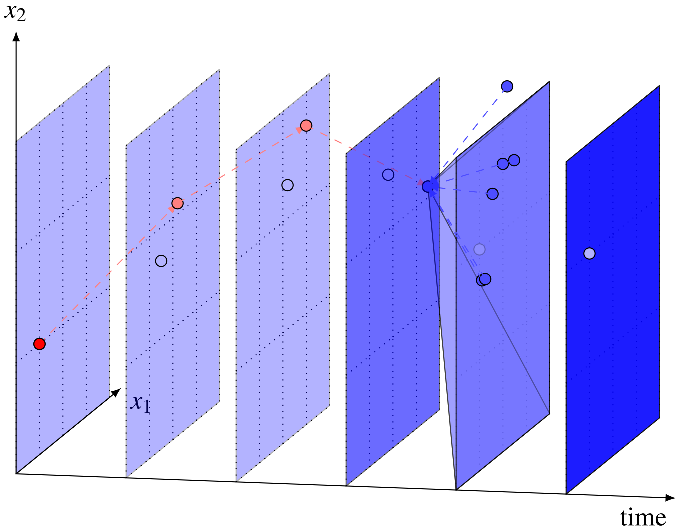
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



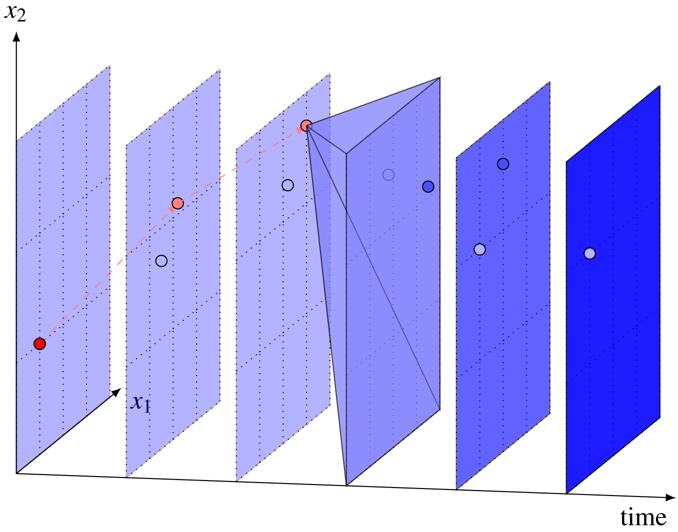
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



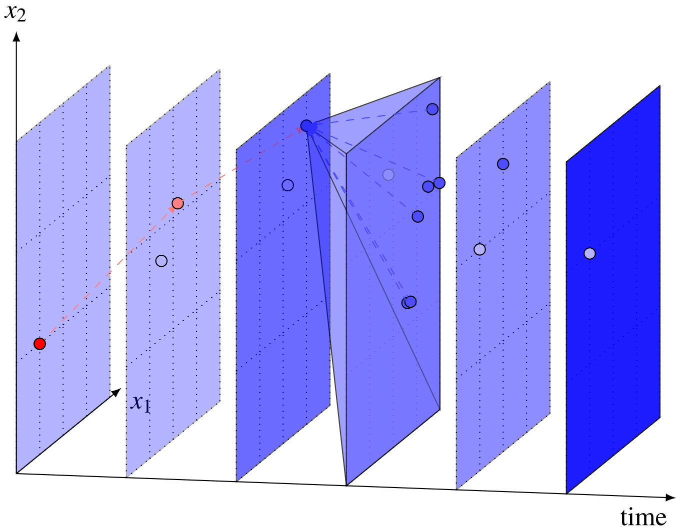
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



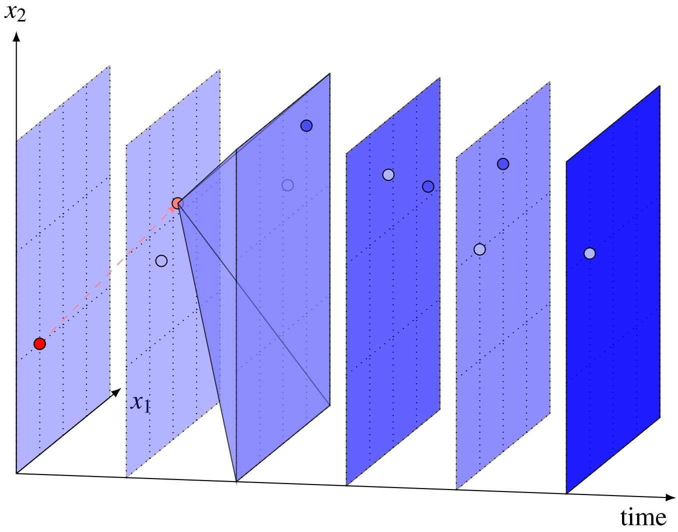
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



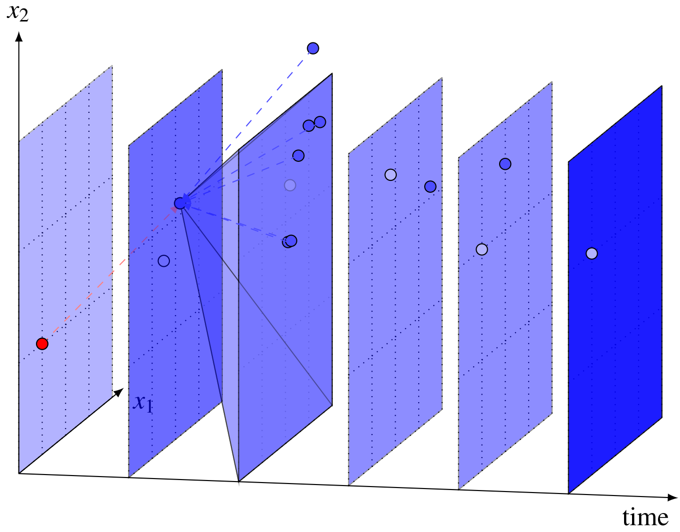
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



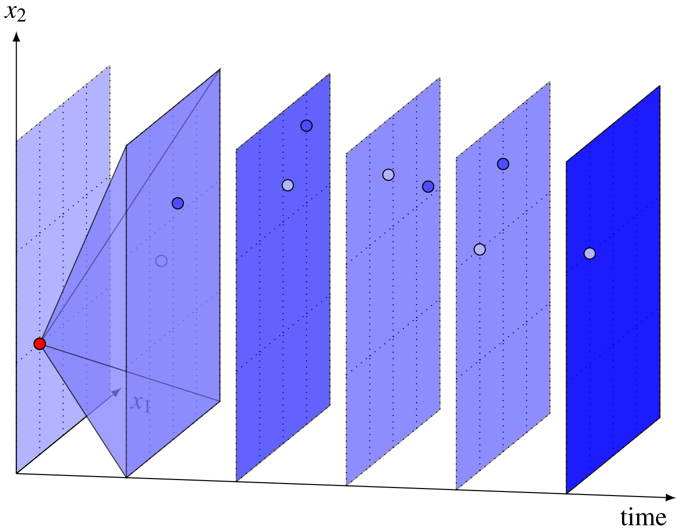
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



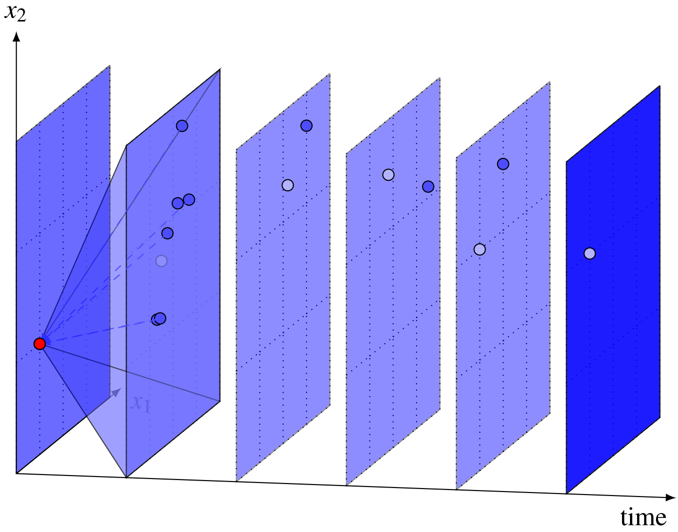
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



