Multi-Objective Planning under Uncertainty

Shimon Whiteson & Diederik M. Roijers

Department of Computer Science
University of Oxford

June 13, 2016

Schedule

- 14:00-14:40: Motivation & Concepts (Shimon)
- 14:40-14:50: Short Break
- 14:50-15:30: Motivation & Concepts cont'd (Shimon)
- 15:30-16:00: Coffee Break
- 16:00-16:40: Methods (Diederik)
- 16:40-16:50: Short Break
- 16:50-17:30: Methods & Applications (Diederik)

Note

- Get the latest version of the slides at:

  `http://roijers.info/motutorial.html`

- This tutorial is based on our survey article:

  Diederik Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. *A Survey of Multi-Objective Sequential Decision-Making*. Journal of Artificial Intelligence Research, 48:67—113, 2013.

  and Diederik's dissertation:
  `http://roijers.info/pub/thesis.pdf`

Part 1: Motivation & Concepts

- *Multi-Objective Motivation*
- MDPs & MOMDPs
- Problem Taxonomy
- Solution Concepts

Medical Treatment

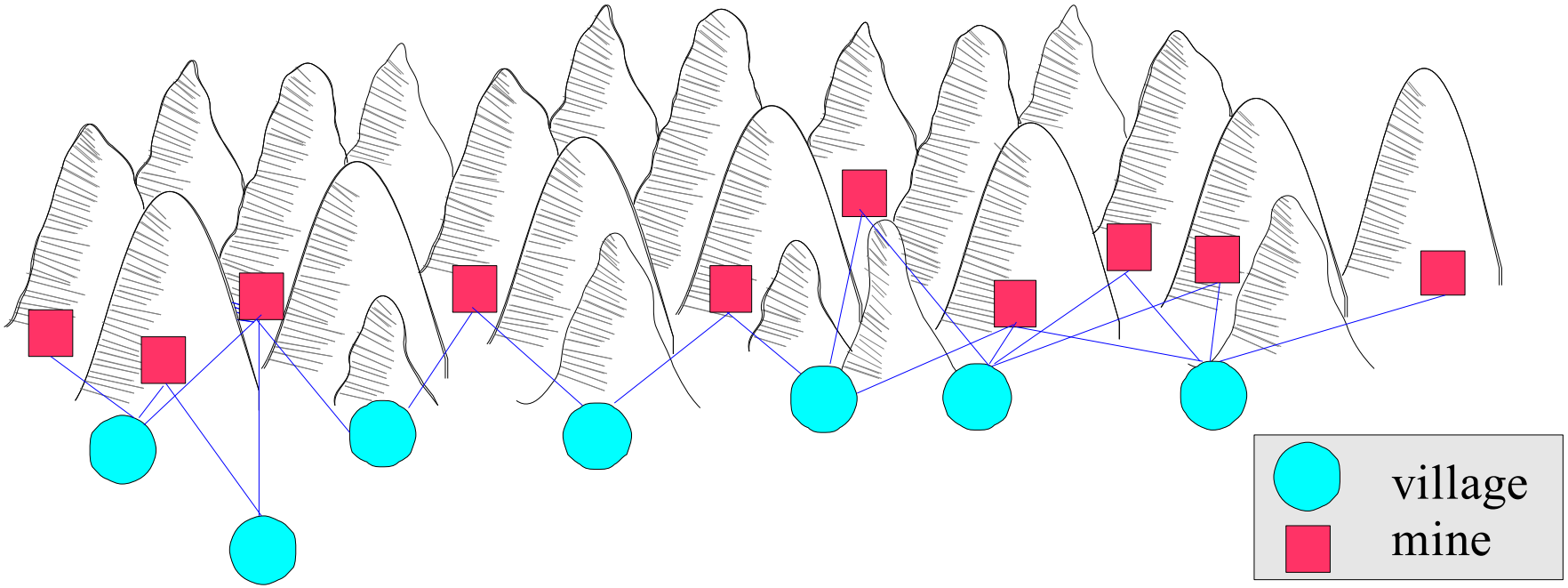
Chance of being cured, having side effects, or dying

Traffic Coordination

Latency, throughput, fairness, environmental impact, etc.

Mining Commodities

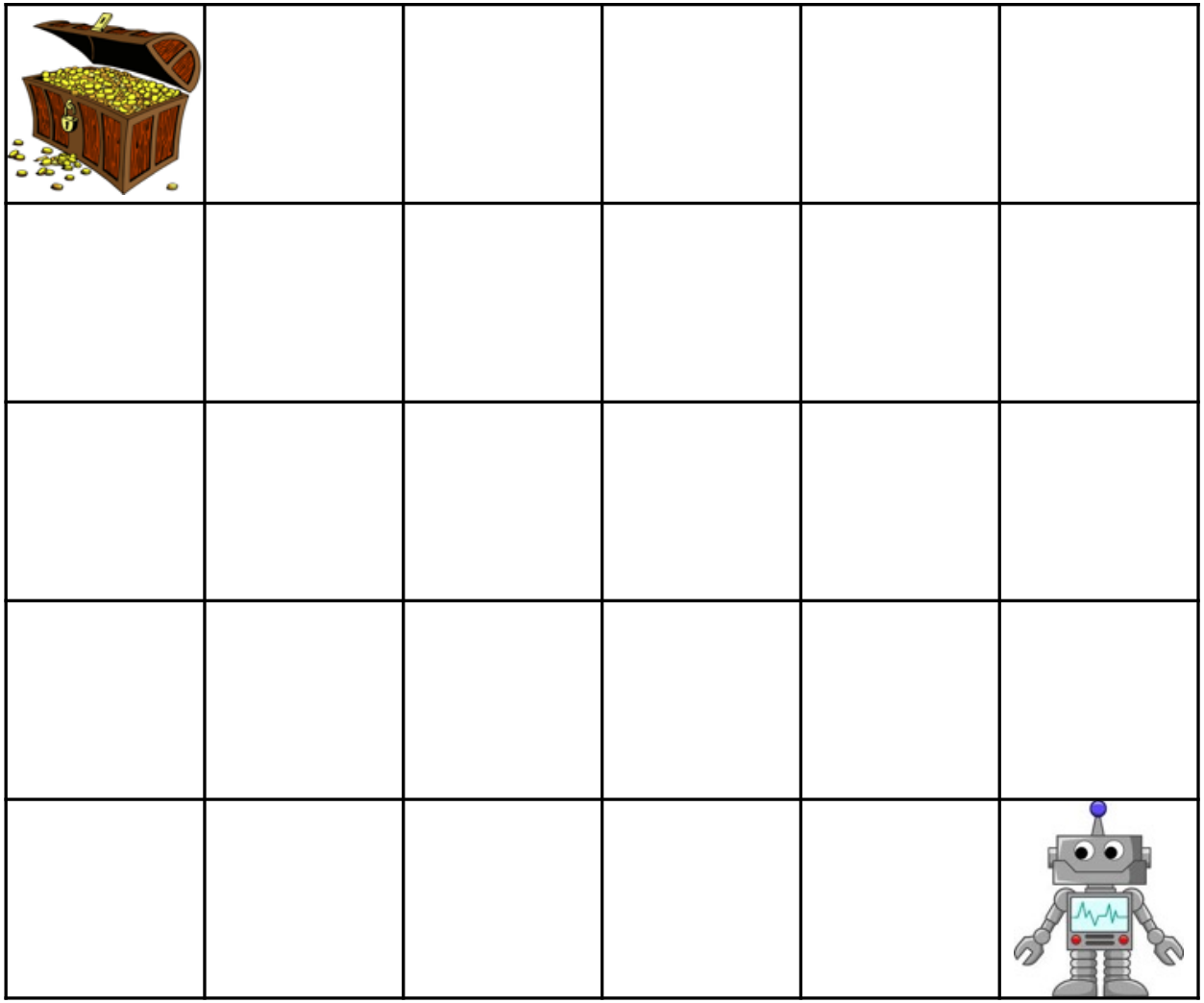
Gold collected, silver collected



[Roijers et al. 2013, 2014]

Grid World

Getting the treasure, minimising fuel costs



Do We Need Multi-Objective Models?

Do We Need Multi-Objective Models?

Sutton's Reward Hypothesis: "All of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received *scalar* signal (reward)."

Source: `http://rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html`

Do We Need Multi-Objective Models?

Sutton's Reward Hypothesis: "All of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received *scalar* signal (reward)."

Source: `http://rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html`

- $V : \Pi \rightarrow \mathbb{R}$
- $V^{\pi} = E_{\pi}[\sum_t r_t]$
- $\pi^* = \max_{\pi} V^{\pi}$

Why Multi-Objective Decision Making?

- *The weak argument*: real-world problems are multi-objective!

$$\mathbf{V} : \Pi \rightarrow \mathbb{R}^n$$

Why Multi-Objective Decision Making?

- *The weak argument*: real-world problems are multi-objective!

$$\mathbf{V} : \Pi \rightarrow \mathbb{R}^n$$

- *Objection*: why not just *scalarize*?

Why Multi-Objective Decision Making?

- *The weak argument*: real-world problems are multi-objective!

$$\mathbf{V} : \Pi \rightarrow \mathbb{R}^n$$

- *Objection*: why not just *scalarize*?
- *Scalarization function* projects multi-objective value to a scalar:

$$V^{\pi}_{\mathbf{w}} = f(\mathbf{V}^{\pi}, \mathbf{w})$$

- Linear case:

$$V^{\pi}_{\mathbf{w}} = \sum_{i=1}^{n} w_i V^{\pi}_i = \mathbf{w} \cdot \mathbf{V}^{\pi}$$

- A priori prioritization of the objectives

Why Multi-Objective Decision Making?

- *The weak argument*: real-world problems are multi-objective!

$$\mathbf{V} : \Pi \rightarrow \mathbb{R}^n$$

- *Objection*: why not just *scalarize*?
- *Scalarization function* projects multi-objective value to a scalar:

$$V^{\pi}_{\mathbf{w}} = f(\mathbf{V}^{\pi}, \mathbf{w})$$

- Linear case:

$$V^{\pi}_{\mathbf{w}} = \sum_{i=1}^{n} w_i V^{\pi}_i = \mathbf{w} \cdot \mathbf{V}^{\pi}$$

- A priori prioritization of the objectives
- *The weak argument is necessary but not sufficient*

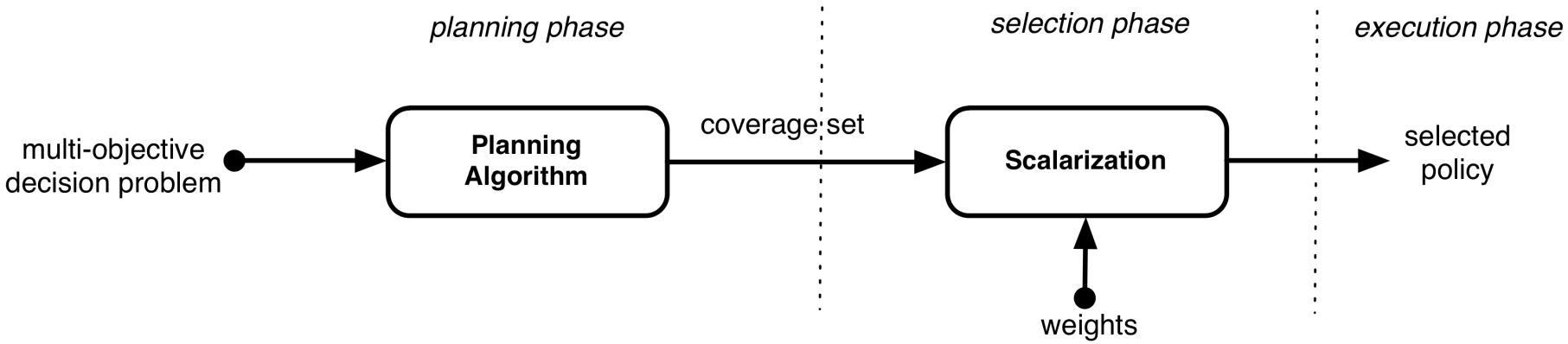
Why Multi-Objective Decision Making?

- *The strong argument*: a priori scalarization is sometimes impossible, infeasible, or undesirable

- Instead produce the *coverage set* of undominated solutions

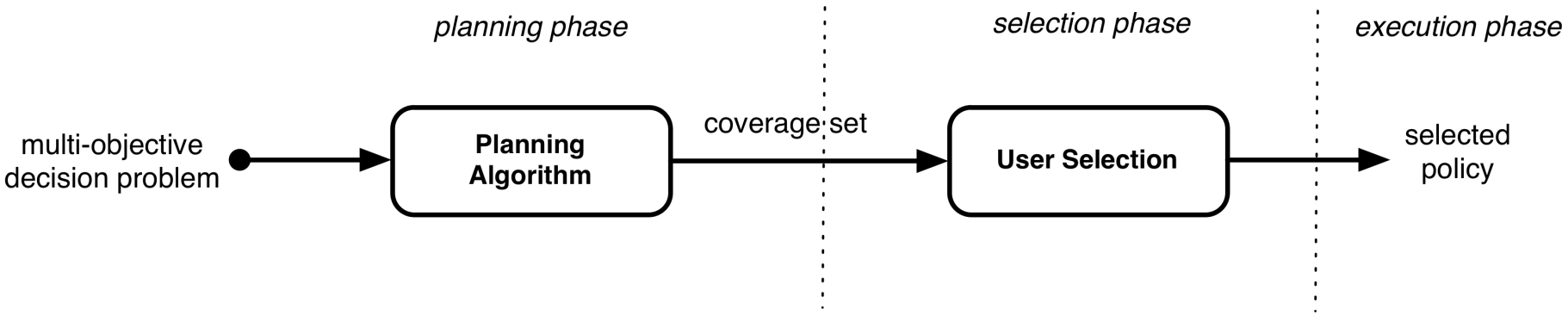
Why Multi-Objective Decision Making?

- *The strong argument*: a priori scalarization is sometimes impossible, infeasible, or undesirable
- Instead produce the *coverage set* of undominated solutions
- *Unknown-weights* scenario
  - Weights known in *execution phase* but not in *planning phase*
  - Example: mining commodities [Roijers et al. 2013]



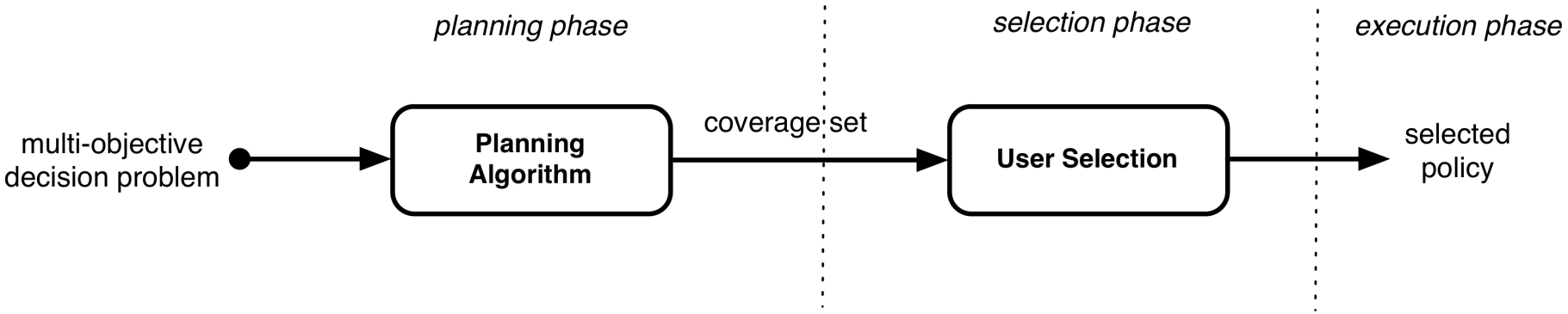
Why Multi-Objective Decision Making?

- *Decision-support* scenario
  - Quantifying priorities is infeasible
  - Choosing between options is easier
  - Example: medical treatment

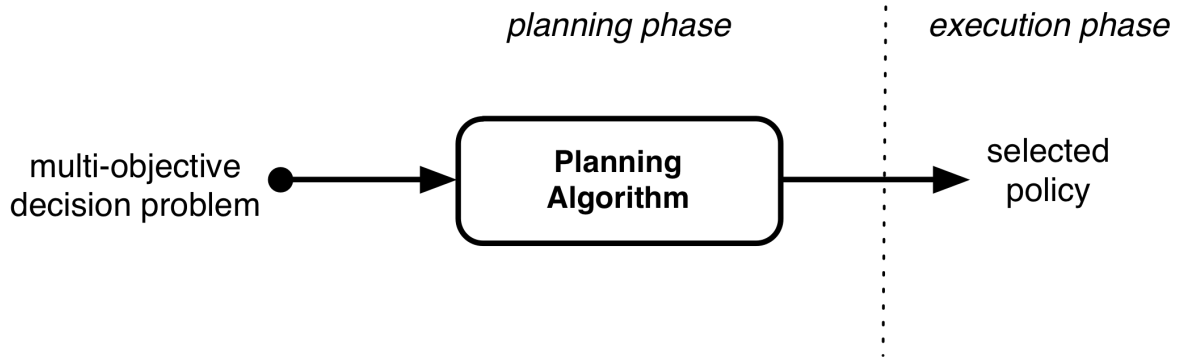


Why Multi-Objective Decision Making?

- *Decision-support* scenario
  - Quantifying priorities is infeasible
  - Choosing between options is easier
  - Example: medical treatment



- *Known-weights* scenario: scalarization yields intractable problem



Summary of Motivation

Multi-objective methods are useful because many problems are naturally characterized by multiple objectives and cannot be easily scalarized a priori.

Summary of Motivation

Multi-objective methods are useful because many problems are naturally characterized by multiple objectives and cannot be easily scalarized a priori.

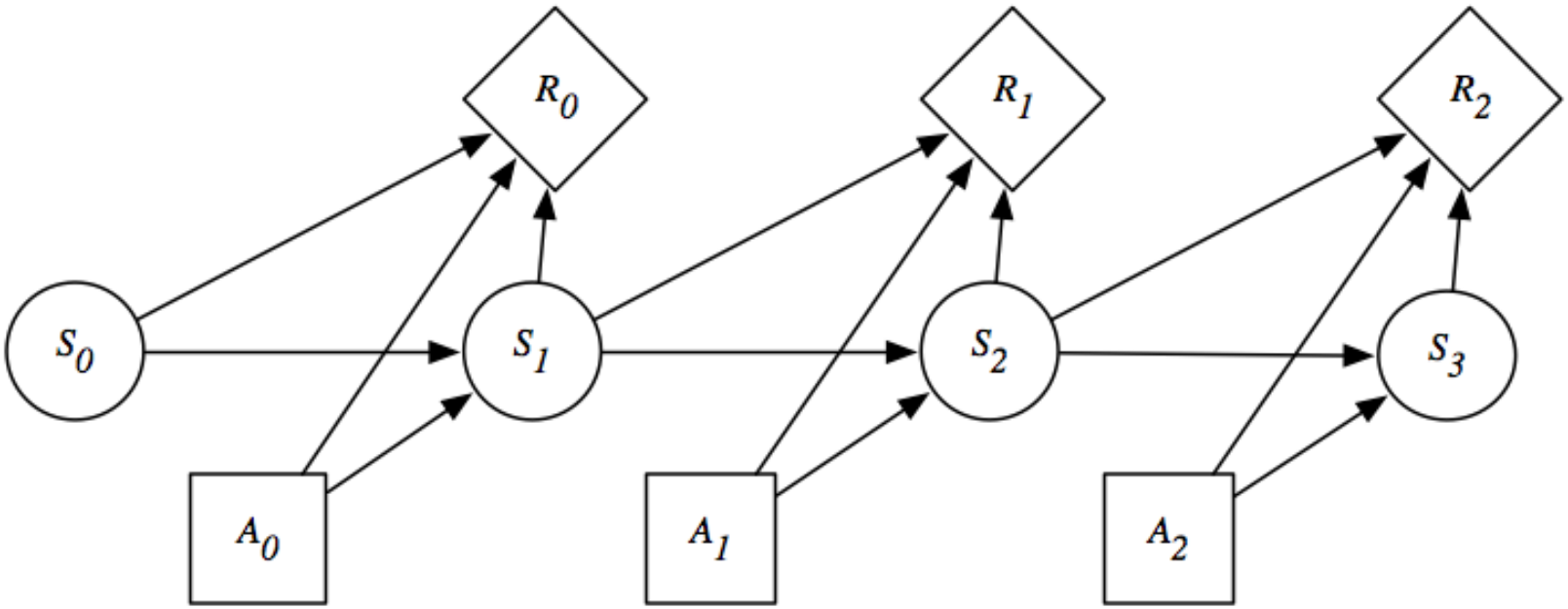
The burden of proof rests with the a priori scalarization, not with the multi-objective modeling.

Part 1: Motivation & Concepts

- Multi-Objective Motivation
- *MDPs & MOMDPs*
- Problem Taxonomy
- Solution Concepts

Markov Decision Process (MDP)

- A single-objective MDP is a tuple $\langle S, A, T, R, \mu, \gamma \rangle$ where:
  - S is a finite set of *states*
  - A is a finite set of *actions*
  - $T : S \times A \times S \rightarrow [0, 1]$ is a *transition function*
  - $R : S \times A \times S \rightarrow \mathbb{R}$ is a *reward function*
  - $\mu : S \rightarrow [0, 1]$ is a probability distribution over initial states
  - $\gamma \in [0, 1)$ is a *discount factor*



(figure from Poole & Mackworth, *Artificial Intelligence: Foundations of Computational Agents*, 2010)

Returns & Policies

- Goal: maximize expected *return*, which is typically *additive*:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$

- A *stationary policy* conditions only on the current state:

$$\pi : S \times A \to [0, 1]$$

- A *deterministic stationary policy* maps states directly to actions:

$$\pi : S \to A$$

Value Functions in MDPs

- A *state-independent value function* V^{π} specifies the expected return when following π from the initial state:

$$V^{\pi} = E[R_0 \mid \pi] \tag{1}$$

- A *state value function* of a policy π:

$$V^{\pi}(s) = E[R_t \mid \pi, s_t = s]$$

- The *Bellman equation* restates this expectation recursively for stationary policies:

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi}(s')]$$

Optimality in MDPs

Theorem

For any additive infinite-horizon single-objective MDP, there exists a deterministic stationary optimal policy [Howard 1960]

- All optimal policies share the same *optimal value function*:

$$V^*(s) = \max_\pi V^\pi(s)$$

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

- Extract the optimal policy using *local action selection*:

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

Multi-Objective MDP (MOMDP)

- Vector-valued reward and value:

$$\mathbf{R} : S \times A \times S \rightarrow \mathbb{R}^n$$

$$\mathbf{V}^{\pi} = E[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{k+1} \mid \pi]$$

$$\mathbf{V}^{\pi}(s) = E[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{t+k+1} \mid \pi, s_t = s]$$

- $\mathbf{V}^{\pi}(s)$ imposes only a *partial ordering*, e.g.,

$$V_i^{\pi}(s) > V_i^{\pi'}(s) \text{ but } V_j^{\pi}(s) < V_j^{\pi'}(s).$$

- Definition of optimality no longer clear

Part 1: Motivation & Concepts

- Multi-Objective Motivation
- MDPs & MOMDPs
- *Problem Taxonomy*
- Solution Concepts

Axiomatic vs. Utility-Based Approach

- *Axiomatic approach*: define optimal solution set to be *Pareto front*

Axiomatic vs. Utility-Based Approach

- *Axiomatic approach*: define optimal solution set to be *Pareto front*
- *Utility-based approach*:
  - *Execution phase*: select one policy maximizing scalar utility $V^{\pi}_{\mathbf{w}}$, where $\mathbf{w}$ may be hidden or implicit

Axiomatic vs. Utility-Based Approach

- *Axiomatic approach*: define optimal solution set to be *Pareto front*
- *Utility-based approach*:
  - *Execution phase*: select one policy maximizing scalar utility $V^{\pi}_{\mathbf{w}}$, where $\mathbf{w}$ may be hidden or implicit
  - *Planning phase*: find set of policies containing optimal solution for each possible $\mathbf{w}$; if $\mathbf{w}$ unknown, size of set generally > 1

Axiomatic vs. Utility-Based Approach

- *Axiomatic approach*: define optimal solution set to be *Pareto front*
- *Utility-based approach*:
  - *Execution phase*: select one policy maximizing scalar utility $V^{\pi}_{\mathbf{w}}$, where $\mathbf{w}$ may be hidden or implicit
  - *Planning phase*: find set of policies containing optimal solution for each possible $\mathbf{w}$; if $\mathbf{w}$ unknown, size of set generally > 1
  - Deduce optimal solution set from three factors:
    1. Multi-objective scenario
    2. Properties of scalarization function
    3. Allowable policies

Three Factors

1. Multi-objective scenario
   - Known weights $\rightarrow$ single policy
   - Unknown weights or decision support $\rightarrow$ multiple policies
2. Properties of scalarization function
   - Linear
   - Monotonically increasing
3. Allowable policies
   - Deterministic
   - Stochastic

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Part 1: Motivation & Concepts

- Multi-Objective Motivation
- MDPs & MOMDPs
- Problem Taxonomy
- *Solution Concepts*

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	*one deterministic stationary policy*		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Linear Scalarization Functions

- Computes inner product of $\mathbf{w}$ and $\mathbf{V}^{\pi}$:

$$V^{\pi}_{\mathbf{w}} = \sum_{i=1}^{n} w_i V^{\pi}_i = \mathbf{w} \cdot \mathbf{V}^{\pi}, \ \mathbf{w} \in \mathbb{R}^n$$

- w_i quantifies importance of i-th objective
- Simple and intuitive, e.g., when utility translates to money:

$$\mathit{revenue} = \#\mathit{cans} \times \mathit{ppc} + \#\mathit{bottles} \times \mathit{ppb}$$

Linear Scalarization Functions

- Computes inner product of $\mathbf{w}$ and $\mathbf{V}^\pi$:

$$V^\pi_{\mathbf{w}} = \sum_{i=1}^{n} w_i V^\pi_i = \mathbf{w} \cdot \mathbf{V}^\pi, \ \mathbf{w} \in \mathbb{R}^n$$

- w_i quantifies importance of i-th objective
- Simple and intuitive, e.g., when utility translates to money:

$$\mathit{revenue} = \#\mathit{cans} \times \mathit{ppc} + \#\mathit{bottles} \times \mathit{ppb}$$

- $V^\pi_{\mathbf{w}}$ typically constrained to be a *convex combination*:

$$\forall i \ w_i \geq 0, \qquad \sum_i w_i = 1$$

$$\mathit{utility} = \#\mathit{cans} \times \frac{\mathit{ppc}}{\mathit{ppc} + \mathit{ppb}} + \#\mathit{bottles} \times \frac{\mathit{ppb}}{\mathit{ppc} + \mathit{ppb}}$$

Linear Scalarization & Single Policy

- No special methods required: just apply f to each reward vector
- Inner product distributes over addition yielding a normal MDP:

$$V^{\pi}_{\mathbf{w}} = \mathbf{w} \cdot \mathbf{V}^{\pi} = \mathbf{w} \cdot E[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{t+k+1}] = E[\sum_{k=0}^{\infty} \gamma^k (\mathbf{w} \cdot \mathbf{r}_{t+k+1})]$$

- Apply standard methods to an MDP with:

$$R(s, a, s') = \mathbf{w} \cdot \mathbf{R}(s, a, s'), \tag{2}$$

yielding a single determinstic stationary policy

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	*one deterministic stationary policy*		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Example: collecting bottles and cans

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	*one deterministic stationary policy*		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Example: collecting bottles and cans

Note: only cell in taxonomy that does not require multi-objective methods

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		*convex coverage set of deterministic stationary policies*	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Multiple Policies

- Unknown weights or decision support $\rightarrow$ multiple policies
- During planning $\mathbf{w}$ is unknown
- Size of solution set is generally > 1
- *Set should not contain policies suboptimal for all* $\mathbf{w}$

Undominated & Coverage Sets

Definition

The undominated set $U(\Pi)$, *is the subset of all possible policies* Π *for which there exists a* $\mathbf{w}$ *for which the scalarized value is maximal,*

$$U(\Pi) = \{\pi : \pi \in \Pi \wedge \exists \mathbf{w} \forall (\pi' \in \Pi)\ V^{\pi}_{\mathbf{w}} \geq V^{\pi'}_{\mathbf{w}}\}$$

Undominated & Coverage Sets

Definition

The undominated set $U(\Pi)$, *is the subset of all possible policies* Π *for which there exists a* $\mathbf{w}$ *for which the scalarized value is maximal,*

$$U(\Pi) = \{\pi : \pi \in \Pi \wedge \exists \mathbf{w} \forall (\pi' \in \Pi)\ V^{\pi}_{\mathbf{w}} \geq V^{\pi'}_{\mathbf{w}}\}$$

Definition

A coverage set $CS(\Pi)$ *is a subset of* $U(\Pi)$ *that, for every* $\mathbf{w}$, *contains a policy with maximal scalarized value, i.e.,*

$$CS(\Pi) \subseteq U(\Pi) \wedge (\forall \mathbf{w})(\exists \pi) \left(\pi \in CS(\Pi) \wedge \forall (\pi' \in \Pi)\ V^{\pi}_{\mathbf{w}} \geq V^{\pi'}_{\mathbf{w}}\right)$$

Example

V_w^π	$w = \text{true}$	$w = \text{false}$
$\pi = \pi_1$	5	0
$\pi = \pi_2$	0	5
$\pi = \pi_3$	5	2
$\pi = \pi_4$	2	2

- One binary weight feature: only two possible weights
- Weights are not objectives but two possible scalarizations

Example

V_w^π	$w = \text{true}$	$w = \text{false}$
$\pi = \pi_1$	5	0
$\pi = \pi_2$	0	5
$\pi = \pi_3$	5	2
$\pi = \pi_4$	2	2

- One binary weight feature: only two possible weights
- Weights are not objectives but two possible scalarizations
- $U(\Pi) = \{\pi_1, \pi_2, \pi_3\}$ but $CS(\Pi) = \{\pi_1, \pi_2\}$ or $\{\pi_2, \pi_3\}$

Execution Phase

- Single policy selected from $CS(\Pi)$ and executed
- *Unknown weights*: weights revealed and maximizing policy selected:

$$\pi^* = \arg\max_{\pi \in CS(\Pi)} V^{\pi}_{\mathbf{w}}$$

- *Decision support*: $CS(\Pi)$ is manually inspected by the user

Linear Scalarization & Multiple Policies

Definition

The convex hull $CH(\Pi)$ is the subset of Π for which there exists a $\mathbf{w}$ that maximizes the linearly scalarized value:

$$CH(\Pi) = \{\pi : \pi \in \Pi \wedge \exists \mathbf{w} \forall (\pi' \in \Pi)\ \mathbf{w} \cdot \mathbf{V}^{\pi} \geq \mathbf{w} \cdot \mathbf{V}^{\pi'}\}$$

Linear Scalarization & Multiple Policies

Definition

The convex hull $CH(\Pi)$ *is the subset of* Π *for which there exists a* $\mathbf{w}$ *that maximizes the linearly scalarized value:*

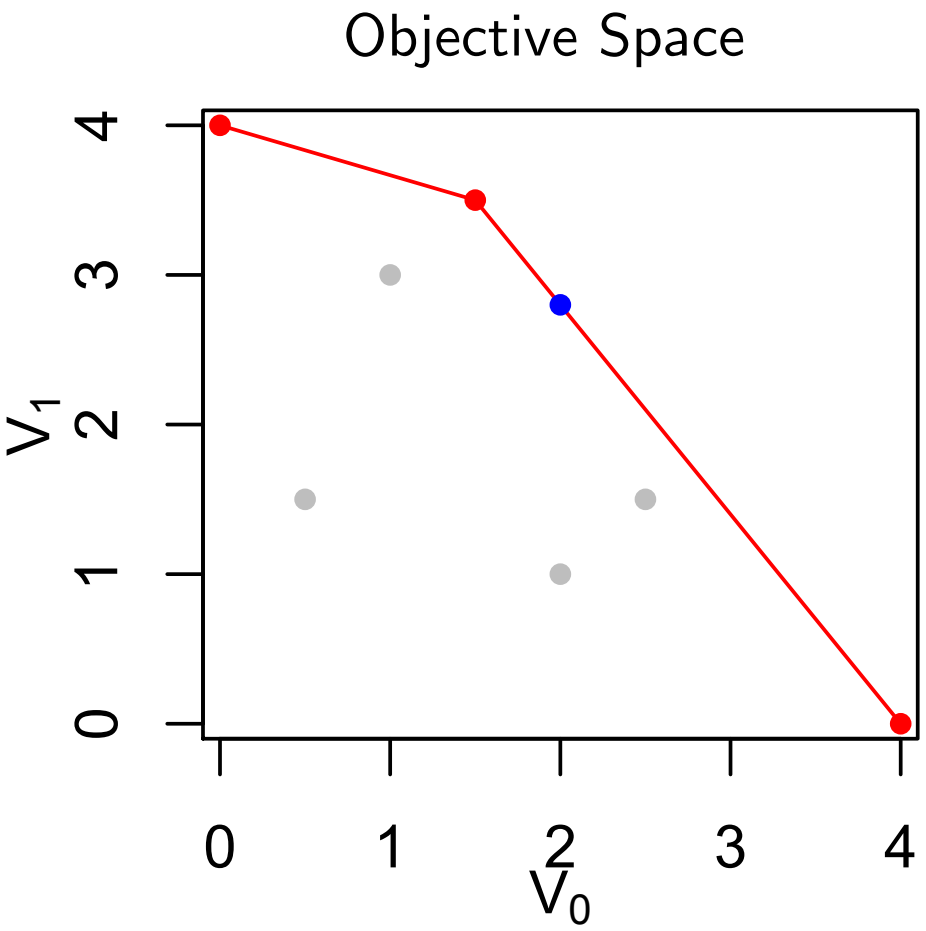
$$CH(\Pi) = \{\pi : \pi \in \Pi \wedge \exists \mathbf{w} \forall (\pi' \in \Pi)\ \mathbf{w} \cdot \mathbf{V}^{\pi} \geq \mathbf{w} \cdot \mathbf{V}^{\pi'}\}$$

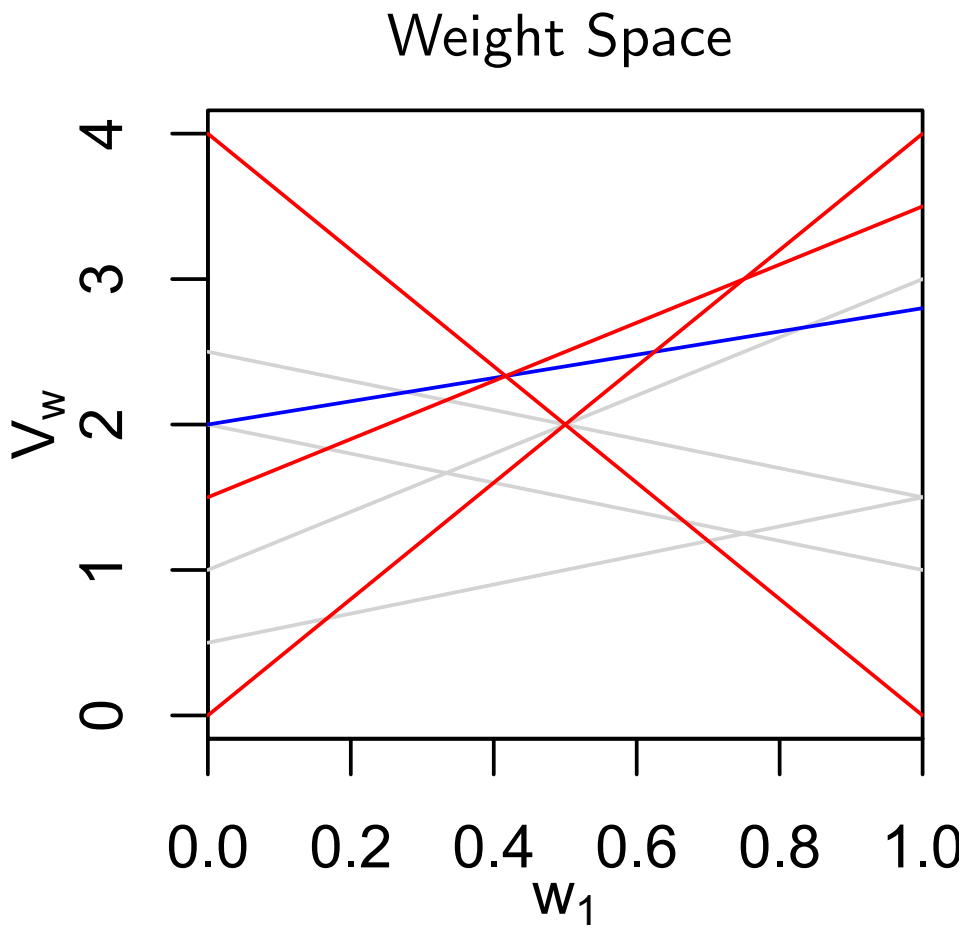
Definition

The convex coverage set $CCS(\Pi)$ *is a subset of* $CH(\Pi)$ *that, for every* $\mathbf{w}$*, contains a policy whose linearly scalarized value is maximal, i.e.,*

$$CCS(\Pi) \subseteq CH(\Pi) \wedge (\forall \mathbf{w})(\exists \pi) \left(\pi \in CCS(\Pi) \wedge \forall (\pi' \in \Pi)\ \mathbf{w} \cdot \mathbf{V}^{\pi} \geq \mathbf{w} \cdot \mathbf{V}^{\pi'}\right)$$

Visualization





$$V_{\mathbf{w}} = w_0 V_0 + w_1 V_1 \ , \ \ w_0 = 1 - w_1$$

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		*convex coverage set of deterministic stationary policies*	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