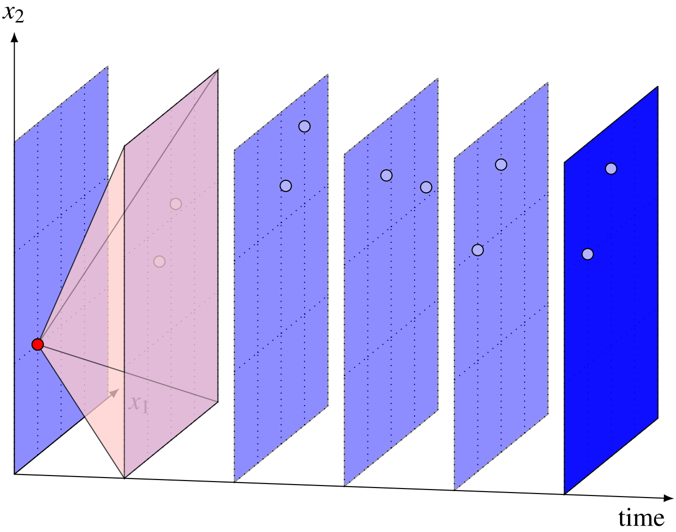
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



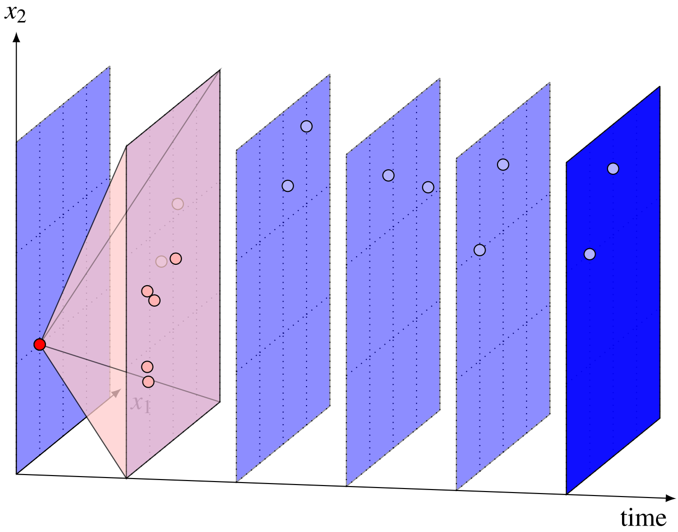
second backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



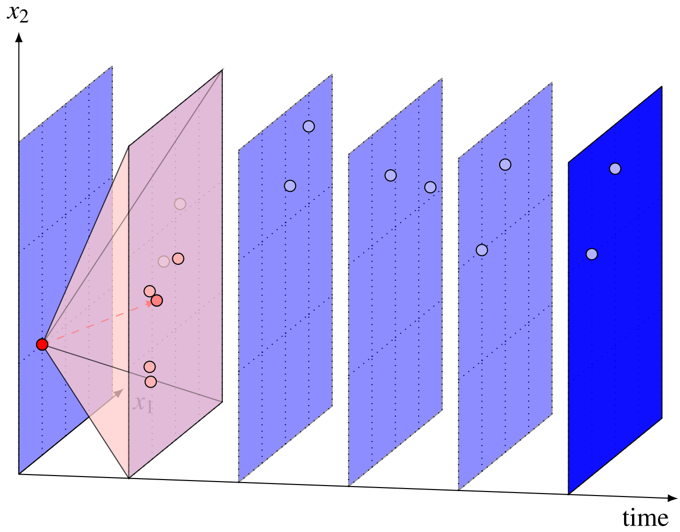
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



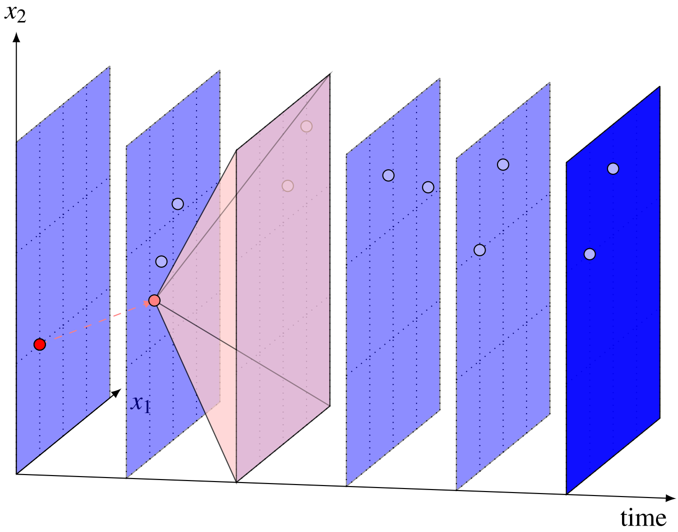
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



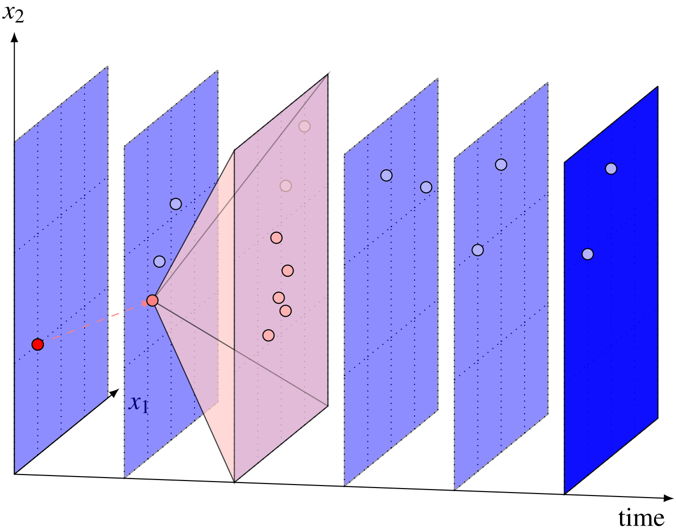
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



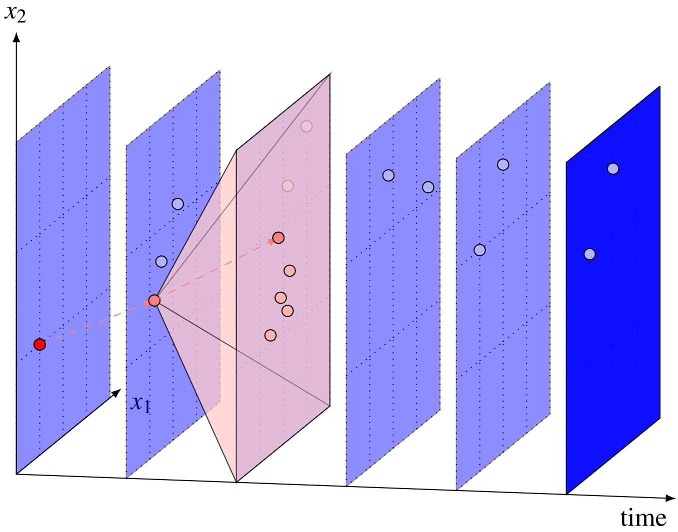
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



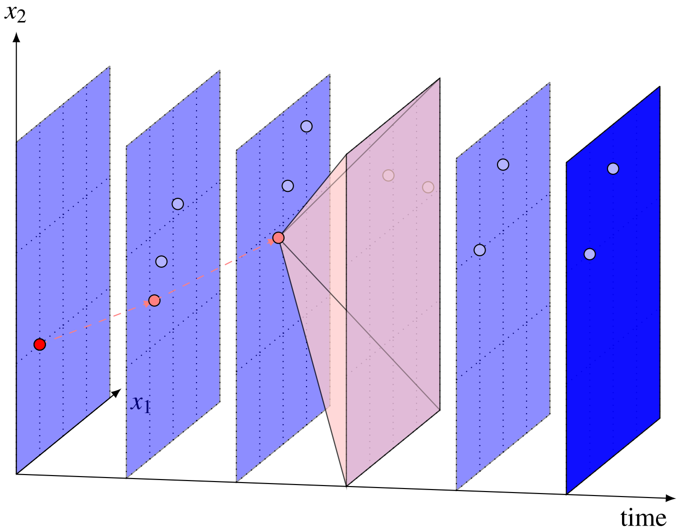
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



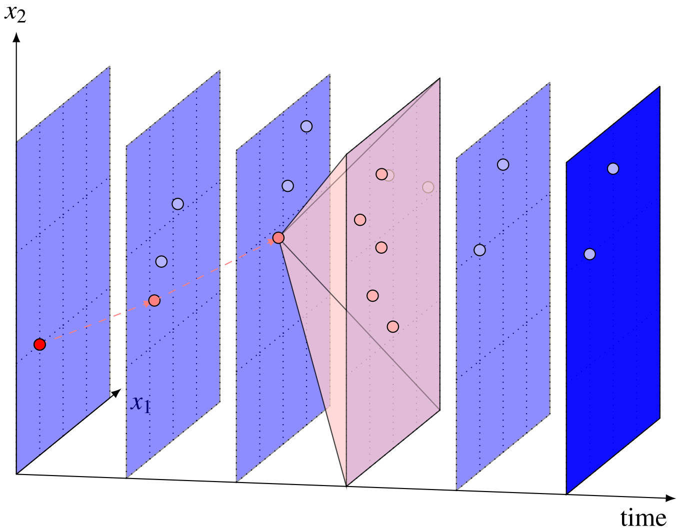
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