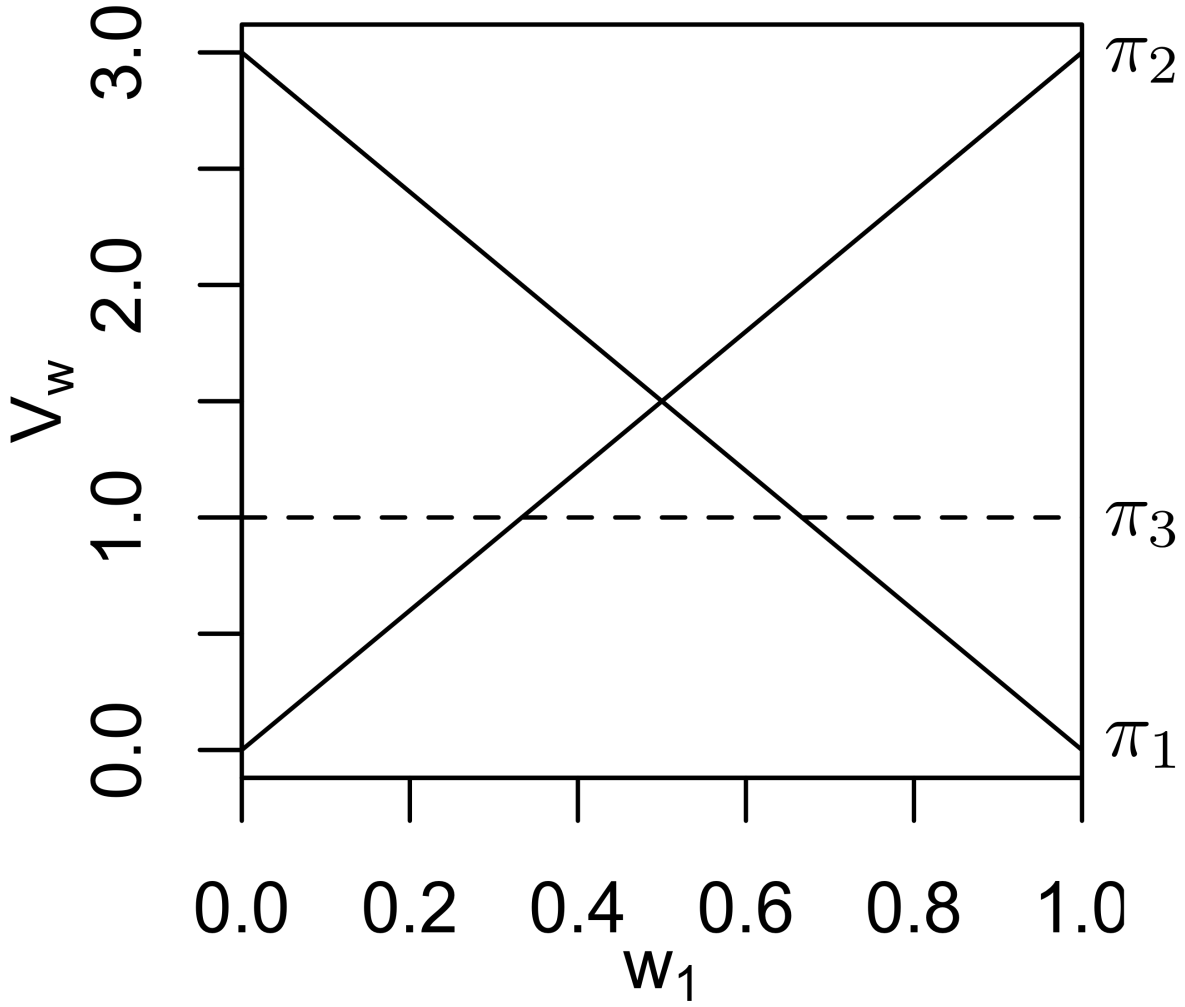
Example: mining gold and silver

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	*one deterministic non-stationary policy*	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Monotonically Increasing Scalarization Functions

- Mining example: $\mathbf{V}^{\pi_1} = (3, 0)$, $\mathbf{V}^{\pi_2} = (0, 3)$, $\mathbf{V}^{\pi_3} = (1, 1)$
- Choosing $\mathbf{V}^{\pi_3}$ implies nonlinear scalarization function



Monotonically Increasing Scalarization Functions

Definition

A scalarization function is strictly monotonically increasing if changing a policy such that its value increases in one or more objectives, without decreasing in any other objectives, also increases the scalarized value:

$$(\forall i \; V_i^{\pi} \geq V_i^{\pi'} \wedge \exists i \; V_i^{\pi} > V_i^{\pi'}) \Rightarrow (\forall \mathbf{w} \; V_{\mathbf{w}}^{\pi} > V_{\mathbf{w}}^{\pi'})$$

Monotonically Increasing Scalarization Functions

Definition

A scalarization function is strictly monotonically increasing if changing a policy such that its value increases in one or more objectives, without decreasing in any other objectives, also increases the scalarized value:

$$(\forall i \; V_i^{\pi} \geq V_i^{\pi'} \wedge \exists i \; V_i^{\pi} > V_i^{\pi'}) \Rightarrow (\forall \mathbf{w} \; V_{\mathbf{w}}^{\pi} > V_{\mathbf{w}}^{\pi'})$$

Definition

A policy π Pareto-dominates another policy π' when its value is at least as high in all objectives and strictly higher in at least one objective:

$$\mathbf{V}^{\pi} \succ_P \mathbf{V}^{\pi'} \Leftrightarrow \forall i \; V_i^{\pi} \geq V_i^{\pi'} \wedge \exists i \; V_i^{\pi} > V_i^{\pi'}$$

A policy is Pareto optimal if no policy Pareto-dominates it.

Nonlinear Scalarization Can Destroy Additivity

- Nonlinear scalarization and expectation do not commute:

$$V^{\pi}_{\mathbf{w}} = f(\mathbf{V}^{\pi}, \mathbf{w}) = f(E[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{t+k+1}], \mathbf{w}) \neq E[\sum_{k=0}^{\infty} \gamma^k f(\mathbf{r}_{t+k+1}, \mathbf{w})]$$

- Bellman-based methods not applicable
- Local action selection no longer yields an optimal policy:

$$\pi^*(s) \neq \arg\max V^*(s)$$

Deterministic vs. Stochastic Policies

- Stochastic policies are fine in most settings
- Sometimes inappropriate, e.g., medical treatment
- In MDPs, requiring deterministic policies is not restrictive
- Optimal value attainable with deterministic stationary policy:

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^*(s')]$$

Deterministic vs. Stochastic Policies

- Stochastic policies are fine in most settings
- Sometimes inappropriate, e.g., medical treatment
- In MDPs, requiring deterministic policies is not restrictive
- Optimal value attainable with deterministic stationary policy:

$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

- Similar for MOMDPs with linear scalarization
- MOMDPs with nonlinear scalarization:
  - Stochastic policies may be preferable if allowed
  - Nonstationary policies may be preferable otherwise

White's Example (1982)

- 3 actions: $\mathbf{R}(a_1) = (3, 0), \mathbf{R}(a_2) = (0, 3), \mathbf{R}(a_3) = (1, 1)$

White's Example (1982)

- 3 actions: $\mathbf{R}(a_1) = (3, 0), \mathbf{R}(a_2) = (0, 3), \mathbf{R}(a_3) = (1, 1)$
- 3 deterministic stationary policies, all Pareto-optimal:

$$\mathbf{V}^{\pi_1} = \left(\frac{3}{1-\gamma}, 0\right), \mathbf{V}^{\pi_2} = \left(0, \frac{3}{1-\gamma}\right), \mathbf{V}^{\pi_3} = \left(\frac{1}{1-\gamma}, \frac{1}{1-\gamma}\right)$$

White's Example (1982)

- 3 actions: $\mathbf{R}(a_1) = (3, 0), \mathbf{R}(a_2) = (0, 3), \mathbf{R}(a_3) = (1, 1)$
- 3 deterministic stationary policies, all Pareto-optimal:

$$\mathbf{V}^{\pi_1} = \left(\frac{3}{1-\gamma}, 0\right), \mathbf{V}^{\pi_2} = \left(0, \frac{3}{1-\gamma}\right), \mathbf{V}^{\pi_3} = \left(\frac{1}{1-\gamma}, \frac{1}{1-\gamma}\right)$$

- π_{ns} alternates between a_1 and a_2, starting with a_1:

$$\mathbf{V}^{\pi_{ns}} = \left(\frac{3}{1-\gamma^2}, \frac{3\gamma}{1-\gamma^2}\right)$$

White's Example (1982)

- 3 actions: $\mathbf{R}(a_1) = (3, 0), \mathbf{R}(a_2) = (0, 3), \mathbf{R}(a_3) = (1, 1)$
- 3 deterministic stationary policies, all Pareto-optimal:

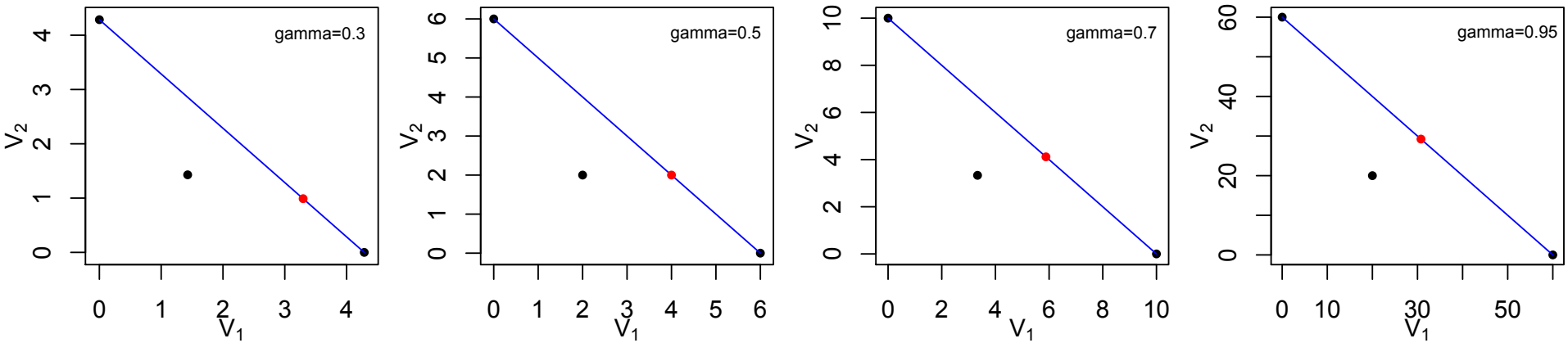
$$\mathbf{V}^{\pi_1} = \left(\frac{3}{1-\gamma}, 0\right), \mathbf{V}^{\pi_2} = \left(0, \frac{3}{1-\gamma}\right), \mathbf{V}^{\pi_3} = \left(\frac{1}{1-\gamma}, \frac{1}{1-\gamma}\right)$$

- π_{ns} alternates between a_1 and a_2, starting with a_1:

$$\mathbf{V}^{\pi_{ns}} = \left(\frac{3}{1-\gamma^2}, \frac{3\gamma}{1-\gamma^2}\right)$$

- Thus $\pi_{ns} \succ_P \pi_3$ when $\gamma \geq 0.5$, e.g., $\gamma = 0.5$ and $f(\mathbf{V}^\pi) = V_1^\pi V_2^\pi$:

$$V^{\pi_1} = V^{\pi_2} = 0, V^{\pi_3} = 4, V^{\pi_{ns}} = 8$$



Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	*one deterministic non-stationary policy*	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Example: radiation vs. chemotherapy

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	*one mixture policy of two or more deterministic stationary policies*	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Mixture Policies

- A *mixture policy* π_m selects i-th policy from set of N deterministic policies with probability p_i, where $\sum_{i=0}^{N} p_i = 1$
- Values are convex combination of values of constituent policies
- In White's example, replace π_{ns} by π_m:

$$\mathbf{V}^{\pi_m} = p_1 \mathbf{V}^{\pi_1} + (1 - p_1)\mathbf{V}^{\pi_2} = \left(\frac{3p_1}{1-\gamma}, \frac{3(1-p_1)}{1-\gamma} \right)$$

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	*one mixture policy of two or more deterministic stationary policies*	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Example: studying vs. networking

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	*Pareto coverage set of deterministic non-stationary policies*	convex coverage set of deterministic stationary policies

Pareto Sets

Definition

The Pareto front is the set of all policies that are not Pareto dominated:

$$PF(\Pi) = \{\pi : \pi \in \Pi \wedge \neg\exists(\pi' \in \Pi), \mathbf{V}^{\pi'} \succ_P \mathbf{V}^{\pi}\}$$

Pareto Sets

Definition

The Pareto front is the set of all policies that are not Pareto dominated:

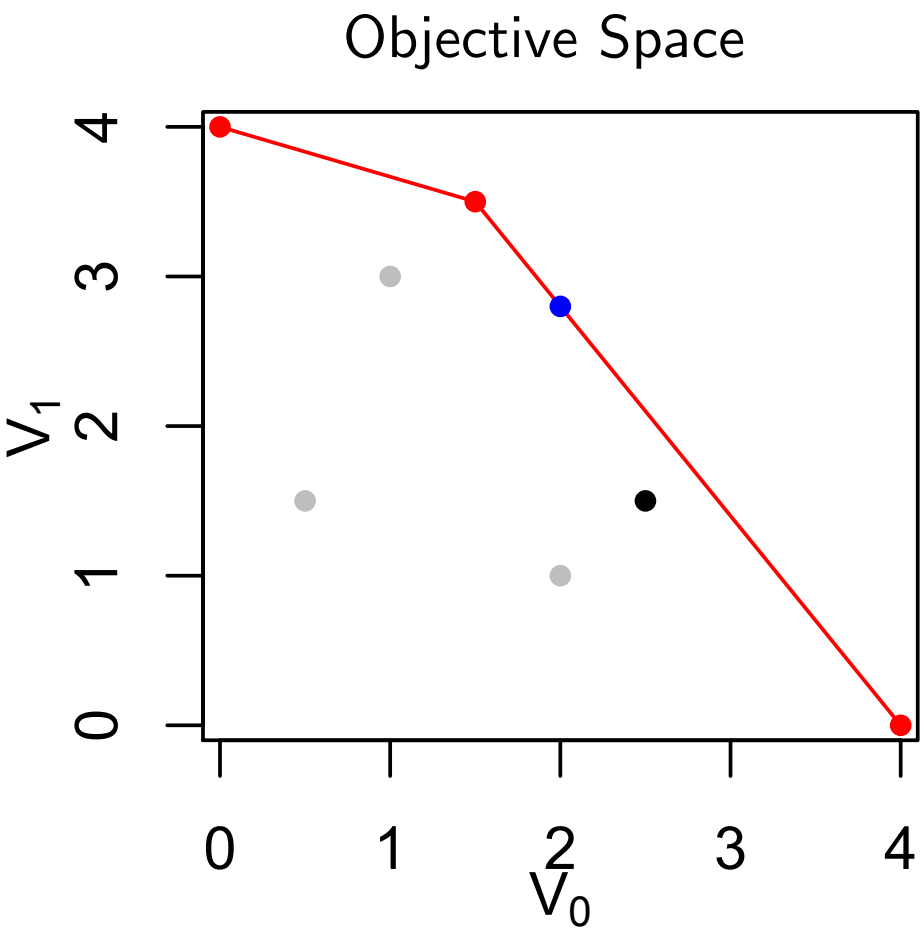
$$PF(\Pi) = \{\pi : \pi \in \Pi \wedge \neg\exists(\pi' \in \Pi), \mathbf{V}^{\pi'} \succ_P \mathbf{V}^{\pi}\}$$

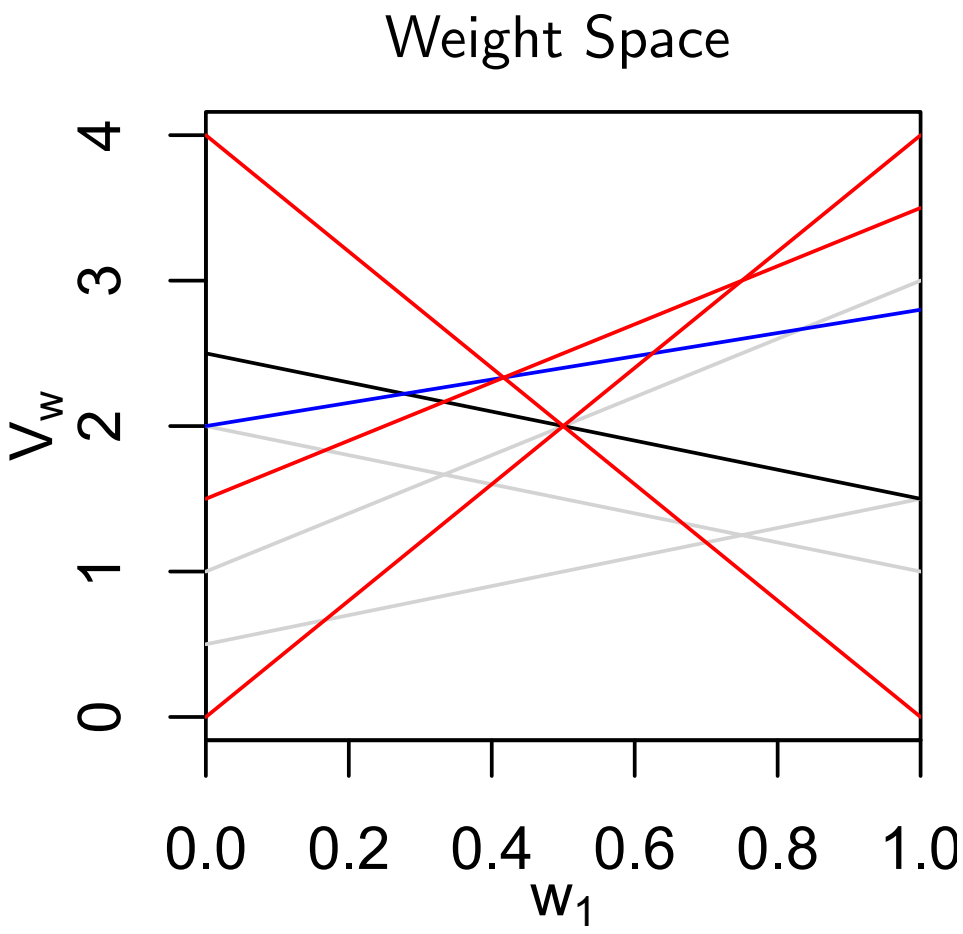
Definition

A Pareto coverage set is a subset of $PF(\Pi)$ such that, for every $\pi' \in \Pi$, it contains a policy that either dominates π' or has equal value to π':