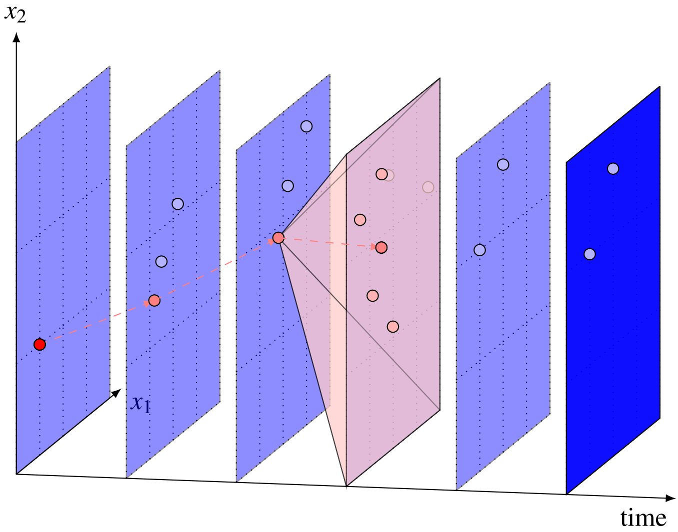
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



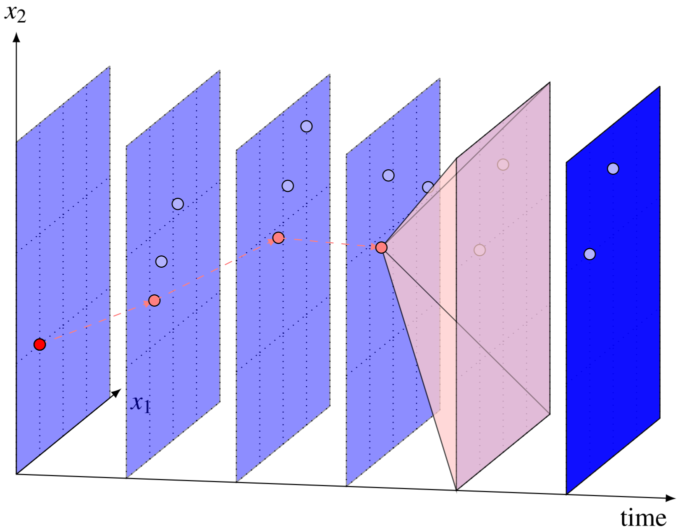
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



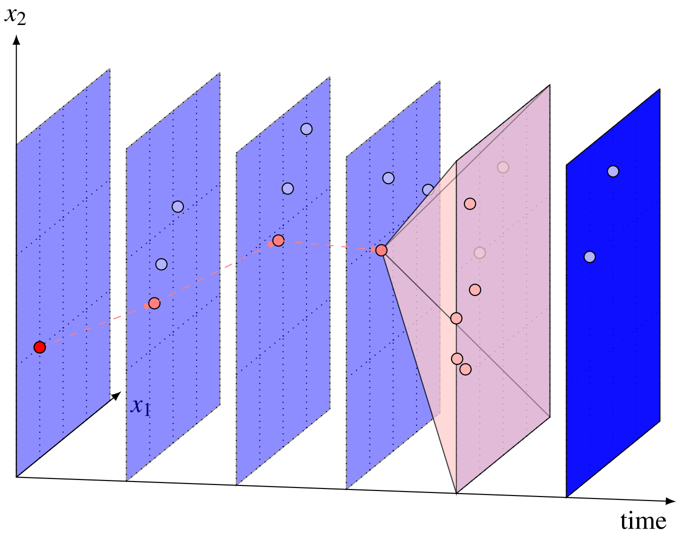
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



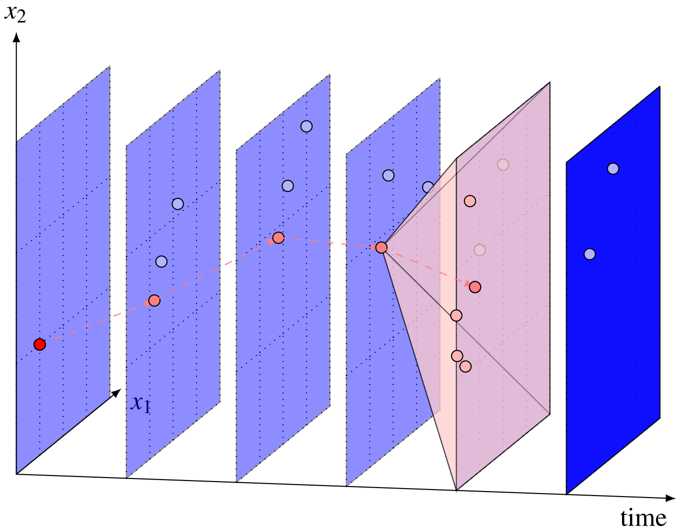
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



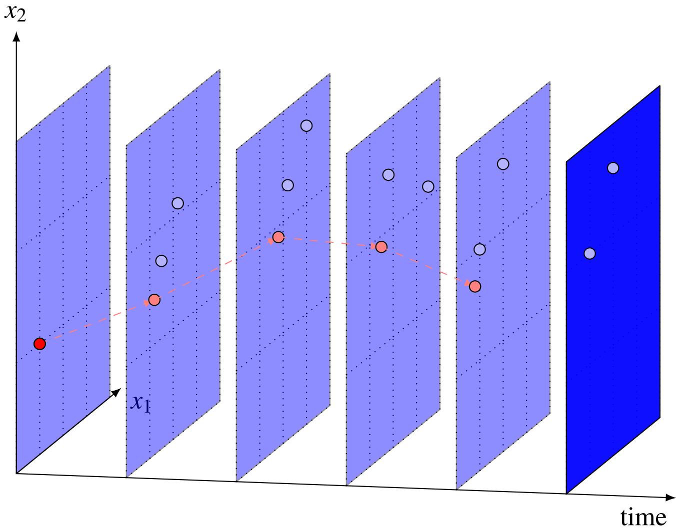
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



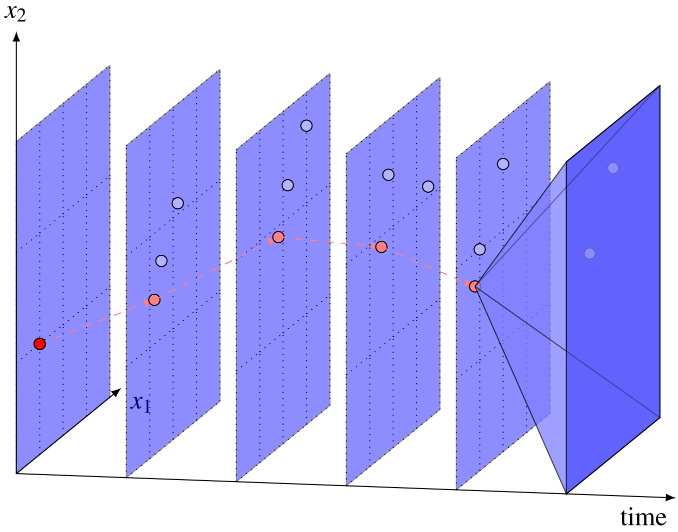
third forward pass : computing trajectory

Trajectory Following Dynamic Programming



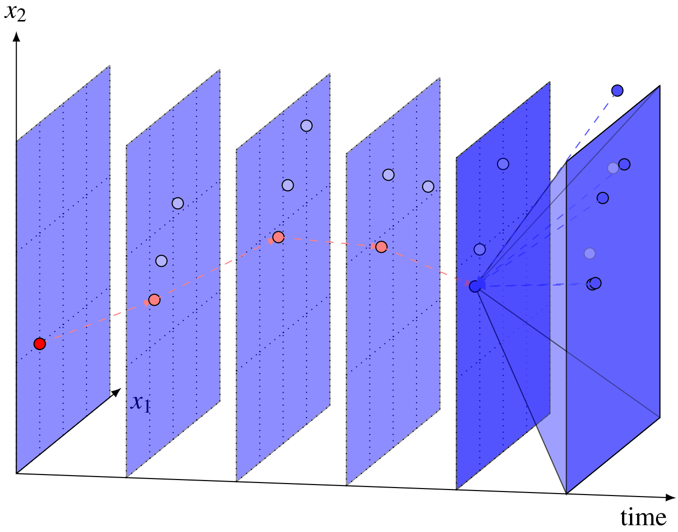
third backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



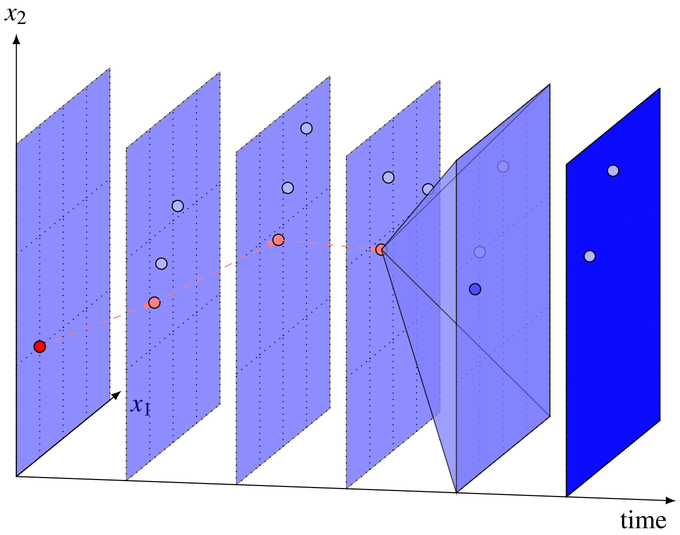
third backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



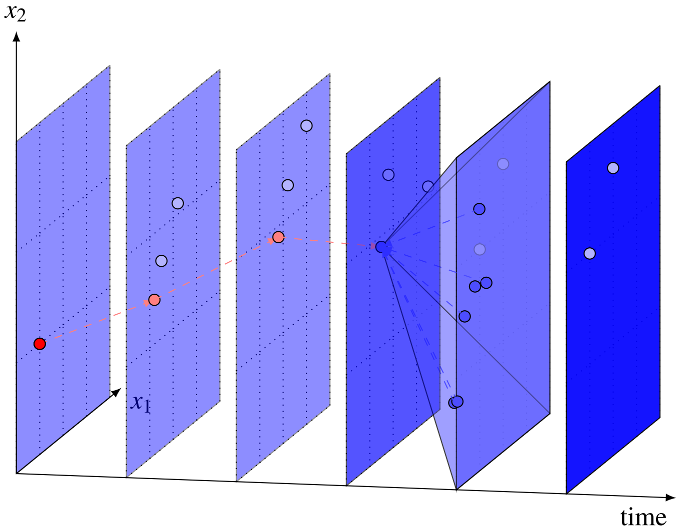
third backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



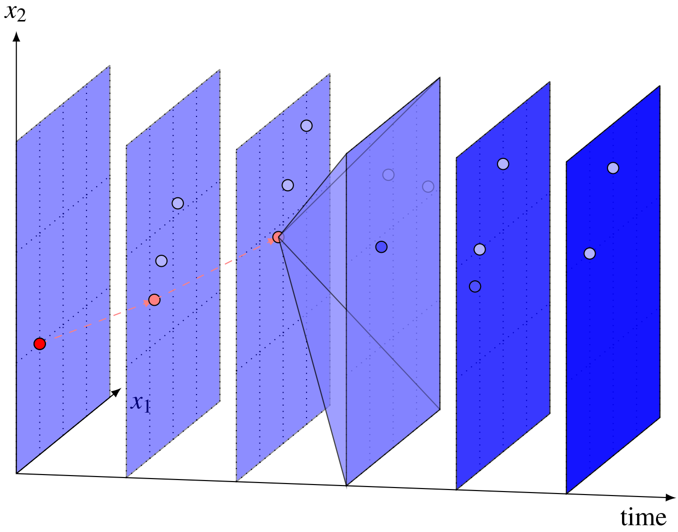
third backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



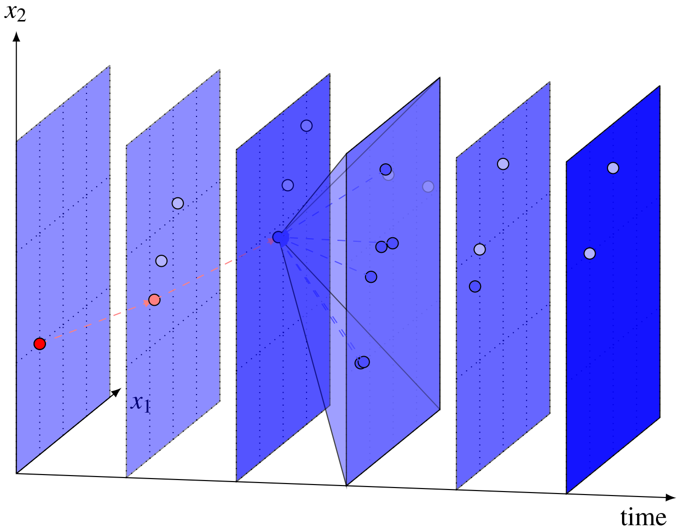
third backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



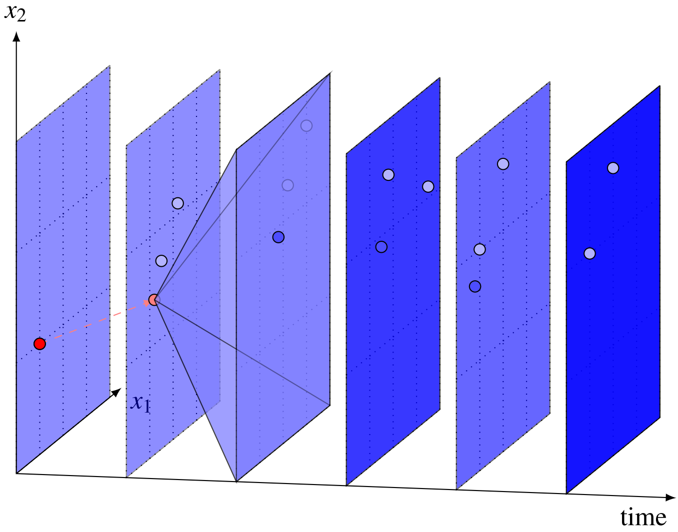
third backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



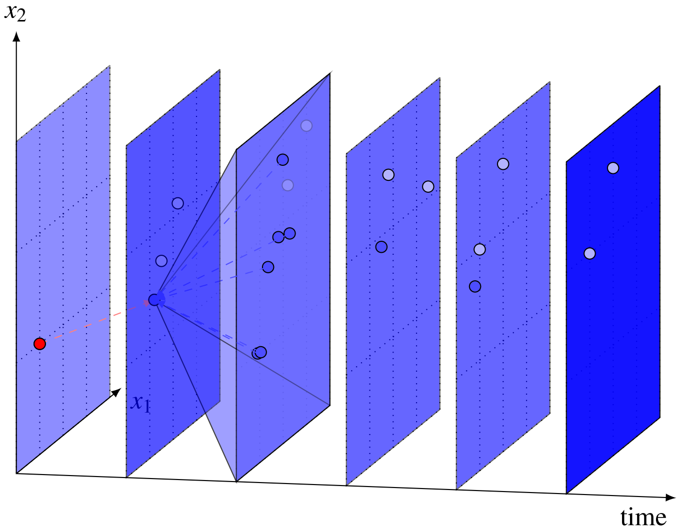
third backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



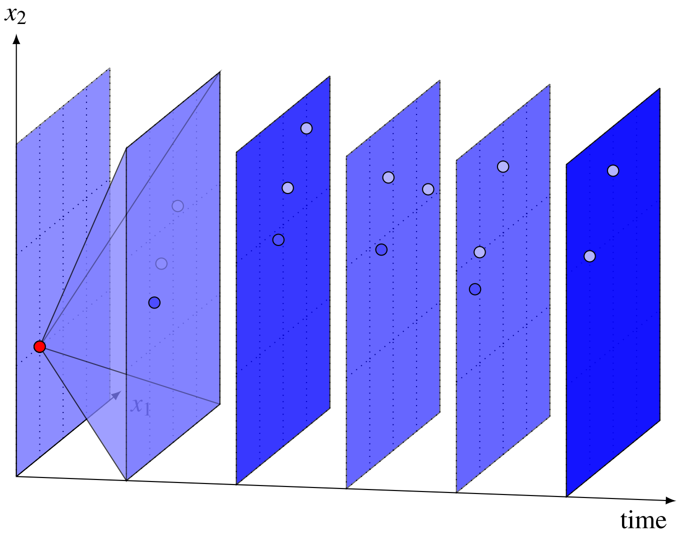
third backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



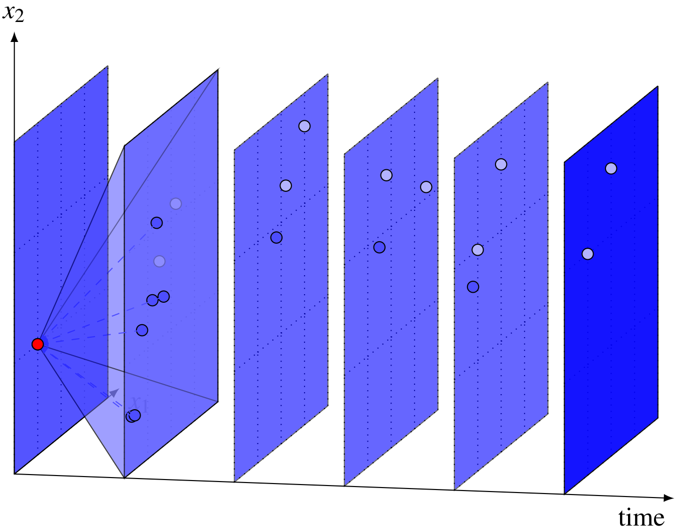
third backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



third backward pass : refining approximation (adding cuts)

Trajectory Following Dynamic Programming



And so on...