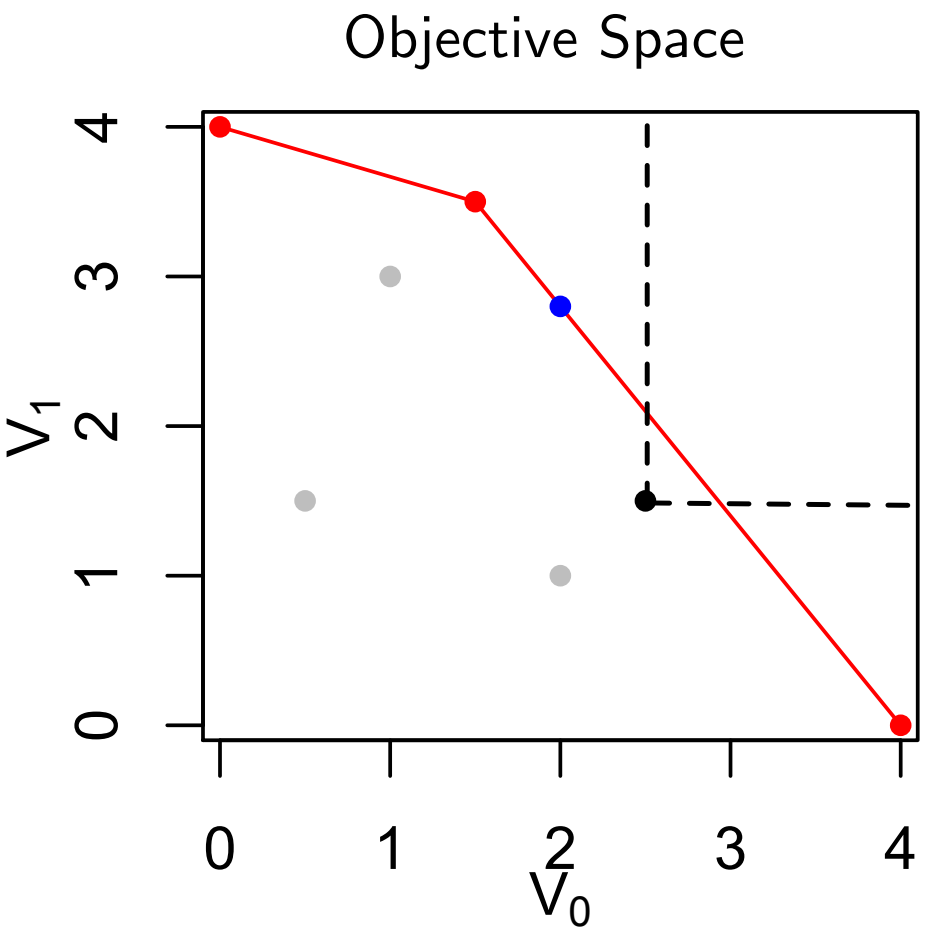
$$PCS(\Pi) \subseteq PF(\Pi) \wedge \forall(\pi' \in \Pi)(\exists\pi)\Big(\pi \in PCS(\Pi) \wedge (\mathbf{V}^{\pi} \succ_P \mathbf{V}^{\pi'} \vee \mathbf{V}^{\pi} = \mathbf{V}^{\pi'})\Big)$$

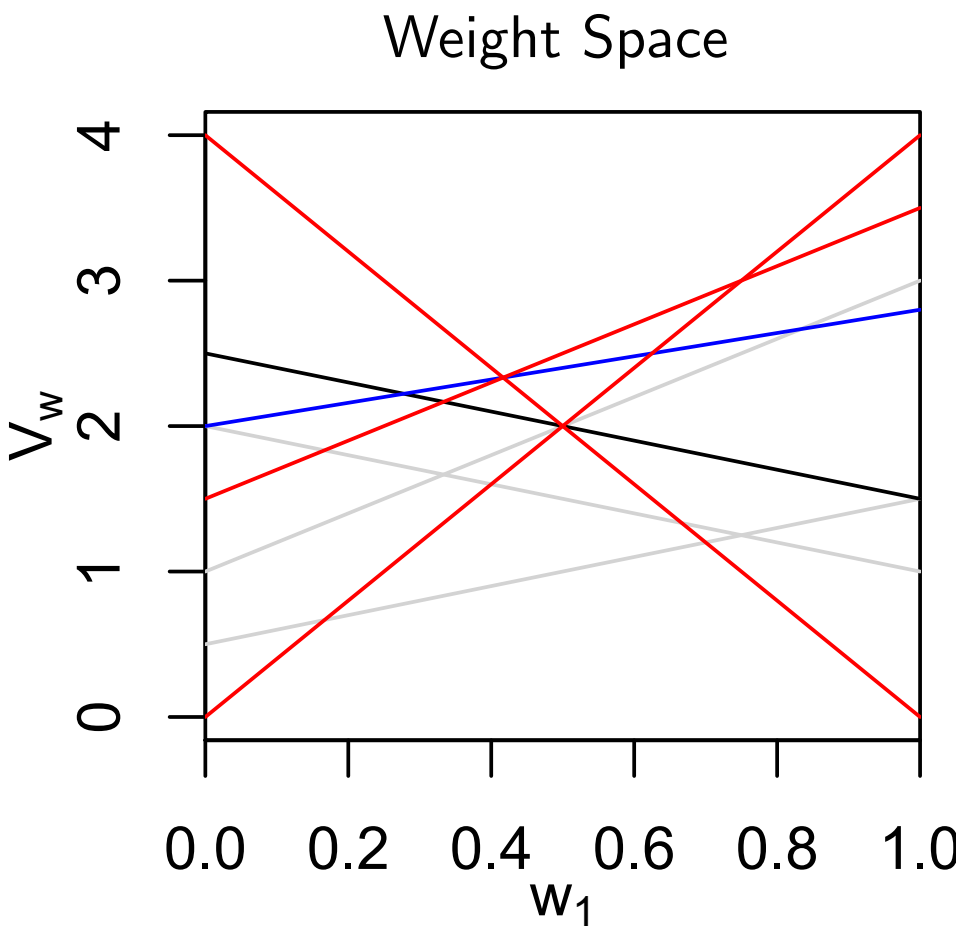
Visualization





Visualization





Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	*Pareto coverage set of deterministic non-stationary policies*	convex coverage set of deterministic stationary policies

Example: radiation vs. chemotherapy (again)

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	*Pareto coverage set of deterministic non-stationary policies*	convex coverage set of deterministic stationary policies

Example: radiation vs. chemotherapy (again)

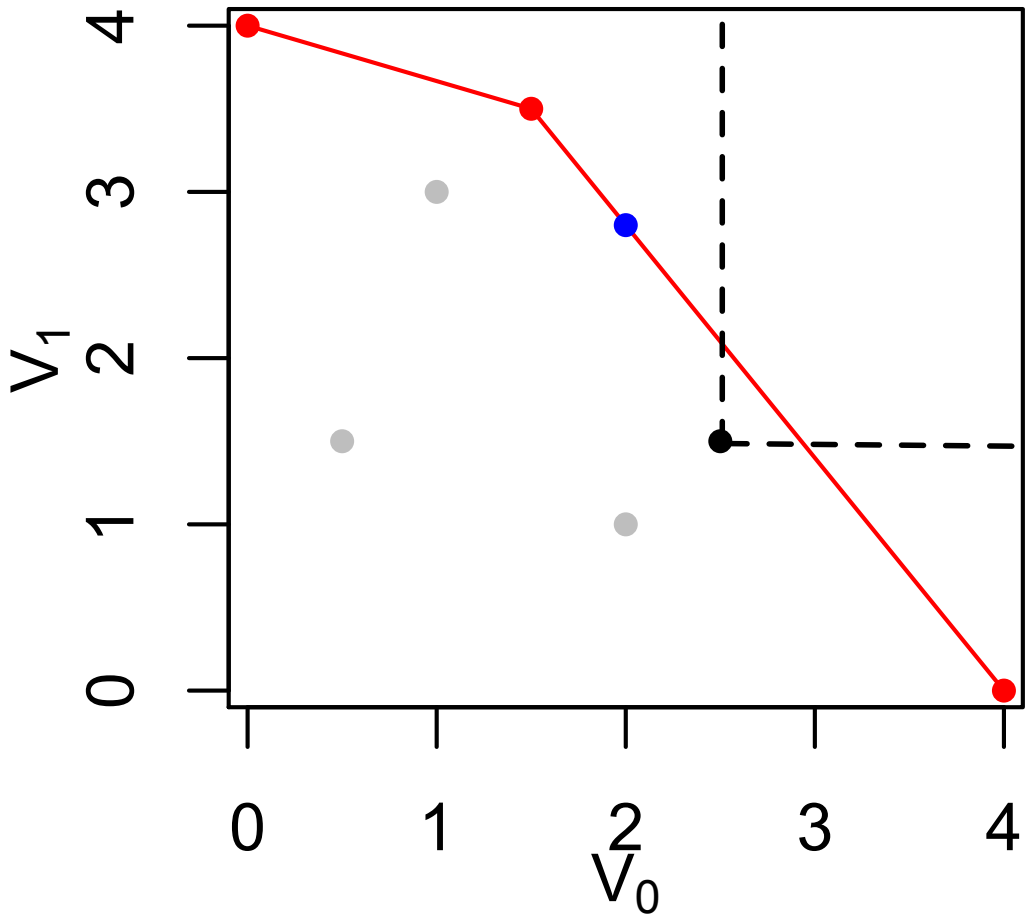
Note: only setting that case requires a Pareto front!

Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	*convex coverage set of deterministic stationary policies*

Mixture Policies

- A $CCS(\Pi_{DS})$ is also a $CCS(\Pi)$ but not necessarily a $PCS(\Pi)$
- But a $PCS(\Pi)$ can be made by mixing policies in a $CCS(\Pi_{DS})$



Problem Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	*convex coverage set of deterministic stationary policies*

Example: studying vs. networking (again)

Part 2: Methods and Applications

- Convex Coverage Set Planning Methods
  - Inner Loop: Convex Hull Value Iteration
  - Outer Loop: Optimistic Linear Support
- Pareto Coverage Set Planning Methods
  - Inner loop (non-stationary): Pareto-Q
  - Outer loop issues
- MOPOMDP Convex Coverage Set Planning: OLSAR
- Applications

Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	*one deterministic stationary policy*		*convex coverage set of deterministic stationary policies*	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	*Pareto coverage set of deterministic non-stationary policies*	*convex coverage set of deterministic stationary policies*

Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	*one deterministic stationary policy*		*convex coverage set of deterministic stationary policies*	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	*Pareto coverage set of deterministic non-stationary policies*	*convex coverage set of deterministic stationary policies*

- Known transition and reward functions $\rightarrow$ *planning*

Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	*one deterministic stationary policy*		*convex coverage set of deterministic stationary policies*	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	*Pareto coverage set of deterministic non-stationary policies*	*convex coverage set of deterministic stationary policies*

- Known transition and reward functions $\rightarrow$ *planning*
- Unknown transition and reward functions $\rightarrow$ learning

Background: Value Iteration

- Initial estimate value estimate $V_0(s)$
- Apply *Bellman backups* until convergence:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \Big[R(s,a,s') + \gamma V_k(s') \Big]$$

Background: Value Iteration

- Initial estimate value estimate $V_0(s)$
- Apply *Bellman backups* until convergence:

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \Big[R(s,a,s') + \gamma V_k(s') \Big]$$

- Can also be written:

$$V_{k+1}(s) \leftarrow \max_a Q_{k+1}(s,a),$$
$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \Big[R(s,a,s') + \gamma V_k(s') \Big]$$

- Optimal policy is easy to retrieve from Q-table

Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	*one deterministic stationary policy*		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	convex coverage set of deterministic stationary policies

Scalarize MOMDP + Value Iteration

- For known $\mathbf{w}$

$$V^{\pi}_{\mathbf{w}} = \mathbf{w} \cdot E[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{t+k+1}] = E[\sum_{k=0}^{\infty} \gamma^k (\mathbf{w} \cdot \mathbf{r}_{t+k+1})].$$

Scalarize MOMDP + Value Iteration

- For known $\mathbf{w}$

$$V^{\pi}_{\mathbf{w}} = \mathbf{w} \cdot E[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{t+k+1}] = E[\sum_{k=0}^{\infty} \gamma^k (\mathbf{w} \cdot \mathbf{r}_{t+k+1})].$$

- Scalarize reward function of MOMDP

$$R_{\mathbf{w}} = \mathbf{w} \cdot \mathbf{R}$$

Scalarize MOMDP + Value Iteration

- For known $\mathbf{w}$

$$V^{\pi}_{\mathbf{w}} = \mathbf{w} \cdot E[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{t+k+1}] = E[\sum_{k=0}^{\infty} \gamma^k (\mathbf{w} \cdot \mathbf{r}_{t+k+1})].$$

- Scalarize reward function of MOMDP

$$R_{\mathbf{w}} = \mathbf{w} \cdot \mathbf{R}$$

- Apply standard VI

Scalarize MOMDP + Value Iteration

- For known $\mathbf{w}$

$$V^{\pi}_{\mathbf{w}} = \mathbf{w} \cdot E[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{t+k+1}] = E[\sum_{k=0}^{\infty} \gamma^k (\mathbf{w} \cdot \mathbf{r}_{t+k+1})].$$

- Scalarize reward function of MOMDP

$$R_{\mathbf{w}} = \mathbf{w} \cdot \mathbf{R}$$

- Apply standard VI
- Does not return multi-objective value

Scalarized Value Iteration

- Adapt Bellman backup:

$$\mathbf{w} \cdot \mathbf{V}_{k+1}(s) \leftarrow \max_a \mathbf{w} \cdot \mathbf{Q}_{k+1}(s,a),$$

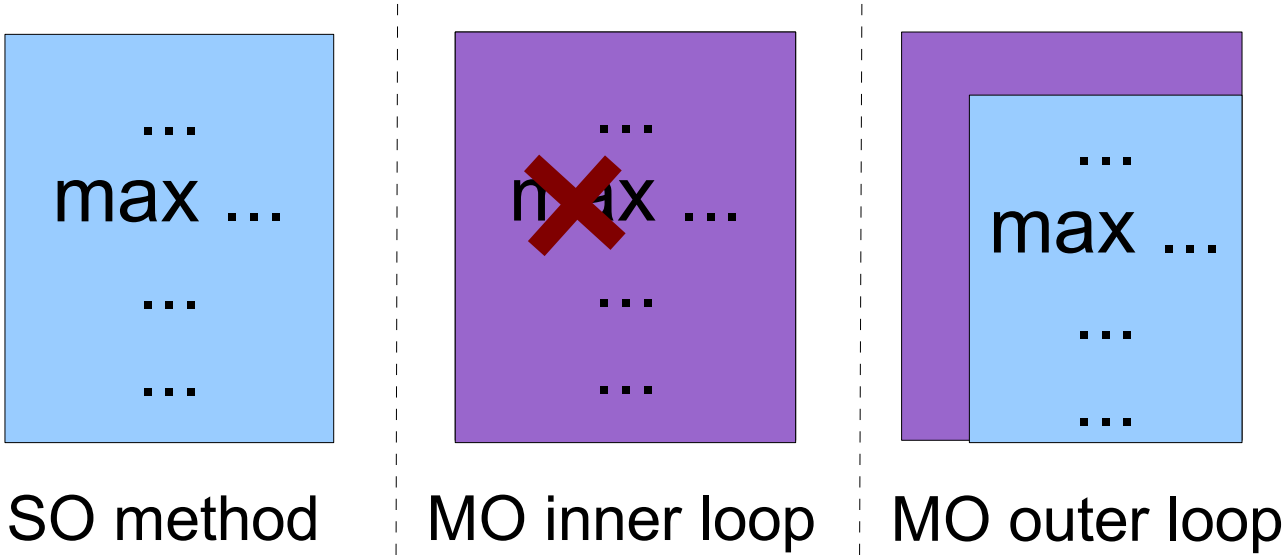
$$\mathbf{Q}_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \Big[\mathbf{R}(s,a,s') + \gamma \mathbf{V}_k(s') \Big]$$

- Returns multi-objective value.

Taxonomy

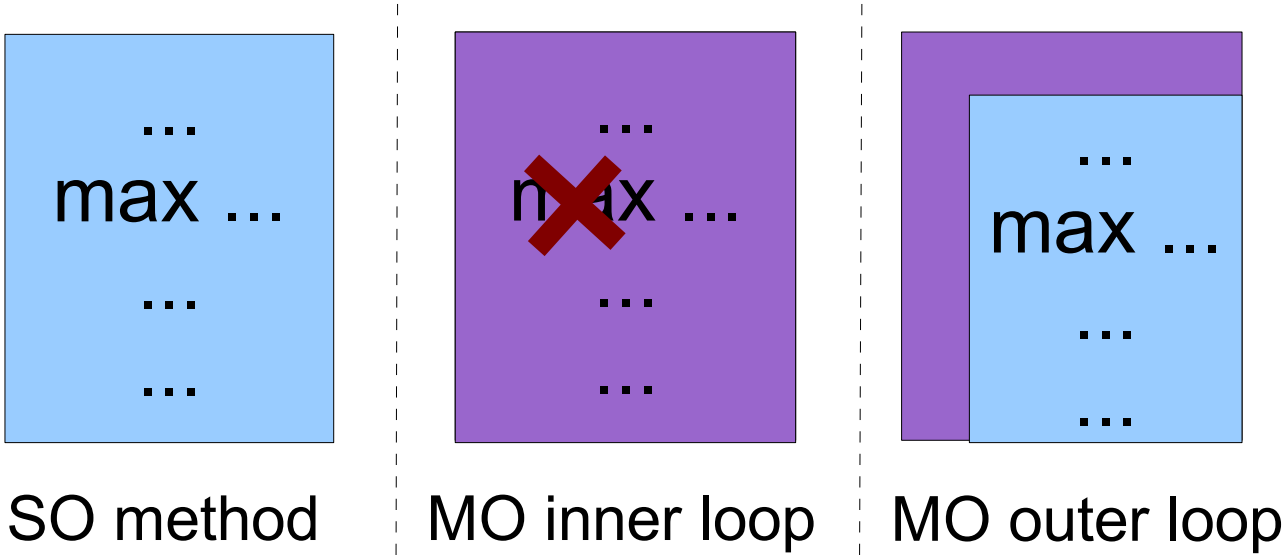
	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		*convex coverage set of deterministic stationary policies*	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	Pareto coverage set of deterministic non-stationary policies	*convex coverage set of deterministic stationary policies*

Inner versus Outer Loop



- Inner loop
  - Adapting operators of single objective method (e.g., value iteration)
  - Series of multi-objective operations (e.g. Bellman backups)

Inner versus Outer Loop



- Inner loop
  - Adapting operators of single objective method (e.g., value iteration)
  - Series of multi-objective operations (e.g. Bellman backups)
- Outer loop
  - Single objective method as subroutine
  - Series of single-objective problems

Inner Loop: Convex Hull Value Iteration

- Barrett & Narayanan (2008)
- Idea: do the backup for all $\mathbf{w}$ in parallel
- New backup operators must handle *sets of values*.
- At backup:
  - generate all value vectors for s, a-pair
  - *prune* away those that are not optimal for any $\mathbf{w}$
- Only need deterministic stationary policies

Inner Loop: Convex Hull Value Iteration

- Initial *set* of *value vectors*, e.g., $V_0(s) = \{(0,0)\}$
- All possible value vectors:

$$\mathbf{Q}_{k+1}(s,a) \leftarrow \bigoplus_{s'} T(s,a,s') \left[\mathbf{R}(s,a,s') + \gamma \mathbf{V}_k(s')\right]$$

where $\mathbf{u} + V = \{\mathbf{u} + \mathbf{v} : \mathbf{v} \in V\}$, and

$U \oplus V = \{\mathbf{u} + \mathbf{v} : \mathbf{u} \in U \wedge \mathbf{v} \in V\}$