Contents

Problem setting

- The generic Multistage Stochastic Program considered reads

$$\begin{aligned} \min \quad & \mathbb{E}\Big[\sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t, \xi_t)\Big] && \text{(MSP)} \\ \text{s.t.} \quad & \boldsymbol{x}_t \in \mathcal{X}_t(\boldsymbol{x}_{t-1}, \xi_t) && \forall t \\ & \sigma(\boldsymbol{x}_t) \subset \sigma(\{\xi_\tau\}_{\tau\in[t]}) && \forall t \end{aligned}$$

- Note that:
  - finite, discrete time
  - contraints are stagewise independent
  - cost could depend on $\boldsymbol{x}_{t-1}$ or $\boldsymbol{u}_t$ if needed
  - risk-neutral[2]

[2] can be relaxed

Problem setting

- The generic Multistage Stochastic Program considered reads

$$
\begin{aligned}
\min \quad & \mathbb{E}\Big[\sum_{t=1}^{T} \ell_t(\boldsymbol{x}_t, \boldsymbol{\xi}_t)\Big] && \text{(MSP)} \\
\text{s.t.} \quad & \boldsymbol{x}_t \in \mathcal{X}_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_t) && \forall t \\
& \sigma(\boldsymbol{x}_t) \subset \sigma(\{\boldsymbol{\xi}_\tau\}_{\tau\in[t]}) && \forall t
\end{aligned}
$$

- Note that:
  - finite, discrete time
  - contraints are stagewise independent
  - cost could depend on $\boldsymbol{x}_{t-1}$ or $\boldsymbol{u}_t$ if needed
  - risk-neutral[2]

[2] can be relaxed

Backward Bellman operators and Dynamic Programming

Define the cost-to-go function

$$
\begin{aligned}
V_{t_0}(x) = \min \quad & \mathbb{E}\Big[\sum_{t=t_0+1}^{T} \ell_t(\boldsymbol{x}_t, \boldsymbol{\xi}_t)\Big] & \\
\text{s.t.} \quad & \boldsymbol{x}_{t_0} = x & \\
& \boldsymbol{x}_t \in \mathcal{X}_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_t) & \forall t > t_0 \\
& \sigma(\boldsymbol{x}_t) \subset \sigma(\{\boldsymbol{\xi}_\tau\}_{\tau \in [t]}) & \forall t > t_0
\end{aligned}
$$

Assuming that $(\boldsymbol{\xi}_\tau)_{\tau \in [T]}$ is stagewise independent, we have

$$V_t = \mathcal{B}_t(V_{t+1})$$

where the Backward Bellman operator $\mathcal{B}_t$ is defined

$$
\hat{\mathcal{B}}_t(\tilde{V}) := \begin{cases} \mathbb{R}^{n_t} \times \Xi_{t+1} & \to \mathbb{R} \cup \{+\infty\} \\ (x_t, \xi_{t+1}) & \mapsto \min\limits_{x_{t+1} \in \mathcal{X}_{t+1}(x_t, \xi_{t+1})} \underbrace{\ell_{t+1}(x_{t+1}, \xi_{t+1})}_{\text{transition costs}} + \underbrace{\tilde{V}(x_{t+1})}_{\text{cost-to-go}} \end{cases}
$$

$$\mathcal{B}_t(\tilde{V}) : x_t \mapsto \mathbb{E}\big[\hat{\mathcal{B}}_t(\tilde{V})(x_t, \boldsymbol{\xi}_{t+1})\big]$$

Backward Bellman operators and Dynamic Programming

Define the cost-to-go function

$$
\begin{aligned}
V_{t_0}(x) = \min \quad & \mathbb{E}\Big[\sum_{t=t_0+1}^{T} \ell_t(\boldsymbol{x}_t, \boldsymbol{\xi}_t) \Big] \\
\text{s.t.} \quad & \boldsymbol{x}_{t_0} = x \\
& \boldsymbol{x}_t \in \mathcal{X}_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_t) && \forall t > t_0 \\
& \sigma(\boldsymbol{x}_t) \subset \sigma(\{\boldsymbol{\xi}_\tau\}_{\tau \in [t]}) && \forall t > t_0
\end{aligned}
$$

Assuming that $(\boldsymbol{\xi}_\tau)_{\tau \in [T]}$ is stagewise independent, we have

$$V_t = \mathcal{B}_t(V_{t+1})$$

where the Backward Bellman operator $\mathcal{B}_t$ is defined

$$
\hat{\mathcal{B}}_t(\tilde{V}) := \begin{cases} \mathbb{R}^{n_t} \times \Xi_{t+1} & \to \mathbb{R} \cup \{+\infty\} \\ (x_t, \xi_{t+1}) & \mapsto \min\limits_{x_{t+1} \in \mathcal{X}_{t+1}(x_t, \xi_{t+1})} \underbrace{\ell_{t+1}(x_{t+1}, \xi_{t+1})}_{\text{transition costs}} + \underbrace{\tilde{V}(x_{t+1})}_{\text{cost-to-go}} \end{cases}
$$

$$\mathcal{B}_t(\tilde{V}) : x_t \mapsto \mathbb{E}\big[\hat{\mathcal{B}}_t(\tilde{V})(x_t, \boldsymbol{\xi}_{t+1})\big]$$

Backward Bellman operators and Dynamic Programming

Define the cost-to-go function

$$
\begin{aligned}
V_{t_0}(x) = \min \quad & \mathbb{E}\Big[\sum_{t=t_0+1}^{T} \ell_t(\boldsymbol{x}_t, \boldsymbol{\xi}_t)\Big] & \\
\text{s.t.} \quad & \boldsymbol{x}_{t_0} = x & \\
& \boldsymbol{x}_t \in \mathcal{X}_t(\boldsymbol{x}_{t-1}, \boldsymbol{\xi}_t) & \forall t > t_0 \\
& \sigma(\boldsymbol{x}_t) \subset \sigma(\{\boldsymbol{\xi}_\tau\}_{\tau\in[t]}) & \forall t > t_0
\end{aligned}
$$

Assuming that $(\boldsymbol{\xi}_\tau)_{\tau\in[T]}$ is stagewise independent, we have

$$V_t = \mathcal{B}_t(V_{t+1})$$

where the Backward Bellman operator $\mathcal{B}_t$ is defined

$$
\hat{\mathcal{B}}_t(\tilde{V}) := \begin{cases} \mathbb{R}^{n_t} \times \Xi_{t+1} & \to \mathbb{R} \cup \{+\infty\} \\ (x_t, \xi_{t+1}) & \mapsto \min\limits_{x_{t+1}\in\mathcal{X}_{t+1}(x_t,\xi_{t+1})} \underbrace{\ell_{t+1}(x_{t+1}, \xi_{t+1})}_{\text{transition costs}} + \underbrace{\tilde{V}(x_{t+1})}_{\text{cost-to-go}} \end{cases}
$$

$$\mathcal{B}_t(\tilde{V}) : x_t \mapsto \mathbb{E}\big[\hat{\mathcal{B}}_t(\tilde{V})(x_t, \boldsymbol{\xi}_{t+1})\big]$$

Reachable sets

We define the reachable sets

$$
\begin{aligned}
X_0^r &= \{x_0\} \\
X_t^r &= \bigcup_{x_{t-1} \in X_{t-1}^r} \bigcup_{\xi \in \Xi_t} \mathcal{X}_t(x_{t-1}, \xi) && \forall t \in [T].
\end{aligned}
$$

➥ this is the set of state that one can attain starting from the initial point.