Inner Loop: Convex Hull Value Iteration

- Initial *set* of *value vectors*, e.g., $V_0(s) = \{(0,0)\}$
- All possible value vectors:

$$\mathbf{Q}_{k+1}(s,a) \leftarrow \bigoplus_{s'} T(s,a,s') \left[\mathbf{R}(s,a,s') + \gamma \mathbf{V}_k(s')\right]$$

where $\mathbf{u} + V = \{\mathbf{u} + \mathbf{v} : \mathbf{v} \in V\}$, and

$U \oplus V = \{\mathbf{u} + \mathbf{v} : \mathbf{u} \in U \wedge \mathbf{v} \in V\}$

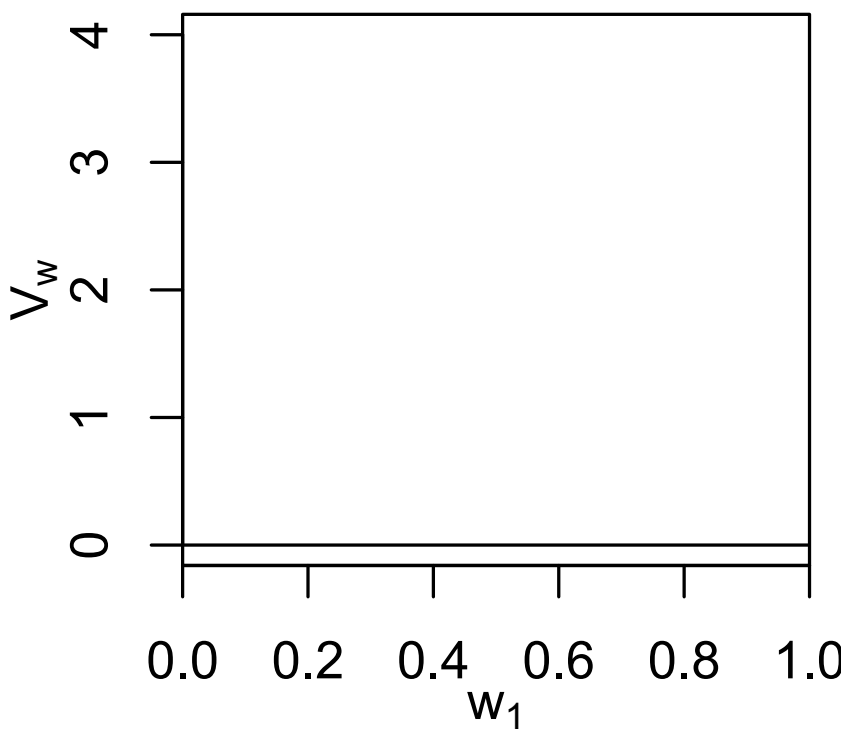
- *Prune* value vectors

$$\mathbf{V}_{k+1}(s) \leftarrow \texttt{CPrune}\left(\bigcup_a \mathbf{Q}_{k+1}(s,a)\right)$$

- CPrune uses *linear programs* (e.g., Roijers et al. (2015))

CHVI Example

- Extremely simple MOMDP:
  1 state: s;
  2 actions: a_1 and a_2
- Deterministic transitions
- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- $\mathbf{V}_0(s) = \{(0, 0)\}$



CHVI Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 1:
  $\mathbf{V}_0(s) = \{(0, 0)\}$

CHVI Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 1:
  $\mathbf{V}_0(s) = \{(0, 0)\}$

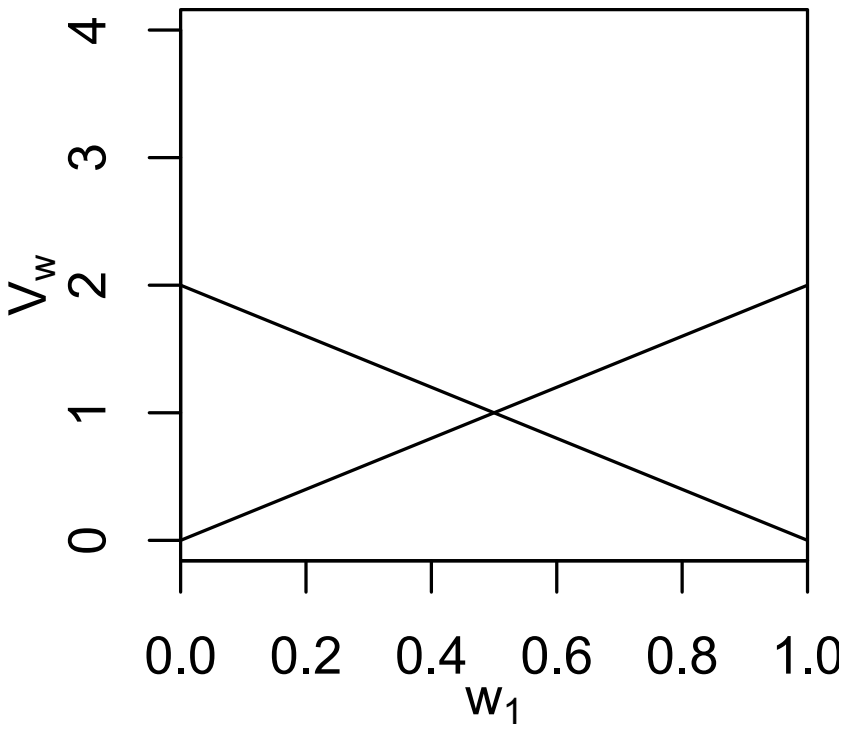
  $\mathbf{Q}_1(s, a_1) = \{(2, 0)\}$
  $\mathbf{Q}_1(s, a_2) = \{(0, 2)\}$

CHVI Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 1:
  $\mathbf{V}_0(s) = \{(0, 0)\}$

  $\mathbf{Q}_1(s, a_1) = \{(2, 0)\}$
  $\mathbf{Q}_1(s, a_2) = \{(0, 2)\}$

  $\mathbf{V}_1(s) = \texttt{CPrune}(\bigcup_a \mathbf{Q}_1(s, a)) =$
  $\{(2, 0), (0, 2)\}$



CHVI Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 2:
  $\mathbf{V}_1(s) = \{(2, 0), (0, 2)\}$

CHVI Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 2:
  $\mathbf{V}_1(s) = \{(2, 0), (0, 2)\}$

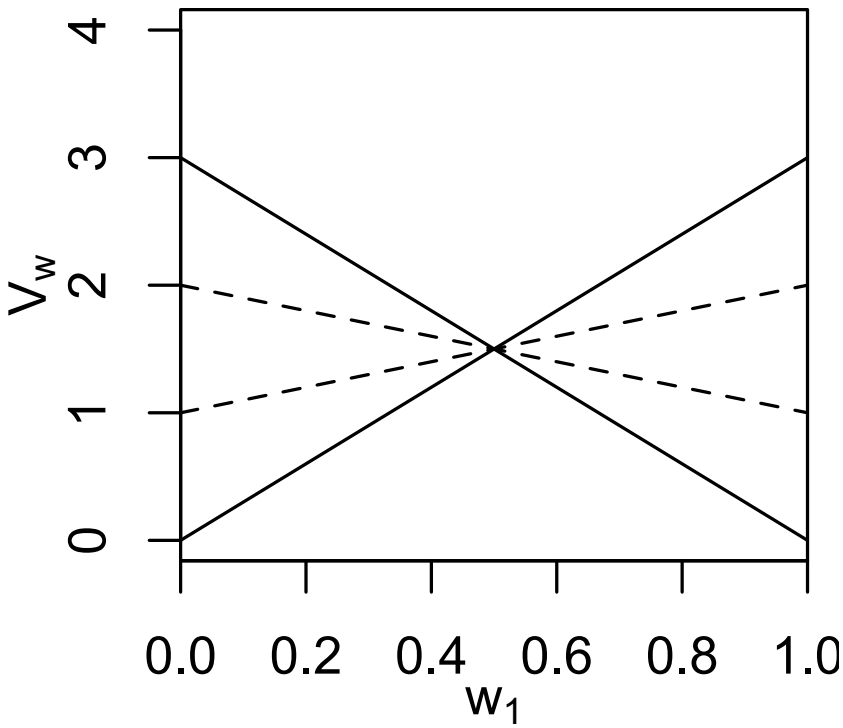
  $\mathbf{Q}_2(s, a_1) = \{(3, 0), (2, 1)\}$
  $\mathbf{Q}_2(s, a_2) = \{(1, 2), (0, 3)\}$

CHVI Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 2:
  $\mathbf{V}_1(s) = \{(2, 0), (0, 2)\}$

  $\mathbf{Q}_2(s, a_1) = \{(3, 0), (2, 1)\}$
  $\mathbf{Q}_2(s, a_2) = \{(1, 2), (0, 3)\}$

  $\mathbf{V}_2(s) =$
  $\texttt{CPrune}(\{(3, 0), (2, 1), (1, 2), (0, 3)\})$

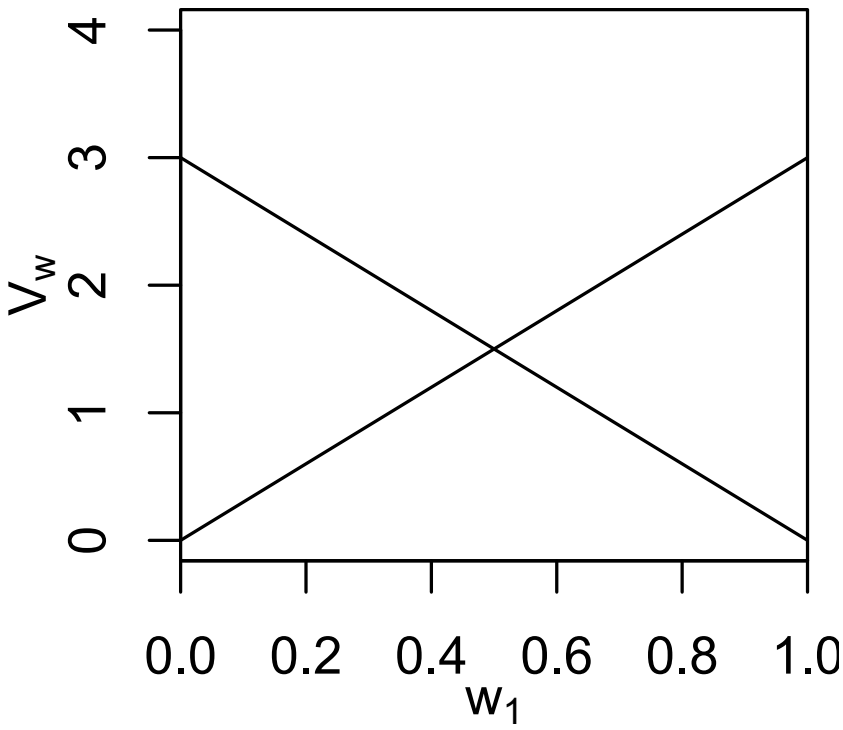


CHVI Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 2:
  $\mathbf{V}_1(s) = \{(2, 0), (0, 2)\}$

  $\mathbf{Q}_2(s, a_1) = \{(3, 0), (2, 1)\}$
  $\mathbf{Q}_2(s, a_2) = \{(1, 2), (0, 3)\}$

  $\mathbf{V}_2(s) =$
  $\{(3, 0), (0, 3)\}$

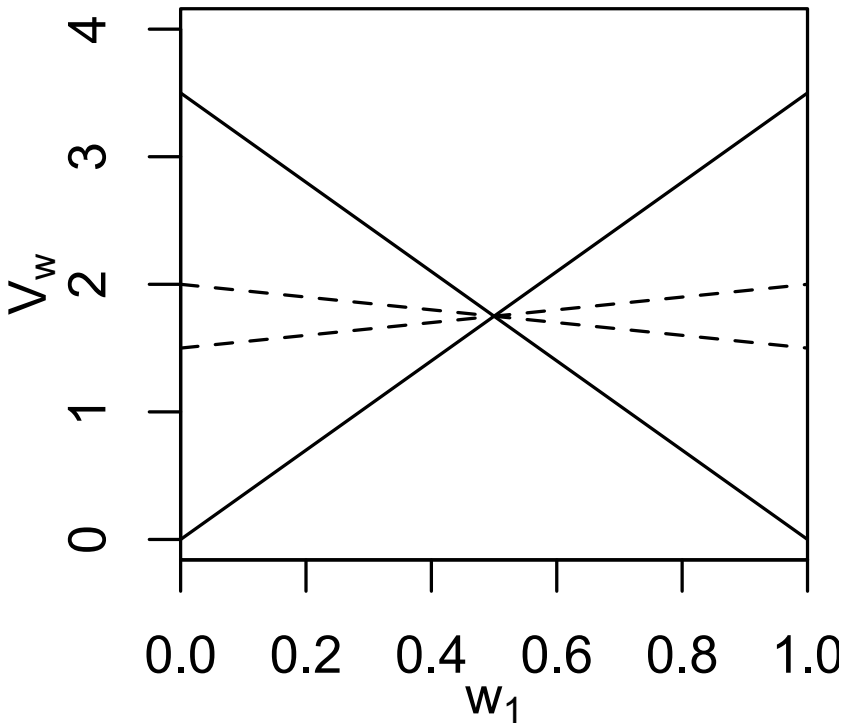


CHVI Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 3:
  $\mathbf{V}_2(s) = \{(3, 0), (0, 3)\}$

  $\mathbf{Q}_3(s, a_1) = \{(3.5, 0), (2, 1.5)\}$
  $\mathbf{Q}_3(s, a_2) = \{(1.5, 2), (0, 3.5)\}$

  $\mathbf{V}_3(s) =$
  $\texttt{CPrune}(\{(3.5, 0), (2, 1.5), (1.5, 2), (0, 3.5)\}) =$
  $\{(3.5, 0), (0, 3.5)\}$



Convex Hull Value Iteration

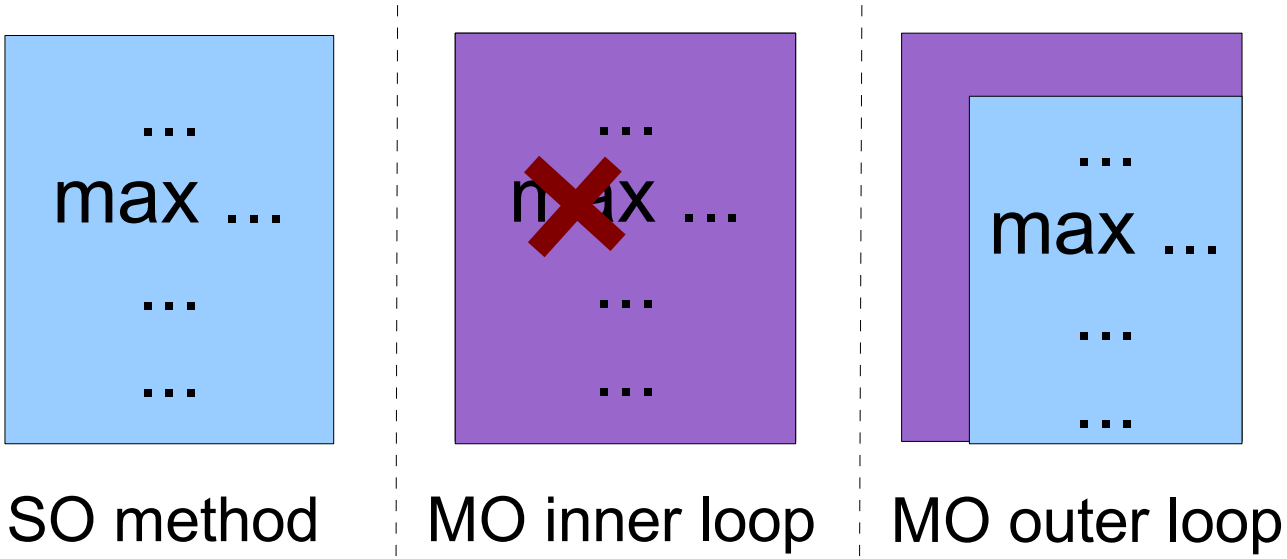
- `CPrune` retains at least one optimal vector for each $\mathbf{w}$
- Therefore, $V_{\mathbf{w}}$ that would have been computed by VI is kept
- CHVI does not retain excess value vectors

Convex Hull Value Iteration

- `CPrune` retains at least one optimal vector for each $\mathbf{w}$
- Therefore, $V_{\mathbf{w}}$ that would have been computed by VI is kept
- CHVI does not retain excess value vectors

- CHVI *generates* a lot of excess value vectors
- Removal with linear programs (`CPrune`) is expensive

Outer Loop



- Repeatly calls a single-objective solver
- Generic multi-objective method
  - multi-objective coordination graphs
  - multi-objective (multi-agent) MDPs
  - multi-objective partially observable MDPs

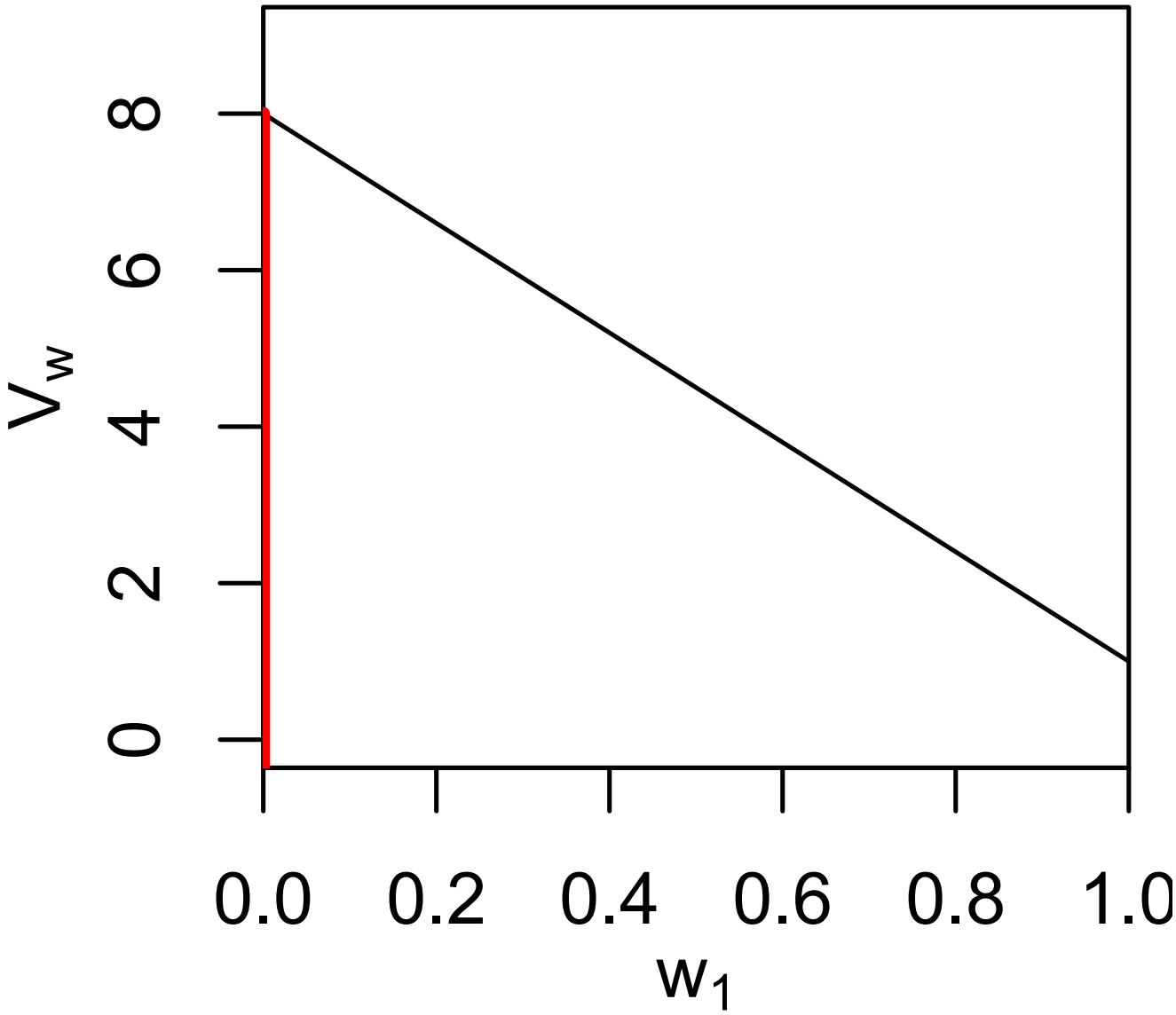
Outer Loop: Optimistic Linear Support

- *Optimistic linear support* (OLS) adapts and improves linear support for POMDPs (Cheng (1988))
- Solves *scalarized* instances for specific **w**