Cost-to-go induced policy and Forward Bellman operator

- Denote the set of γ-optimal solution of stage-t problem

$$\mathcal{X}^{\sharp}_{\gamma,t}(\tilde{V}) : (x, \xi) \mapsto \gamma\text{-} \operatorname*{arg\,min}_{y \in \mathcal{X}_t(x,\xi)} \ell_t(y, \xi) + \tilde{V}(y).$$

- We say that $\mathcal{F}_t$ is a γ-forward Bellman operator of step t, if, for all function[3] $\tilde{V}$, $\mathcal{F}_t(\tilde{V})$ is a measurable selection of $\mathcal{X}^{\sharp}_{\gamma,t}(\tilde{V})$.

➥ It means that the stage problem are solve with the same deterministic solver.

- A given (collection of) forward operator $(\mathcal{F}_t)_{t\in[T]}$ define, for any (collection of) cost-to-go approximations $(\tilde{V}_t)_{t\in[T]}$, a policy.
- A policy define, for any scenario $(\xi_t)_{t\in[T]}$, a trajectory and its associated cost.

[3] Lipschitz on X_t^r

Cost-to-go induced policy and Forward Bellman operator

- Denote the set of γ-optimal solution of stage-t problem

$$\mathcal{X}^{\sharp}_{\gamma,t}(\tilde{V}) : (x, \xi) \mapsto \gamma\text{-} \operatorname*{arg\,min}_{y \in \mathcal{X}_t(x,\xi)} \ell_t(y, \xi) + \tilde{V}(y).$$

- We say that $\mathcal{F}_t$ is a γ-forward Bellman operator of step t, if, for all function[3] $\tilde{V}$, $\mathcal{F}_t(\tilde{V})$ is a measurable selection of $\mathcal{X}^{\sharp}_{\gamma,t}(\tilde{V})$.

➥ It means that the stage problem are solve with the same deterministic solver.

- A given (collection of) forward operator $(\mathcal{F}_t)_{t\in[T]}$ define, for any (collection of) cost-to-go approximations $(\tilde{V}_t)_{t\in[T]}$, a policy.
- A policy define, for any scenario $(\xi_t)_{t\in[T]}$, a trajectory and its associated cost.

[3] Lipschitz on X_t^r

Cost-to-go induced policy and Forward Bellman operator

- Denote the set of γ-optimal solution of stage-t problem

$$\mathcal{X}^{\sharp}_{\gamma,t}(\tilde{V}) : (x, \xi) \mapsto \gamma\text{-} \operatorname*{arg\,min}_{y \in \mathcal{X}_t(x,\xi)} \ell_t(y, \xi) + \tilde{V}(y).$$

- We say that $\mathcal{F}_t$ is a γ-forward Bellman operator of step t, if, for all function[3] $\tilde{V}$, $\mathcal{F}_t(\tilde{V})$ is a measurable selection of $\mathcal{X}^{\sharp}_{\gamma,t}(\tilde{V})$.

➥ It means that the stage problem are solve with the same deterministic solver.

- A given (collection of) forward operator $(\mathcal{F}_t)_{t\in[T]}$ define, for any (collection of) cost-to-go approximations $(\tilde{V}_t)_{t\in[T]}$, a policy.
- A policy define, for any scenario $(\xi_t)_{t\in[T]}$, a trajectory and its associated cost.

[3] Lipschitz on X_t^r

Trajectory Following Dynamic Programming algorithms

TFDP algorithms refine outer-approximations of the cost-to-go functions:

1. using the current outer-approximation we compute a trajectory (forward phase)
2. around the computed trajectory we refine the outer-approximations (backward phase)

A few comments:

- The forward phase depends on two elements:
  - the chosen forward operator $\mathcal{F}_t$
  - the node-selection ξ_t^k method
- An inner cost-to-go approximation is sometimes computed and used in the node-selection process. It is required for the complexity analysis, but can be set to V_t.
- Outer approximation are defined as maximum of elementary functions called cuts.

Trajectory Following Dynamic Programming algorithms

TFDP algorithms refine outer-approximations of the cost-to-go functions:

1. using the current outer-approximation we compute a trajectory (forward phase)
2. around the computed trajectory we refine the outer-approximations (backward phase)

A few comments:

- The forward phase depends on two elements:
  - the chosen forward operator $\mathcal{F}_t$
  - the node-selection ξ_t^k method
- An inner cost-to-go approximation is sometimes computed and used in the node-selection process. It is required for the complexity analysis, but can be set to V_t.
- Outer approximation are defined as maximum of elementary functions called cuts.

Cuts

Consider a function F that we approximate with a function f^k, called a cut, with respect to a point x^k.
We say that

1. f^k is $\underline{\gamma}_t$-tight if $f^k(x^k) \geq F(x^k) - \underline{\gamma}_t$
2. f^k is valid if $f^k \leq F$

These definitions are used to define the outer-approximations of V_t:

$$\underline{V}_t^k := \max_{\kappa \leq k} f_t^\kappa$$

where f_t^κ is a

- L_t-Lipschitz on X_t^r
- valid
- $\underline{\gamma}$-tight

cut of $\mathcal{B}_t(\underline{V}_{t+1}^\kappa)$

Cuts

Consider a function F that we approximate with a function f^k, called a cut, with respect to a point x^k.
We say that

1. f^k is $\underline{\gamma}_t$-tight if $f^k(x^k) \geq F(x^k) - \underline{\gamma}_t$
2. f^k is valid if $f^k \leq F$

These definitions are used to define the outer-approximations of V_t:

$$\underline{V}_t^k := \max_{\kappa \leq k} f_t^\kappa$$

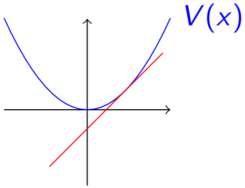
where f_t^κ is a

- L_t-Lipschitz on X_t^r
- valid
- $\underline{\gamma}$-tight

cut of $\mathcal{B}_t(\underline{V}_{t+1}^\kappa)$

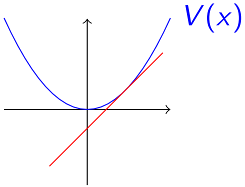
Example of cuts

1. Affine Bender's cut
2. Affine Lagrangian cuts
3. Affine integer cuts

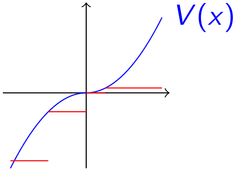


Example of cuts

1. Affine Bender's cut
2. Affine Lagrangian cuts
3. Affine integer cuts

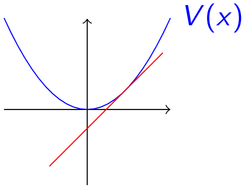


4. Step cuts

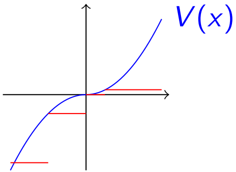


Example of cuts

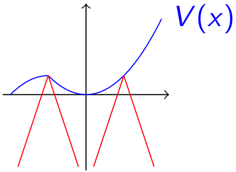
1. Affine Bender's cut
2. Affine Lagrangian cuts
3. Affine integer cuts



4. Step cuts



5. Lipschitz-cuts



Algorithm 2: A general framework for TFDP algorithms

$\underline{V}_t^0 \equiv -\infty$ and $\overline{V}_t^0 \equiv +\infty$ for $t \in [T]$;

for $k \in \mathbb{N}$ **do**

  `/* Forward phase: compute trajectory */`

  Set $x_0^k = x_0$;

  for $t = 1 \to T-1$ **do**

    Choose $\xi_t^k \in \operatorname{supp}(\boldsymbol{\xi}_t)$; (node seletion)

    $x_t^k = \mathcal{F}_{t-1}(\underline{V}_t^{k-1})(x_{t-1}^k, \xi_t^k)$; (forward operator)

  `/* Backward phase: update approximations */`

  Set $\underline{V}_T^k \equiv \overline{V}_T^k \equiv 0$;

  for $t = T-1 \to 1$ **do**

    $f_t^k \leftarrow \underline{L}_t$-Lipschitz on X_t^r, valid and $\underline{\gamma}$-tight cut of $\mathcal{B}_t(\underline{V}_{t+1}^k)$ at x_t^k;

    $\underline{V}_t^k \leftarrow \max(\underline{V}_t^{k-1}, f_t^k)$;

    Define monotonous, $\bar{L}$-Lipschitz, valid, $\bar{\gamma}$-tight, $\overline{V}_t^k$;

Inner-approximation requirements

The inner approximation $\overline{V}_t^k$, not necessarily computed, shall satisfy the following properties:

1. $\overline{V}_t^k(x_t^k) \leq \mathcal{B}_t(\overline{V}_{t+1}^k)(x_t^k) + \bar{\gamma}_t$ (tightness)
2. $\overline{V}_t^k \geq \mathcal{B}_t(\overline{V}_{t+1}^k)$ (validity)
3. $\overline{V}_t^k \leq \overline{V}_t^{k-1}$ (monotonicity)
4. $\overline{V}_t^k$ is $\bar{L}_t$-Lipschitz

Some TFDP algorithms

Algorithm's name	Node selection: Choice ξ_t^k	$\mathcal{F}_t$	$\underline{V}_t^k$	$\overline{V}_t^k$	Hypothesis	Complexity known
SDDP	Random sampling	Exact	Benders cuts	V_t	Convex	✔
EDDP	Explorative	Exact	Benders cuts	V_t	Convex	✔
APSDDP	Random sampling	Exact	Adaptive partition	V_t	Linear	✖
SDDiP	Random sampling	Exact	Lagrangian or integer cuts	V_t	Mixed Integer Linear	✖
MIDAS	Random sampling	Exact	Step cuts	V_t	Monotonic Mixed Integer	✖
SLDP	Random sampling	Exact	Reverse norm cuts	V_t	Non-Convex	✖
BDZ17	Problem child	Exact	Benders cuts	Epigraph as convex hull	Convex	✖
BDZ18	Problem child	Exact	Benders $\times$ Epigraph	Hypograph $\times$ Benders	Convex-Concave	✖
RDDP	Deterministic	Exact	Benders cuts	Epigraph as convex hull	Robust	✖
ISDDP	Random sampling	Inexact	Inexact Lagrangian cuts	V_t	Convex	✖
TDP	Problem child	Exact	Benders cuts	Min of quadratic	Convex	✖
ZS19	Random or Problem	Regularized	Generalized conjugacy cuts	Norm cuts	Mixed Integer Convex	✔
NDDP	Random or Problem	Regularized	Benders cuts	Norm cuts	Distributionally Robust	✔
DSDDP	Random sampling	Exact	Benders cuts	Fenchel transform	Linear	✖

Contents

Structural Assumptions I

Independence of noises

$(\xi_t)_{t\in[T]}$ is a sequence of independent exogeneous random variables, i.e. such that the law of ξ_t is independent of all decisions variables.

Structural Assumptions

Independence of noises

$(\boldsymbol{\xi}_t)_{t\in[T]}$ is a sequence of independent exogeneous random variables, i.e. such that the law of $\boldsymbol{\xi}_t$ is independent of all decisions variables.

Compatibility of constraints

We make the following assumptions, for all $t \in [T]$,

1. ℓ_t is a proper normal integrand;
2. for all $x_t \in X_t^r$, the random variable $\ell_t(x_t, \boldsymbol{\xi}_t)$ is integrable;
3. for all $x_{t-1} \in X_{t-1}^r$ and almost all $\xi_t \in \Xi_{t-1}$,
   $\mathcal{X}_t(x_{t-1}, \xi_t)$ is non-empty compact

➥ imply relatively complete recourse

Structural Assumptions II

Lipschitz

For $t \in [T]$, we assume that

1. X_t^r has a diameter smaller than $D_t < +\infty$;
2. the expected cost-to-go function V_t is L_t-Lipschitz.

Structural Assumptions II

Lipschitz

For $t \in [T]$, we assume that

1. X_t^r has a diameter smaller than $D_t < +\infty$;
2. the expected cost-to-go function V_t is L_t-Lipschitz.

Existence of cuts

For every $t \in [T]$ and $k \in \mathbb{N}^\star$, there exists at least one $\underline{L}_t$-Lipschitz on X_t^r, valid and $\underline{\gamma}$-tight cut of $\mathcal{B}_t(\underline{V}_{t+1}^k)$ at x_t^k.

➥ Usually guaranteed through recourse assumptions.

Node selection

We study three node selection procedures:

1. random node selection: the noise ξ_t^k used to obtain x_t^k in the forward pass is selected randomly, independently of other node selection.
2. problem-child node selection: we choose the ξ_t^k that lead to a x_t^k maximizing the current gap estimate.
3. explorative node selection: we choose the ξ_t^k that lead to a x_t^k as far as possible of the set of "good points".

Node selection

We study three node selection procedures:

1. random node selection: the noise ξ_t^k used to obtain x_t^k in the forward pass is selected randomly, independently of other node selection.

➥ the most common, but hardest to study.

2. problem-child node selection: we choose the ξ_t^k that lead to a x_t^k maximizing the current gap estimate.

3. explorative node selection: we choose the ξ_t^k that lead to a x_t^k as far as possible of the set of "good points".

Node selection

We study three node selection procedures:

1. random node selection: the noise ξ_t^k used to obtain x_t^k in the forward pass is selected randomly, independently of other node selection.

➡ the most common, but hardest to study.

2. problem-child node selection: we choose the ξ_t^k that lead to a x_t^k maximizing the current gap estimate.

➡ some numerical advantages, and good theoretical guarantees.

3. explorative node selection: we choose the ξ_t^k that lead to a x_t^k as far as possible of the set of “good points”.

Node selection

We study three node selection procedures:

1. random node selection: the noise ξ_t^k used to obtain x_t^k in the forward pass is selected randomly, independently of other node selection.

➥ the most common, but hardest to study.

2. problem-child node selection: we choose the ξ_t^k that lead to a x_t^k maximizing the current gap estimate.

➥ some numerical advantages, and good theoretical guarantees.

3. explorative node selection: we choose the ξ_t^k that lead to a x_t^k as far as possible of the set of "good points".

➥ mainly theoretical.

Effective iterations

Consider $(\delta_t)_{t\in[T]} > 0$ and define

$$\begin{aligned}
\varepsilon_{T-1} &:= \underline{\gamma}_{T-1} + \bar{\gamma}_{T-1} \\
\varepsilon_t &:= \varepsilon_{t+1} + (\bar{L}_{t+1} + \underline{L}_{t+1})\delta_{t+1} + \gamma^F_{t+1} + \underline{\gamma}_t + \bar{\gamma}_t \quad \forall t \in [T-2] \\
\varepsilon_0 &:= \varepsilon_1 + (\bar{L}_1 + \underline{L}_1)\delta_1 + \gamma^F_1
\end{aligned}$$

- x_t^k is ε_t-saturated, if $\overline{V}_t^k(x_t^k) - \underline{V}_t^k(x_t^k) \leq \varepsilon_t$
- x_t^k is δ_t-distinguishable if $\|x_t^k - x_t^\kappa\| > \delta_t$ $\forall \kappa < k \mid x_t^\kappa$ is ε_t-saturated.

➥ An effective iteration $k \in \mathbb{N}$ generates either a ε_0 first stage lower-bound, or a new ε_t-saturated and δ_t-distinguishable point for at least one $t \in [T]$.

Effective iterations

Consider $(\delta_t)_{t\in[T]} > 0$ and define

$$\begin{aligned}
\varepsilon_{T-1} &:= \underline{\gamma}_{T-1} + \bar{\gamma}_{T-1} \\
\varepsilon_t &:= \varepsilon_{t+1} + (\bar{L}_{t+1} + \underline{L}_{t+1})\delta_{t+1} + \gamma^F_{t+1} + \underline{\gamma}_t + \bar{\gamma}_t \qquad \forall t \in [T-2] \\
\varepsilon_0 &:= \varepsilon_1 + (\bar{L}_1 + \underline{L}_1)\delta_1 + \gamma^F_1
\end{aligned}$$

- x^k_t is ε_t-saturated, if $\overline{V}^k_t(x^k_t) - \underline{V}^k_t(x^k_t) \leq \varepsilon_t$
- x^k_t is δ_t-distinguishable if $\|x^k_t - x^\kappa_t\| > \delta_t$ $\forall \kappa < k \mid x^\kappa_t$ is ε_t-saturated.

➥ An effective iteration $k \in \mathbb{N}$ generates either a ε_0 first stage lower-bound, or a new ε_t-saturated and δ_t-distinguishable point for at least one $t \in [T]$.

Effective iterations

Consider $(\delta_t)_{t\in[T]} > 0$ and define

$$\varepsilon_{T-1} := \underline{\gamma}_{T-1} + \bar{\gamma}_{T-1}$$
$$\varepsilon_t := \varepsilon_{t+1} + (\bar{L}_{t+1} + \underline{L}_{t+1})\delta_{t+1} + \gamma^F_{t+1} + \underline{\gamma}_t + \bar{\gamma}_t \qquad \forall t \in [T-2]$$
$$\varepsilon_0 := \varepsilon_1 + (\bar{L}_1 + \underline{L}_1)\delta_1 + \gamma^F_1$$

- x^k_t is ε_t-saturated, if $\overline{V}^k_t(x^k_t) - \underline{V}^k_t(x^k_t) \leq \varepsilon_t$
- x^k_t is δ_t-distinguishable if $\|x^k_t - x^\kappa_t\| > \delta_t$ $\forall \kappa < k \mid x^\kappa_t$ is ε_t-saturated.