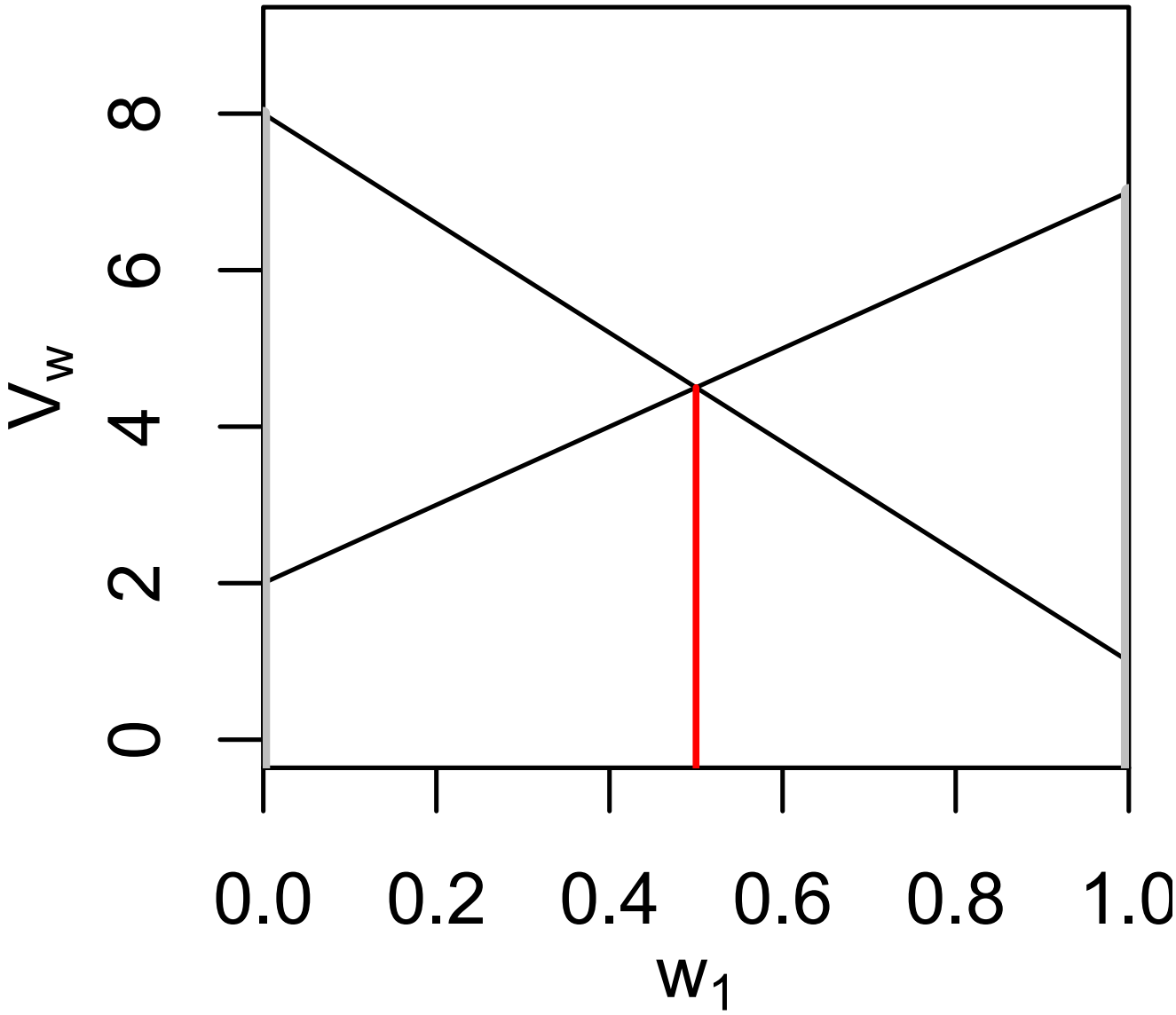
Outer Loop: Optimistic Linear Support

- *Optimistic linear support* (OLS) adapts and improves linear support for POMDPs (Cheng (1988))
- Solves *scalarized* instances for specific **w**
- Terminates after checking only a finite number of weights
- Returns exact CCS

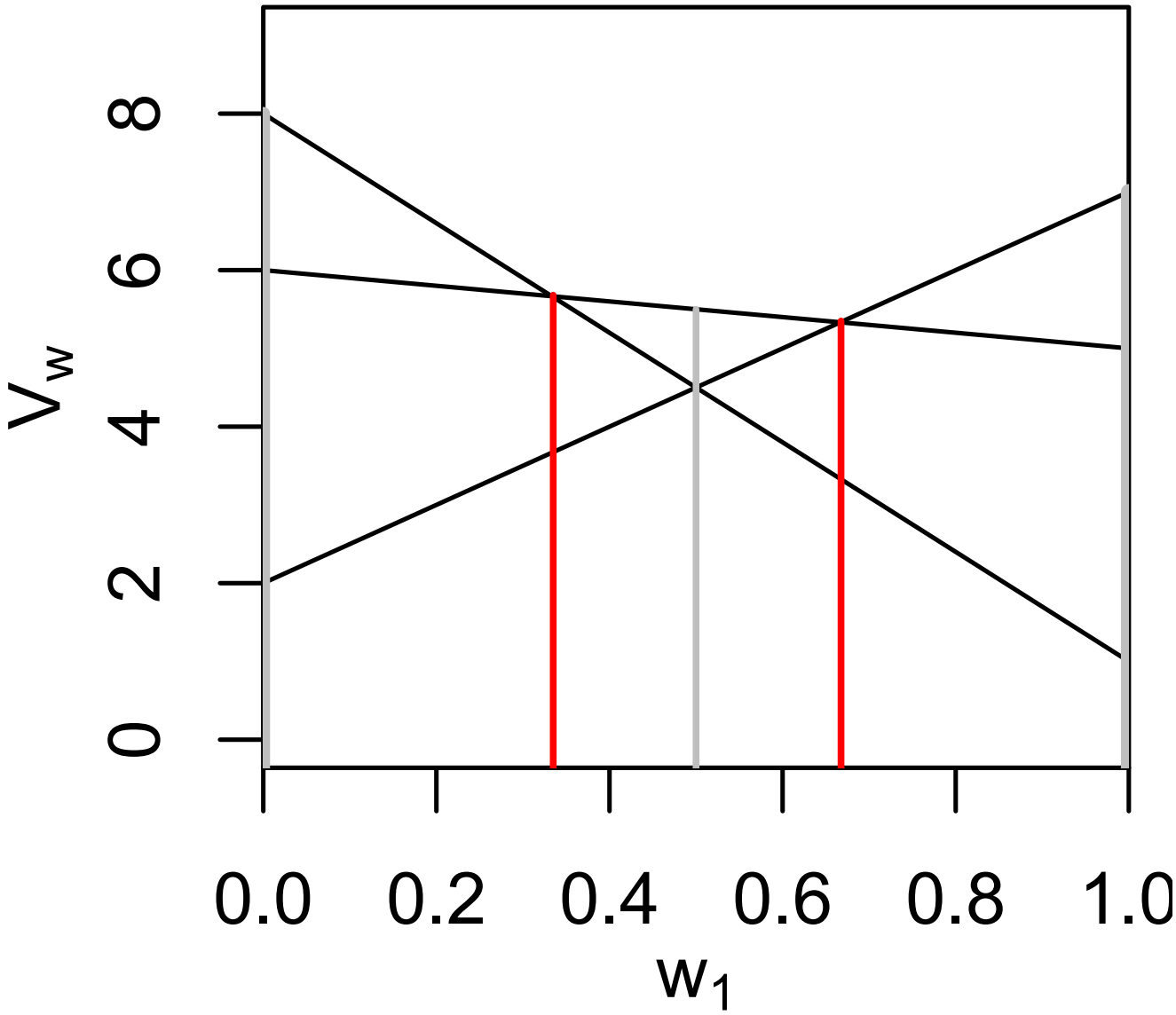
Linear Support



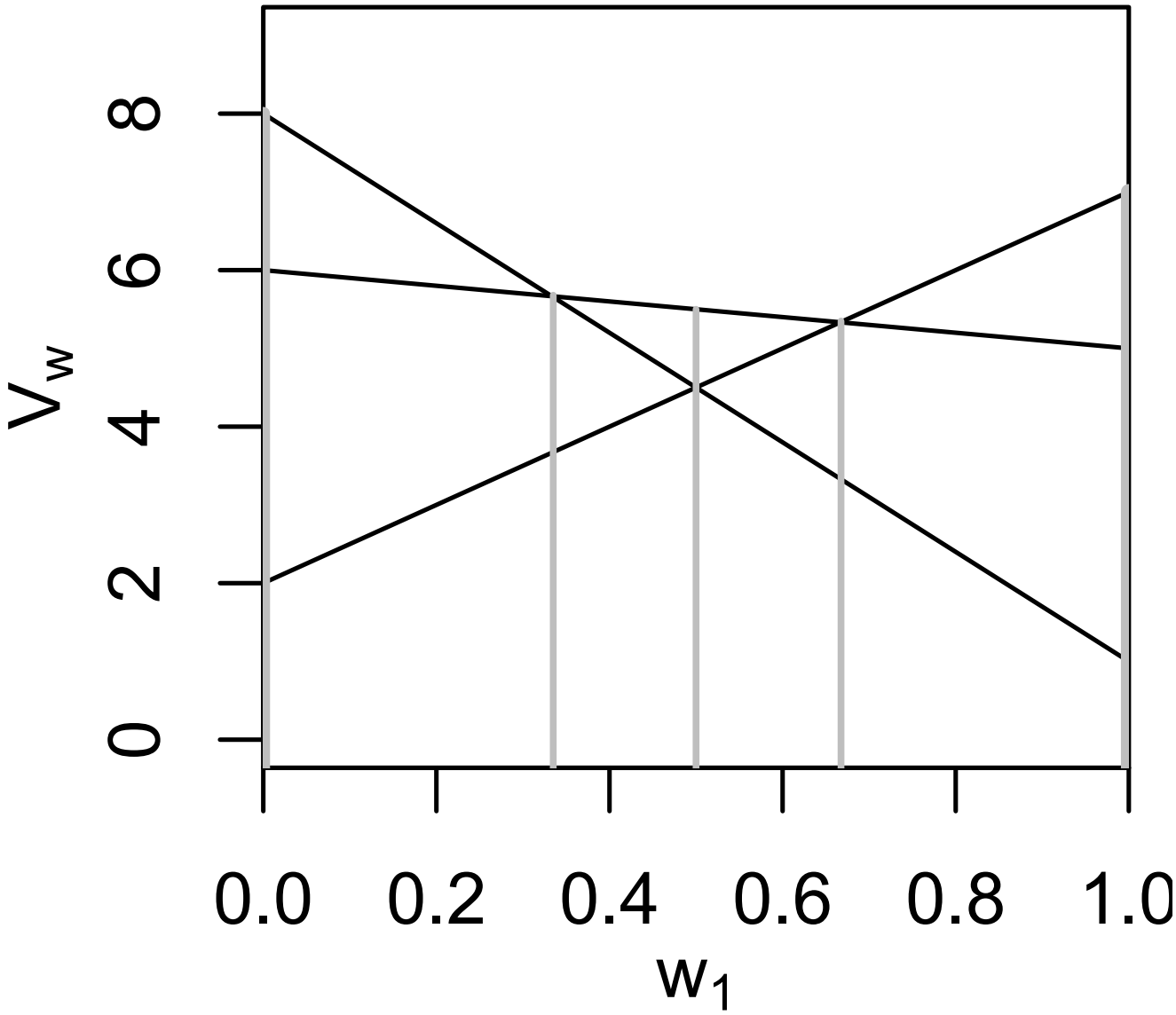
Linear Support



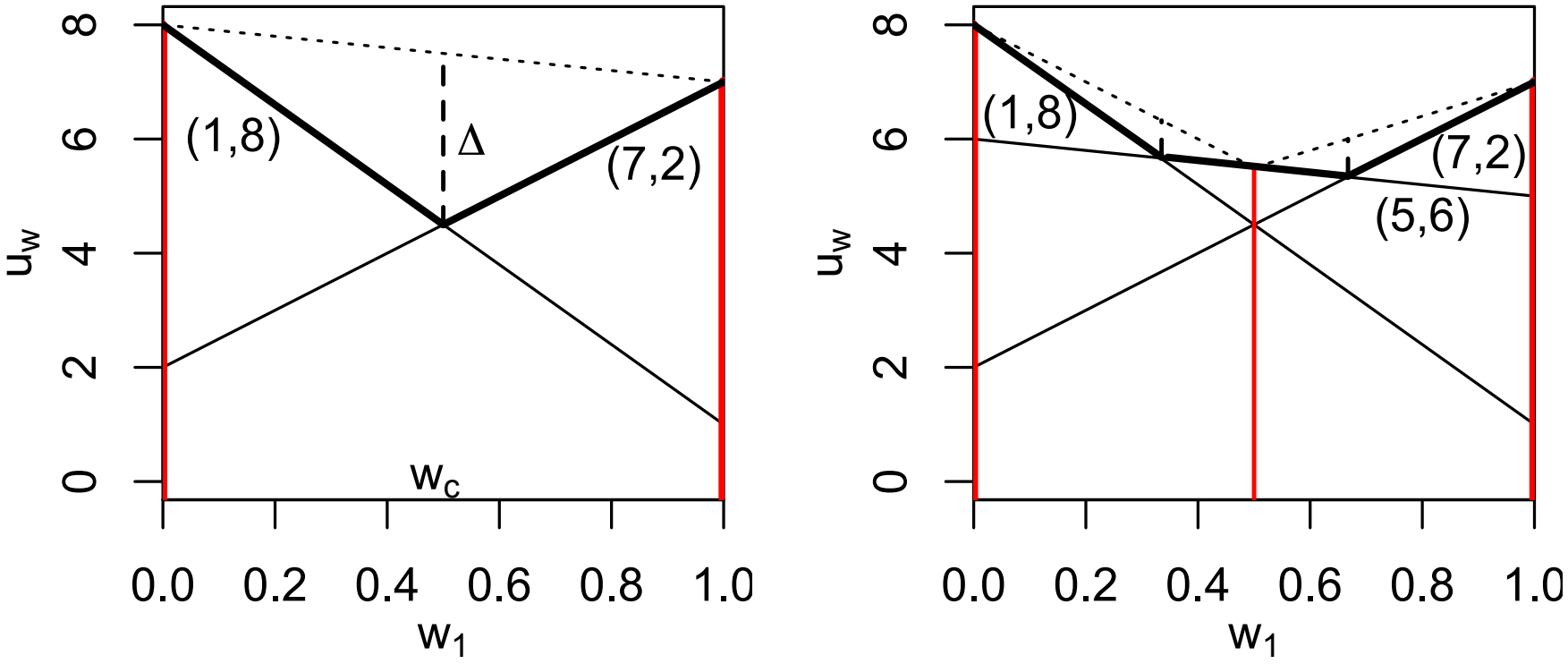
Linear Support



Linear Support

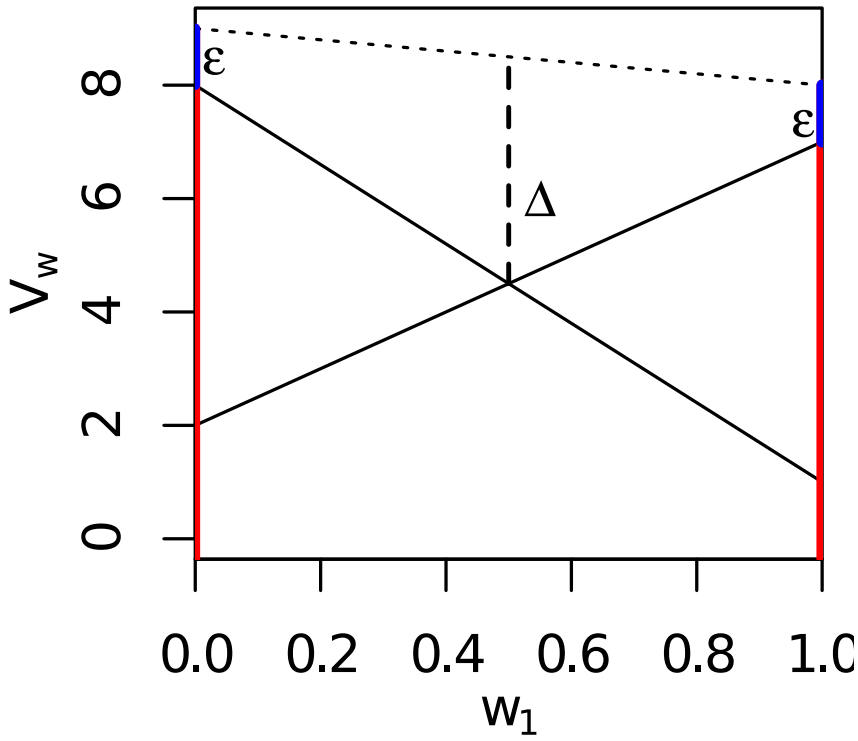


Optimistic Linear Support



- Priority queue, Q, for corner weights
- Maximal possible improvement Δ as priority
- Stop when $\Delta < \varepsilon$

Optimistic Linear Support



- Solving scalarized instance not always possible
- ε-approximate solver
- Produces an ε-CCS

Comparing Inner and Outer Loop

- OLS (outer loop) advantages
  - Any (cooperative) multi-objective decision problem
  - Any single-objective / scalarized subroutine
  - Inherits quality guarantees
  - Faster for small and medium numbers of objectives
- Inner loop faster for large numbers of objectives

Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	*Pareto coverage set of deterministic non-stationary policies*	convex coverage set of deterministic stationary policies

Inner Loop: Pareto-Q

- Similar to CHVI
- Different pruning operator
- Pairwise comparisons: $\mathbf{V}(s) \succ_P \mathbf{V}'(s)$
- Comparisons cheaper but much more vectors
- Converges to correct Pareto coverage set (White (1982))
- Executing a policy is no longer trivial (Van Moffaert & Nowé (2014))

Inner Loop: Pareto-Q

- Compute all possible vectors

$$\mathbf{Q}_{k+1}(s,a) \leftarrow \bigoplus_{s'} T(s,a,s') \left[\mathbf{R}(s,a,s') + \gamma \mathbf{V}_k(s')\right]$$

where $\mathbf{u} + V = \{\mathbf{u} + \mathbf{v} : \mathbf{v} \in V\}$,
$U \oplus V = \{\mathbf{u} + \mathbf{v} : \mathbf{u} \in U \wedge \mathbf{v} \in V\}$

Inner Loop: Pareto-Q

- Compute all possible vectors

$$\mathbf{Q}_{k+1}(s,a) \leftarrow \bigoplus_{s'} T(s,a,s') \left[\mathbf{R}(s,a,s') + \gamma \mathbf{V}_k(s')\right]$$

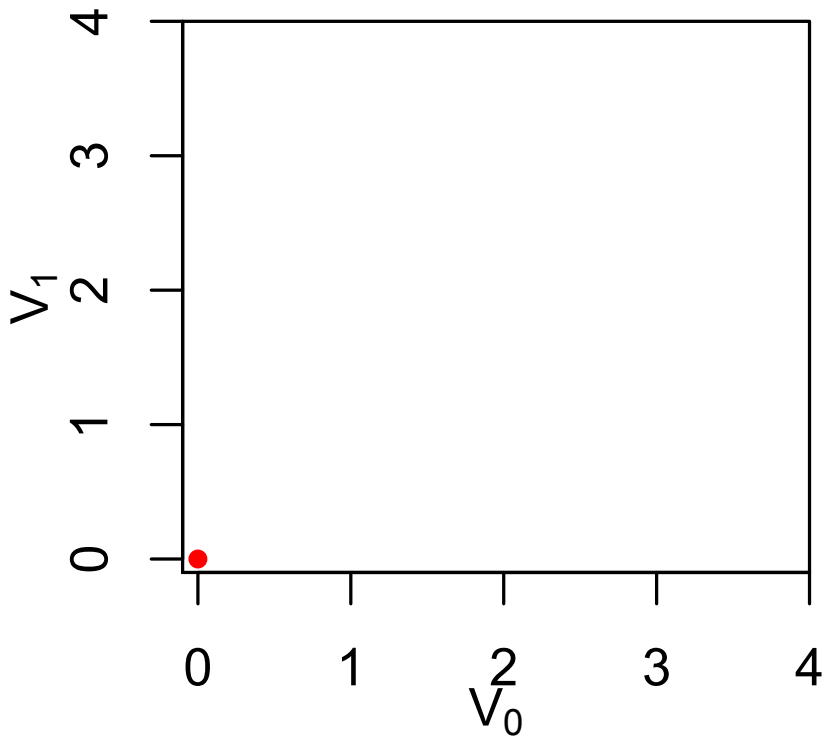
where $\mathbf{u} + V = \{\mathbf{u} + \mathbf{v} : \mathbf{v} \in V\}$,
$U \oplus V = \{\mathbf{u} + \mathbf{v} : \mathbf{u} \in U \wedge \mathbf{v} \in V\}$

- Take the union across a
- Prune Pareto-dominated vectors

$$\mathbf{V}_{k+1}(s) \leftarrow \texttt{PPrune}\left(\bigcup_a \mathbf{Q}_{k+1}(s,a)\right)$$

Pareto-Q Example

- Extremely simple MOMDP:
  1 state: s;
  2 actions: a_1 and a_2
- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- $\mathbf{V}_0(s) = \{(0, 0)\}$



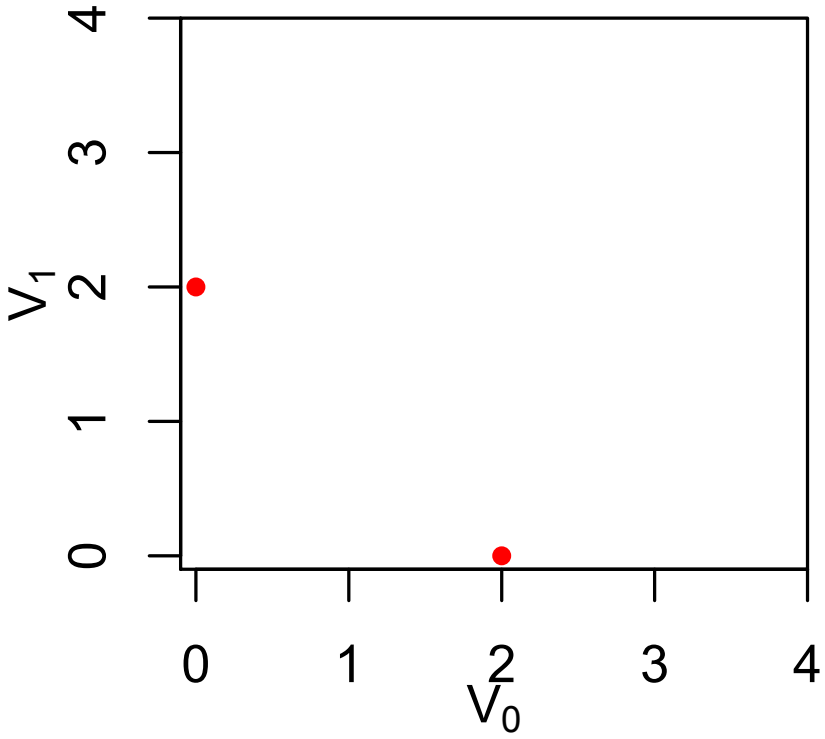
Pareto-Q Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 1:

  $\mathbf{V}_0(s) = \{(0, 0)\}$

  $\mathbf{Q}_1(s, a_1) = \{(2, 0)\}$
  $\mathbf{Q}_1(s, a_2) = \{(0, 2)\}$

  $\mathbf{V}_1(s) = \texttt{PPrune}(\bigcup_a \mathbf{Q}_1(s, a)) =$
  $\{(2, 0), (0, 2)\}$



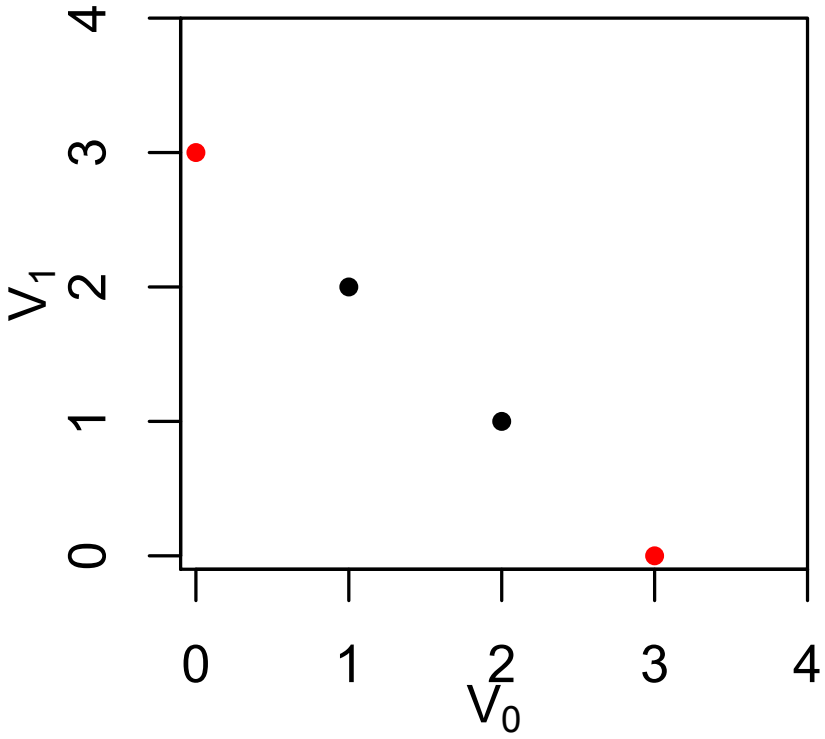
Pareto-Q Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 2:

  $\mathbf{V}_1(s) = \{(2, 0), (0, 2)\}$

  $\mathbf{Q}_2(s, a_1) = \{(3, 0), (2, 1)\}$
  $\mathbf{Q}_2(s, a_2) = \{(1, 2), (0, 3)\}$

  $\mathbf{V}_2(s) =$
  $\texttt{PPrune}(\{(3, 0), (2, 1), (1, 2), (0, 3)\})$



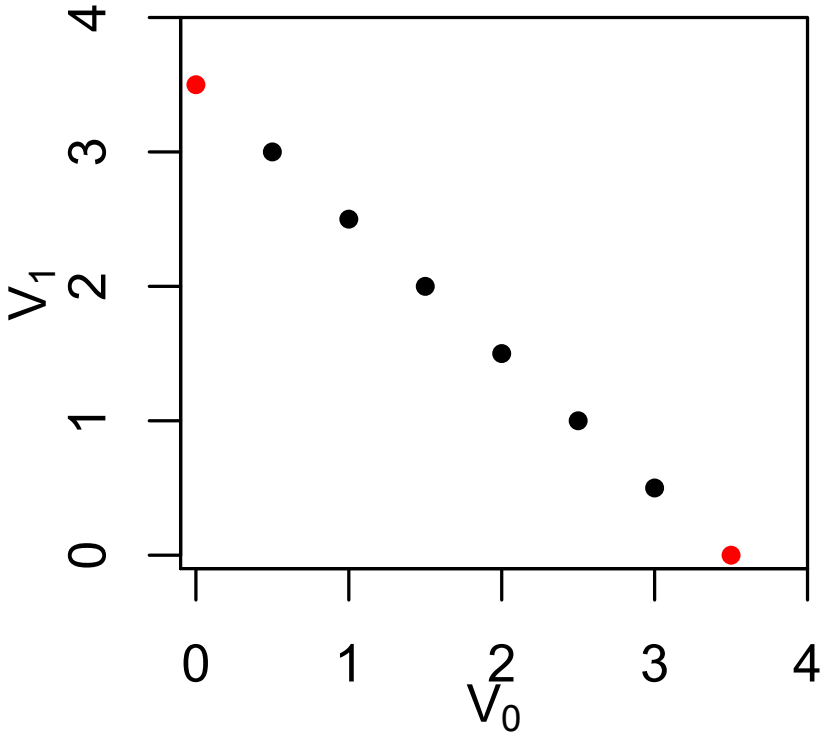
Pareto-Q Example

- Deterministic rewards:
  $\mathbf{R}(s, a_1, s) \rightarrow (2, 0)$
  $\mathbf{R}(s, a_2, s) \rightarrow (0, 2)$
- $\gamma = 0.5$
- Iteration 2:

  $\mathbf{V}_2(s) =$
  $\{(3, 0), (2, 1), (1, 2), (0, 3)\}$

  $\mathbf{Q}_3(s, a_1) =$
  $\{(3.5, 0), (3, 0.5), (2.5, 1), (2, 1.5)\}$
  $\mathbf{Q}_3(s, a_2) =$
  $\{(1.5, 2), (1, 2.5), (0.5, 3), (0, 3.5)\}$

  $\mathbf{V}_3(s) =$
  $\texttt{PPrune}(\{(3.5, 0), (3, 0.5), (2.5, 1), (2, 1.5),$
  $(1.5, 2), (1, 2.5), (0.5, 3), (0, 3.5)\})$



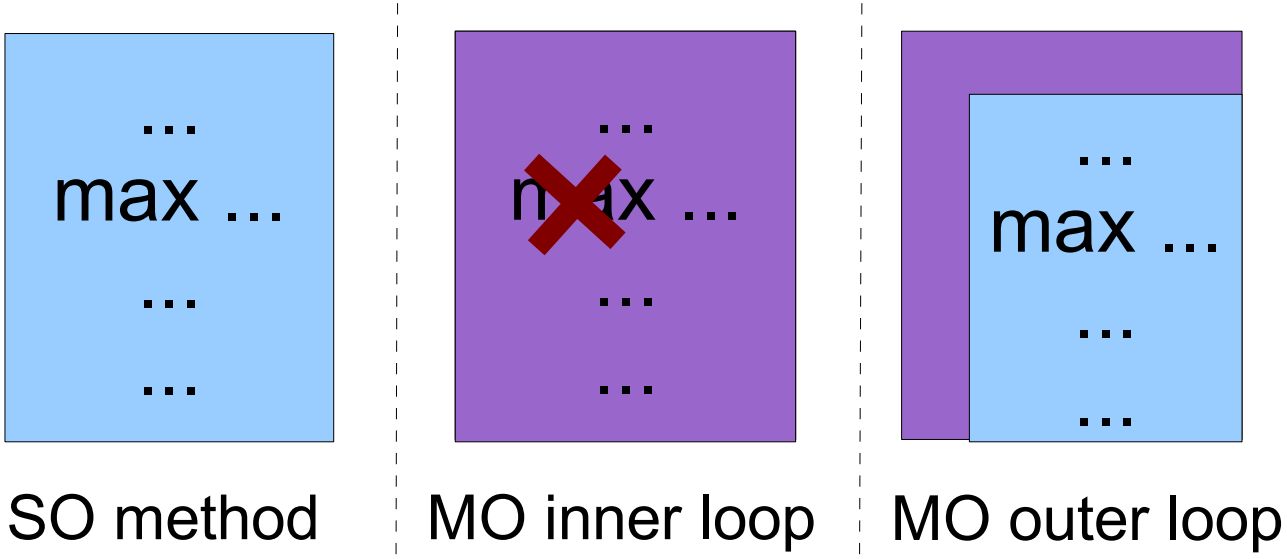
Inner Loop: Pareto-Q

- PCS size can explode
- No longer deterministic
- Cannot read policy from Q-table
- Except for first action

Inner Loop: Pareto-Q

- PCS size can explode
- No longer deterministic
- Cannot read policy from Q-table
- Except for first action
- "Track" a policy during execution (Van Moffaert & Nowé (2014))
  - For deterministic transitions: $s, a \rightarrow s'$
  - From $\mathbf{Q}_{t=0}(s, a)$ substract $\mathbf{R}(s, a)$
  - Correct for discount factor $\rightarrow \mathbf{V}_{t=1}(s')$
  - Find $\mathbf{V}_{t=1}(s')$ in Q-tables for s'
- For stochastic transitions, see Kristoff van Moffaert's PhD thesis

Outer Loop?



- Outer loop very difficult:

$$V^{\pi}_{\mathbf{w}} = f(E[\sum_{k=0}^{\infty} \gamma^k \mathbf{r}_{t+k+1}], \mathbf{w}) \neq E[\sum_{k=0}^{\infty} \gamma^k f(\mathbf{r}_{t+k+1}, \mathbf{w})]$$

- Maximization does not do the trick!
- Heuristic with non-linear f (Van Moffaert, Drugan, Nowé (2013))
- Not guaranteed to find optimal policy, or converge

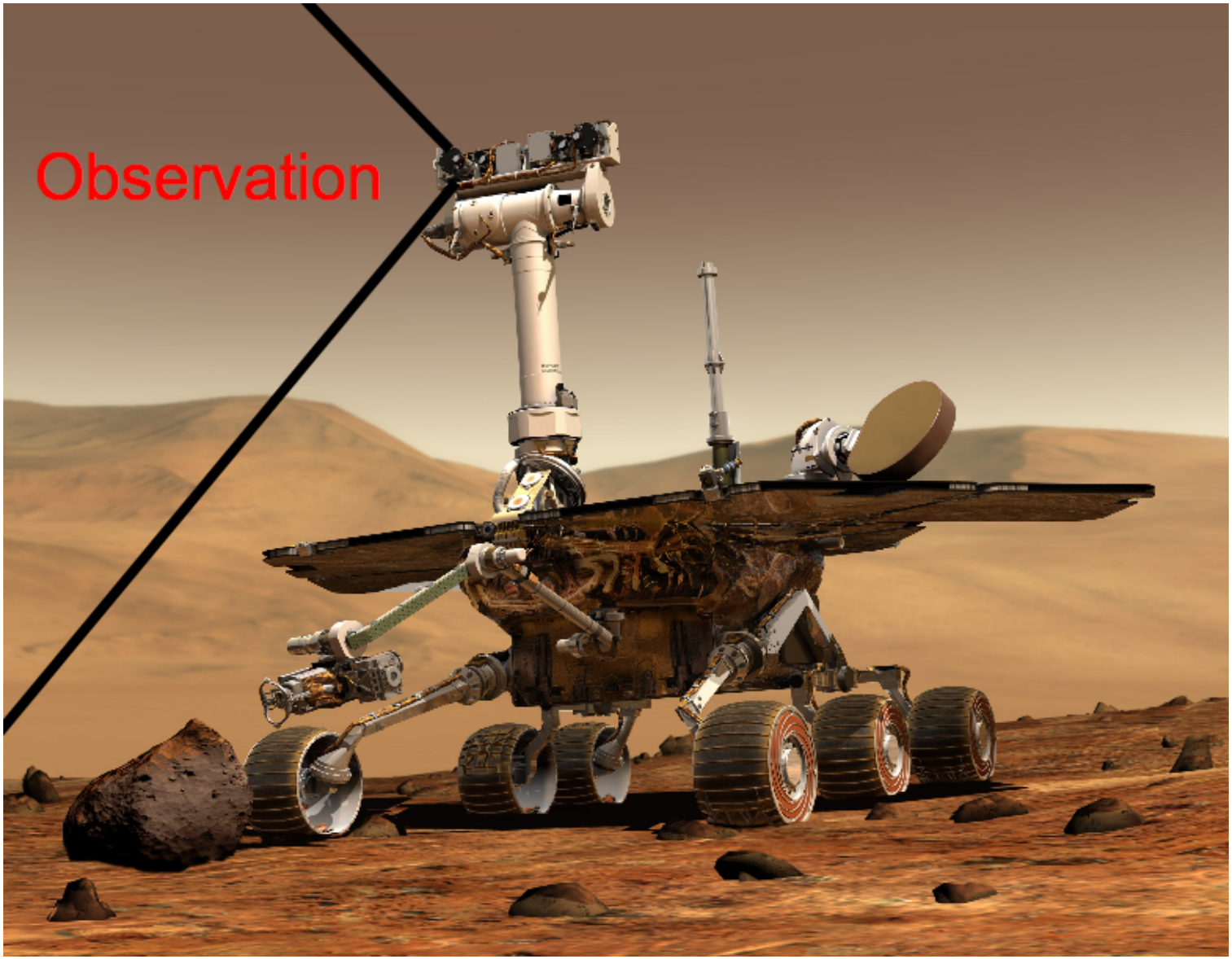
Taxonomy

	single policy (known weights)		*multiple policies (unknown weights or decision support)*	
	deterministic	stochastic	deterministic	stochastic
linear scalarization	one deterministic stationary policy		convex coverage set of deterministic stationary policies	
monotonically increasing scalarization	one deterministic non-stationary policy	one mixture policy of two or more deterministic stationary policies	*Pareto coverage set of deterministic non-stationary policies*	convex coverage set of deterministic stationary policies