➥ An effective iteration $k \in \mathbb{N}$ generates either a ε_0 first stage lower-bound, or a new ε_t-saturated and δ_t-distinguishable point for at least one $t \in [T]$.

Convergence results: by effective iterations

Theorem (bound on effective iterations number)

Assume that $\delta_t \in [0, D_t]$ and $\eta_t \in \mathbb{R}_+$ are given and ε_t defined as above. Let

$$\overline{K} := \sum_{t=1}^{T-1} \left(\frac{D_t}{\delta_t} + 1 \right)^{n_t}$$

After at most $\overline{K} + 1$ effective iterations we have a ε_1-lower bound:

$$\underline{V}_0^k(x_0) = \ell_1(x_1^k, \xi_1) + \underline{V}_1^k(x_1^k) \geq \mathrm{val}(MSP) - \varepsilon_1$$

Convergence results: by effective iterations

Theorem (bound on effective iterations number)

Assume that $\delta_t \in [0, D_t]$ *and* $\eta_t \in \mathbb{R}_+$ *are given and* ε_t *defined as above. Let*

$$\overline{K} := \sum_{t=1}^{T-1} \left(\frac{D_t}{\delta_t} + 1 \right)^{n_t}$$

After at most $\overline{K} + 1$ *effective iterations we have a* ε_1*-lower bound:*

$$\underline{V}_0^k(x_0) = \ell_1(x_1^k, \xi_1) + \underline{V}_1^k(x_1^k) \geq \mathrm{val}(MSP) - \varepsilon_1$$

Further, there exists, among those $\bar{K} + 1$ *effective iteration, at least one such that* x_1^k *is an* ε_0*-solution to problem (MSP):*

$$\ell_1(x_1^k, \xi_1) + V_1(x_1^k) \leq \mathrm{val}((MSP)) + \varepsilon_0$$

Convergence result: deterministic node selection

Theorem

If the node selection is done by problem child method or explorative method then each iteration is effective.

Convergence result: deterministic node selection

Theorem

If the node selection is done by problem child method or explorative method then each iteration is effective.

Theorem

Assume that each iteration is effective.
$\gamma_\Sigma := \sum_{t=1}^{T-1} \underline{\gamma}_t + \bar{\gamma}_t + \gamma_t^F$
$n_t \leq n$, $D_t = D$, $\overline{L}_t = \underline{L}_t = L$.

Then, for every $\varepsilon > \gamma_\Sigma$, sufficiently small (e.g. such that $\varepsilon \leq 2DL + \gamma_\Sigma$), TFDP finds an ε-first stage solution x_1^k within at most $\bar{K}_\varepsilon$ iterations where

$$\bar{K}_\varepsilon := \left(\frac{2DL}{\varepsilon - \gamma_\Sigma}\right)^n (T-1)^{n+1}$$

Convergence result: random node selection

Lemma

Assume that we draw $\xi_t^k \sim \boldsymbol{\xi}_t$, and independently of all other $\tilde{\xi}_\tau^\kappa$ as well as $(\boldsymbol{\xi}_\tau)_{\tau \in [T-1]}$.
Then,

$$\mathbb{P}\left[\textit{Iteration } k \textit{ is effective.} \,\middle|\, A^{k-1}\right] \geq \prod_{t=1}^{T} \left(1 - e^{\frac{-2\eta_t^2}{D_t^2}}\right)$$

Convergence result: random node selection

Lemma

Assume that we draw $\xi_t^k \sim \boldsymbol{\xi}_t$*, and independently of all other* $\tilde{\xi}_\tau^\kappa$ *as well as* $(\boldsymbol{\xi}_\tau)_{\tau \in [T-1]}$.
Then,

$$\mathbb{P}\left[\text{Iteration } k \text{ is effective.} \,\middle|\, A^{k-1}\right] \geq \prod_{t=1}^{T} \left(1 - e^{\frac{-2\eta_t^2}{D_t^2}}\right)$$

Theorem

Set $\gamma_\Sigma := \sum_{t=1}^{T-1} \underline{\gamma}_t + \bar{\gamma}_t + \gamma_t^F$ *and choose* n, D, L *such that, for all* $t \in [T-1]$, $n_t \leq n$, $D_t = D$, $\overline{L}_t = \underline{L}_t = L$.
Then, for $\varepsilon > \gamma_\Sigma$*, sufficiently small, the expected number of iterations of TFDP required to find an* ε*-first stage solution* x_1^k*, is bounded by* $(T-1)\left(\frac{4DL(T-1)}{\varepsilon - \gamma_\Sigma}\right)^{n+2(T-1)}$.

Contents

The assumption we did not make

Finitely supported noise

The support of the random process $(\xi_t)_{t\in[T]}$ is finite.

➥ To our knowledge all previous convergence results require this assumption.

➥ Further this assumption is sometimes "abused", using the fact that each scenario (10^{52}) is sampled an infinite number of time.

Computing cuts: finitely supported case

Recall that

$$\mathcal{B}_t(\tilde{V})(\cdot) = \mathbb{E}\big[\hat{\mathcal{B}}_t(\tilde{V})(\cdot, \boldsymbol{\xi}_{t+1})\big]$$

Thus, we can get cut for $\mathcal{B}(\tilde{V})$ as average of cuts for $\hat{\mathcal{B}}(\tilde{V})$

➥ This is easily done with the finitely supported noise assumption.

More precisely, if $\mathrm{supp}(\boldsymbol{\xi}_t) = \{\xi^1, \dots, \xi^N\}$, with $\mathbb{P}(\boldsymbol{\xi}_t = \xi^n) = \pi_n$:

- for each $n \in [N]$ compute a cut $\hat{f}_n$ for $\hat{\mathcal{B}}_t(\tilde{V})(\cdot, \xi^n)$ at x
- define $f := \sum_{n=1}^{N} \pi_n \hat{f}_n$ as a cut for $\mathcal{B}_t(\tilde{V})$ at x

Computing cuts: finitely supported case

Recall that

$$\mathcal{B}_t(\tilde{V})(\cdot) = \mathbb{E}\big[\hat{\mathcal{B}}_t(\tilde{V})(\cdot, \boldsymbol{\xi}_{t+1})\big]$$

Thus, we can get cut for $\mathcal{B}(\tilde{V})$ as average of cuts for $\hat{\mathcal{B}}(\tilde{V})$

➥ This is easily done with the finitely supported noise assumption.

More precisely, if $\mathrm{supp}(\boldsymbol{\xi}_t) = \{\xi^1, \dots, \xi^N\}$, with $\mathbb{P}(\boldsymbol{\xi}_t = \xi^n) = \pi_n$:

- for each $n \in [N]$ compute a cut $\hat{f}_n$ for $\hat{\mathcal{B}}_t(\tilde{V})(\cdot, \xi^n)$ at x
- define $f := \sum_{n=1}^N \pi_n \hat{f}_n$ as a cut for $\mathcal{B}_t(\tilde{V})$ at x

Computing cuts: non-finitely supported case

If ξ_t is not finitely supported, computing a cut is harder. However, there are at least two cases:

1. In the linear setting, using advanced polyhedral geometry tools (more on that in a few moments), we can compute exact cuts for some non-finitely supported noises.

2. In the convex setting, using convexity (Jensen's and Edmunson-Madanski) inequalities, we can derive inexact cuts.

Algorithm variations

Node selection Two main possibilities: random or problem-child. Other exists, like quasi-montercarlo, not covered by our results.

Forward operator Usually taken as an optimal solution, can also be the optimal solution of a regularized problem.

Multiple forward phases It is usual to simulate multiple trajectory in the forward phase before updating approximation to leverage parallelization.

Multicut Included in the framework with finitely supported assumption, unclear otherwise.

Cut selection To alleviate each iteration burden we sometimes use heuristics to drop cuts. This usually lose the convergence results and is not covered by our results.

Setting extensions

We can adapt the results to other problem settings:

Minimax Can be seen as a two-player repeated stochastic game. The results can be adapted as long as we can compute inner and outer approximation and both reachable sets are of finite dimension and diameter.

Robust Special case of minimax problem.

Risk-averse Using nested coherent risk-measure formulation it is a special case of minimax. Results apply if the risk set can be described by a finite number of parameters, in particular if either

- we consider polyhedral coherent risk measures;
- or we consider mix of expectation and AVAR.

Questions ?