Part 2: Methods and Applications

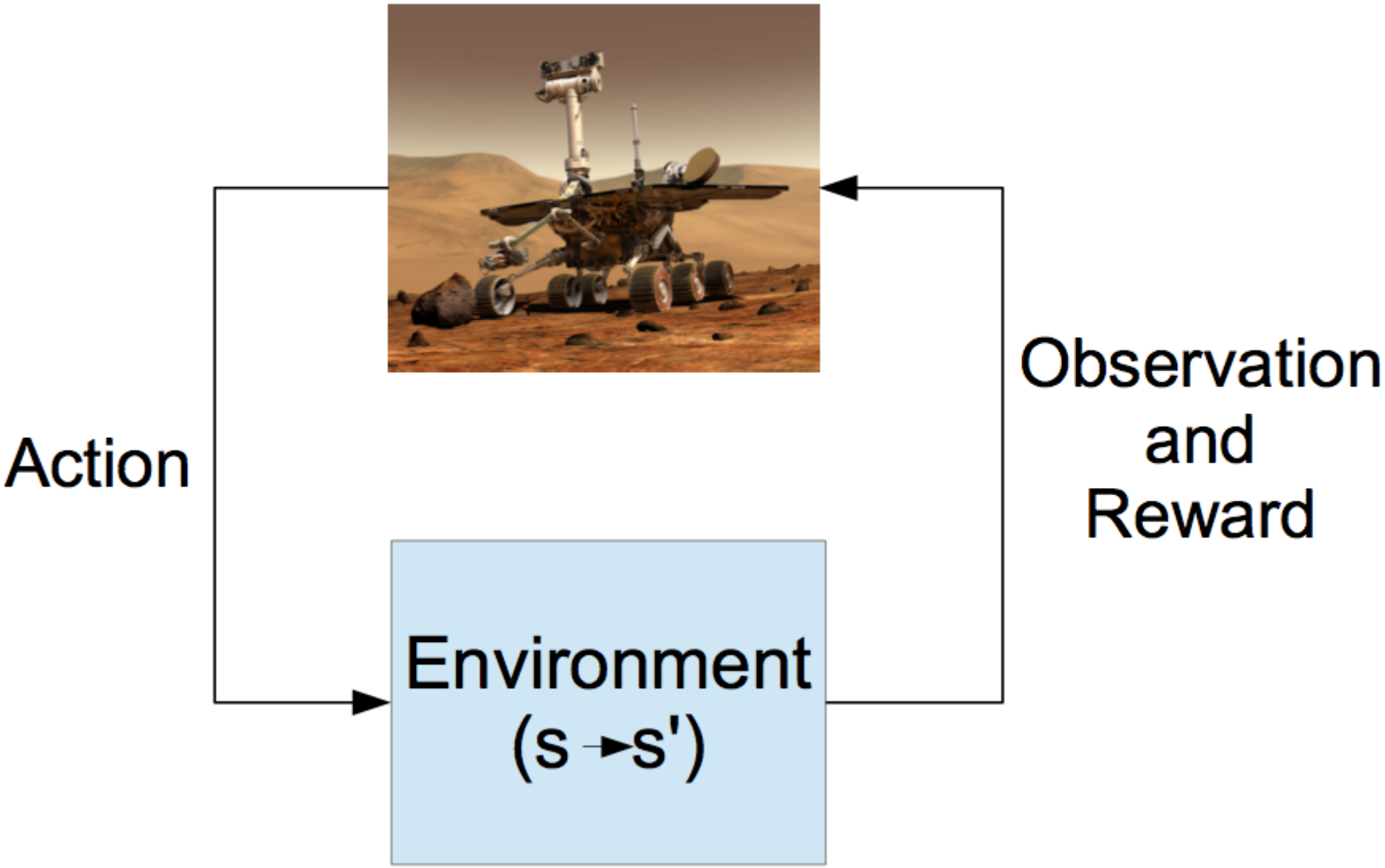
- Convex Coverage Set Planning Methods
  - Inner Loop: Convex Hull Value Iteration
  - Outer Loop: Optimistic Linear Support
- Pareto Coverage Set Planning Methods
  - Inner loop (non-stationary): Pareto-Q
  - Outer loop issues
- *MOPOMDP Convex Coverage Set Planning: OLSAR*
- Applications

Multiple objectives and partial observability



Maximize coverage while minimizing damage

Multi-objective Partially Observable MDPs



Partially Observable MDPs

A POMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \Omega, O \rangle$ where,

- $\mathcal{S}$, $\mathcal{A}$, T, and R are the same as in an MDP,
- Ω, is the set of possible *observations*, and
- O is the *observation function*: $\mathcal{A} \times \mathcal{S} \times \Omega \rightarrow [0, 1]$.

Partially Observable MDPs

Equivalent to *belief-MDP* $\langle \Delta\mathcal{S}, \mathcal{A}, T_b, R_b \rangle$:

- $\Delta\mathcal{S}$ is the *belief simplex* over $\mathcal{S}$,
- $\mathcal{A}$ is the same set of actions as in the POMDP
- $T_b(b, a, b')$ belief-transition function defined using Bayesian *belief updates* using b, a, and o:

$$b'(s') = \frac{O(o|s', a)}{P(o|b, a)} \sum_{s \in \mathcal{S}} T(s'|s, a) b(s),$$

- $R_b(b, a) = \sum_{s \in \mathcal{S}} b(s) R(s, a)$

Multi-Objective Partially Observable MDPs (MOPOMDPs)

- Vector-valued reward functions:
  - State-based: $\mathbf{R}(s, a)$
  - Belief-based: $\mathbf{R}_b(b, a)$
- Challenges:
  - How to represent the value?
  - Single-objective POMDP planning is expensive

Approach

- Starting from point-based planning for POMDPs
- Value representation for MOPOMDPs
- Scalarized point-based planning
- Point-based CCS planning
- OLS with Alpha Reuse

Value functions for POMDPs

- Point-based methods represent value by α-vectors

$$\alpha = \begin{pmatrix} V(s_1) \\ V(s_2) \\ V(s_3) \\ V(s_4) \end{pmatrix}$$

- $V^{\alpha}(b_0) = b_0 \cdot \alpha$

Scheme point-based planners

- Sampled set of beliefs B
- Set of α-vectors, $\mathcal{A}$
- Repeatly pick a $b \in B$
- Perform *point-based* backup (Bellman update for b only).
- Until $\mathcal{A}$ converges

Point-based backups

Back-projection of α-vectors $\alpha_i \in \mathcal{A}_k$:

$$g_i^{a,o}(s) = \sum_{s' \in S} O(a, s', o) T(s, a, s') \alpha_i(s')$$

Value of next belief-state for o, times probability of o

$$\alpha_{k+1}^{b,a} = r^a + \gamma \sum_{o \in \Omega} \arg\max_{g^{a,o}} b \cdot g^{a,o}$$

Action-values: $Q_{k+1}(b, a)$

$$\text{backup}(\mathcal{A}_k, b) = \arg\max_{\alpha_{k+1}^{b,a}} b \cdot \alpha_{k+1}^{b,a}$$

Maximization over action values: $V_{k+1}(b)$

Point-based planners for (MO)POMDPs

- The bottleneck: the number of back-projected vectors is: $|\mathcal{A}_k| \cdot |A| \cdot |\Omega|$
- $|\mathcal{A}_k|$ can be huge
- Multi-objective?
  - Known weights $\rightarrow$ scalarized
  - CCS version?
    - ⋆ Inner loop?

Point-based planners for (MO)POMDPs

- The bottleneck: the number of back-projected vectors is: $|\mathcal{A}_k| \cdot |A| \cdot |\Omega|$
- $|\mathcal{A}_k|$ can be huge
- Multi-objective?
  - Known weights $\rightarrow$ scalarized
  - CCS version?
    - ★ Inner loop? Will have way more $|\mathcal{A}_k|$
    - ★ Outer loop!
    - ★ Scalarized point-based planner as subroutine

Scalarized point-based planners for POMDPs

- Set of α-vectors $\rightarrow$ Set of α-matrices
- Repeatly pick a $b \in B$
- Perform *point-based* backup for given b *and* **w**
- Until $\mathcal{A}$ converges

Value functions for (MO)POMDPs

- Point-based methods represent value by α-vectors

$$\alpha = \begin{pmatrix} V(s_1) \\ V(s_2) \\ V(s_3) \\ V(s_4) \end{pmatrix}$$

- $V^{\alpha}(b_0) = b_0 \cdot \alpha$

Value functions for (MO)POMDPs

- Point-based methods represent value by α-vectors

$$\alpha = \begin{pmatrix} V(s_1) \\ V(s_2) \\ V(s_3) \\ V(s_4) \end{pmatrix}$$

- $V^{\alpha}(b_0) = b_0 \cdot \alpha$

- Adapt point-based methods to return α-matrices

$$A = \begin{array}{c} \begin{matrix} obj\ 1: & obj\ 2: \end{matrix} \\ \begin{pmatrix} V_1(s_1) & V_2(s_1) \\ V_1(s_2) & V_2(s_2) \\ V_1(s_3) & V_2(s_3) \\ V_1(s_4) & V_2(s_4) \end{pmatrix} \end{array}$$

- $\mathbf{V}^{A}(b_0) = b_0 A$

- Adapted point-based backups

(Scalarized) point-based backups

Back-projection of α-vectors $\alpha_i \in \mathcal{A}_k$:

$$g_i^{a,o}(s) =$$

$$\sum_{s' \in S} O(a, s', o) T(s, a, s') \alpha_i(s')$$

$$\alpha_{k+1}^{b,a} = r^a + \gamma \sum_{o \in \Omega} \arg\max_{g^{a,o}} b \cdot g^{a,o}$$

$$\text{backup}(\mathcal{A}_k, b) = \arg\max_{\alpha_{k+1}^{b,a}} b \cdot \alpha_{k+1}^{b,a}$$

(Scalarized) point-based backups

Back-projection of α-vectors $\alpha_i \in \mathcal{A}_k$:

$$g_i^{a,o}(s) = \sum_{s' \in S} O(a, s', o) T(s, a, s') \alpha_i(s')$$

$$\alpha_{k+1}^{b,a} = r^a + \gamma \sum_{o \in \Omega} \arg\max_{g^{a,o}} b \cdot g^{a,o}$$

$$\texttt{backup}(\mathcal{A}_k, b) = \arg\max_{\alpha_{k+1}^{b,a}} b \cdot \alpha_{k+1}^{b,a}$$

Back-projection of α-matrices $\mathbf{A}_i \in \mathcal{A}_k$, for a *given* $\mathbf{w}$:

$$\mathbf{G}_i^{a,o}(s) = \sum_{s' \in S} O(a, s', o) T(s, a, s') \mathbf{A}_i(s')$$

$$\mathbf{A}_{k+1}^{b,a} = r^a + \gamma \sum_{o \in \Omega} \arg\max_{G^{a,o}} b \; \mathbf{G}^{a,o}\mathbf{w}$$

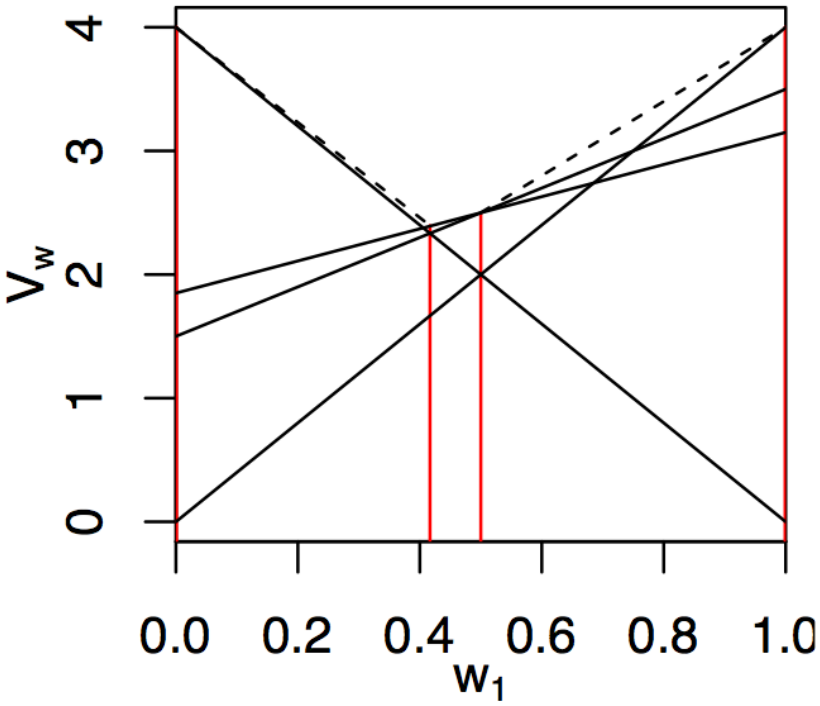
$$\texttt{backupMO}(\mathcal{A}_k, b, \mathbf{w}) = \arg\max_{\mathbf{A}_{k+1}^{b,a}} b\mathbf{A}_{k+1}^{a,b}\mathbf{w}$$

Optimistic linear support for MOPOMDPs

- Select series of **w** based on maximal possible improvement
- Use scalarized point-based planner as subroutine
  - Returns multi-objective value!
- But would still require an entire run of point-based planner *for each* **w**

Optimistic linear support with alpha reuse

- Starting from scratch for each $\mathbf{w}$ is inefficient
- Intuition: when $\mathbf{w}$ and $\mathbf{w}'$ are close, so are the optimal policies and values
- Hot start point-based planner using α-matrices from previous calls to scalarized point-based planner
- More and more effective as $\mathbf{w}$'s lie closer together



Theoretical results

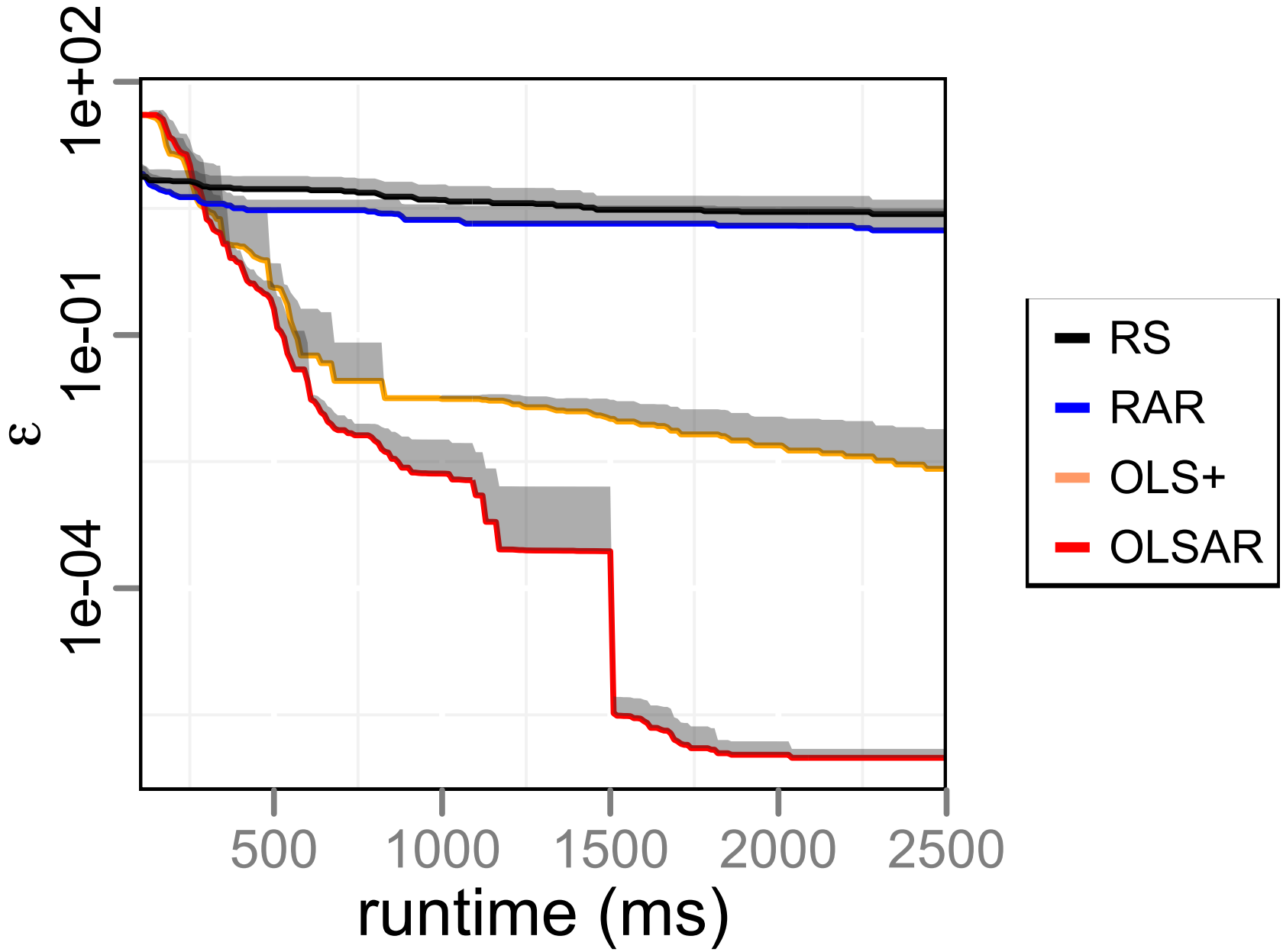
Theorem

OLSAR requires a finite number of calls to the point-based solver to converge.

Theorem

OLSAR produces an ε-approximate solution set.
ε is inherited from the single-objective method.

Sample of results: 3-objective tiger



Conclusions

- Use point-based methods for MOPOMDPs
- First method that reasonably scales
- Bounded approximation
- Alpha reuse is key to keeping MOPOMDPs tractable

Part 2: Methods and Applications

- Convex Coverage Set Planning Methods
  - Inner Loop: Convex Hull Value Iteration
  - Outer Loop: Optimistic Linear Support
- Pareto Coverage Set Planning Methods
  - Inner loop (non-stationary): Pareto-Q
  - Outer loop issues
- MOPOMDP Convex Coverage Set Planning: OLSAR
- *Applications*

Treatment planning

- Lizotte (2010, 2012)
  - Maximizing effectiveness of the treatment
  - Minimizing the severity of the side-effects
- Finite-horizon MOMDPs
- Deterministic policies

Epidemic control

- Anthrax response (Soh & Demiris (2011))
  - Minimizing loss of life
  - Minimizing number of false alarms
  - Minimizing cost of investigation
- Partial observability (MOPOMDP)
- Finite-state controllers
- Evolutionary method
- Pareto coverage set

Semi-autonomous wheelchairs

- Control system for wheelchairs (Soh & Demiris (2011))
  - Maximizing safety
  - Maximizing speed
  - Minimizing power consumption.
- Partial observability (MOPOMDP)
- Finite-state controllers
- Evolutionary method
- Pareto coverage set

Broader Application

- "Probabilistic Planning *is* Multi-objective" — Bryce et al. (2007)
  - The expected return is not enough
  - Cost of a plan
  - Probability of success of a plan
  - Non-goal terminal states

Closing

- Consider multiple objectives
  - most problems have them
  - a priori scalarization can be bad
- Derive your solution set
  - Pareto front often not necessary
- Promising applications