Dual Decomposition of Finite Horizon Markov Decision Processes

Thomas Furmston David Barber

Department of Computer Science
University College London

European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2011

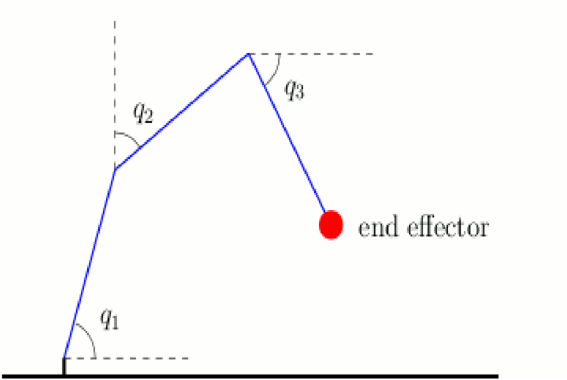
Outline

- Problem Framework
- Dual Decomposition
- Experiments
- Summary

PROBLEM FRAMEWORK

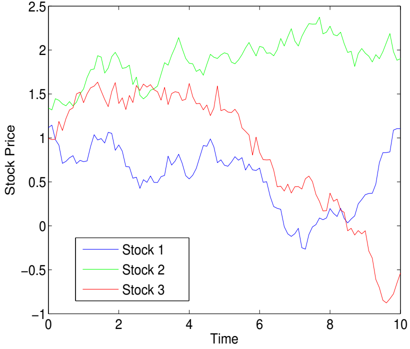
We are interested in the problem of optimal control in a dynamic environment. Examples include

- Robotics.



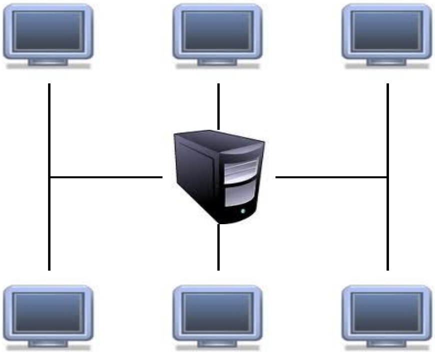
We are interested in the problem of optimal control in a dynamic environment. Examples include

- Robotics.
- Portfolio Optimisation.



We are interested in the problem of optimal control in a dynamic environment. Examples include

- Robotics.
- Portfolio Optimisation.
- Network Management.



Markov Decision Processes

We consider the problem of Markov Decision Processes, which are given by

- action-state space

  action space - $a \in \mathcal{A}$ (discrete).
  state space - $s \in \mathcal{S}$ (discrete).

- initial state distribution - $p_0(s)$.
- policy

  non-stationary - $\pi_t(a|s,t) = p(a|s,t;\pi)$.
  stationary - $\pi(a|s) = p(a|s;\pi)$.

- reward - $R(a,s)$.
- transition dynamics - $p(s'|s,a)$.
- planning horizon - H (finite or infinite).

Markov Decision Processes

We consider the problem of Markov Decision Processes, which are given by

- action-state space

  action space - $a \in \mathcal{A}$ (discrete).
  state space - $s \in \mathcal{S}$ (discrete).

- initial state distribution - $p_0(s)$.
- policy

  non-stationary - $\pi_t(a|s,t) = p(a|s,t;\pi)$.
  stationary - $\pi(a|s) = p(a|s;\pi)$.

- reward - $R(a,s)$.
- transition dynamics - $p(s'|s,a)$.
- planning horizon - H (finite or infinite).

Markov Decision Processes

We consider the problem of Markov Decision Processes, which are given by

- action-state space

  action space - $a \in \mathcal{A}$ (discrete).
  state space - $s \in \mathcal{S}$ (discrete).

- initial state distribution - $p_0(s)$.
- policy

  non-stationary - $\pi_t(a|s,t) = p(a|s,t;\pi)$.
  stationary - $\pi(a|s) = p(a|s;\pi)$.

- reward - $R(a,s)$.
- transition dynamics - $p(s'|s,a)$.
- planning horizon - H (finite or infinite).

Markov Decision Processes

We consider the problem of Markov Decision Processes, which are given by

- action-state space

  action space - $a \in \mathcal{A}$ (discrete).
  state space - $s \in \mathcal{S}$ (discrete).

- initial state distribution - $p_0(s)$.
- policy

  non-stationary - $\pi_t(a|s,t) = p(a|s,t;\pi)$.
  stationary - $\pi(a|s) = p(a|s;\pi)$.

- reward - $R(a,s)$.
- transition dynamics - $p(s'|s,a)$.
- planning horizon - H (finite or infinite).

Markov Decision Processes

We consider the problem of Markov Decision Processes, which are given by

- action-state space

  action space - $a \in \mathcal{A}$ (discrete).
  state space - $s \in \mathcal{S}$ (discrete).

- initial state distribution - $p_0(s)$.
- policy

  non-stationary - $\pi_t(a|s,t) = p(a|s,t;\pi)$.
  stationary - $\pi(a|s) = p(a|s;\pi)$.

- reward - $R(a,s)$.
- transition dynamics - $p(s'|s,a)$.
- planning horizon - H (finite or infinite).

Markov Decision Processes

We consider the problem of Markov Decision Processes, which are given by

- action-state space

  action space - $a \in \mathcal{A}$ (discrete).
  state space - $s \in \mathcal{S}$ (discrete).

- initial state distribution - $p_0(s)$.
- policy

  non-stationary - $\pi_t(a|s,t) = p(a|s,t;\pi)$.
  stationary - $\pi(a|s) = p(a|s;\pi)$.

- reward - $R(a,s)$.
- transition dynamics - $p(s'|s,a)$.
- planning horizon - H (finite or infinite).

Markov Decision Processes

We consider the problem of Markov Decision Processes, which are given by

- action-state space

  action space - $a \in \mathcal{A}$ (discrete).
  state space - $s \in \mathcal{S}$ (discrete).

- initial state distribution - $p_0(s)$.
- policy

  non-stationary - $\pi_t(a|s,t) = p(a|s,t;\pi)$.
  stationary - $\pi(a|s) = p(a|s;\pi)$.

- reward - $R(a,s)$.
- transition dynamics - $p(s'|s,a)$.
- planning horizon - H (finite or infinite).

Markov Decision Processes

Objective - Optimise π to maximise the total expected reward

$$U(\pi) = \sum_{t=1}^{H} \sum_{a_t, s_t} R(a_t, s_t) p(a_t, s_t; \pi),$$

where $p(a_t, s_t; \pi)$ is the marginal of the trajectory distribution

$$p(s_{1:H}, a_{1:H}; \pi) = p(a_H | s_H; \pi) p_0(s_1) \prod_{t=1}^{H-1} p(s_{t+1} | s_t, a_t) p(a_t | s_t; \pi).$$

Problem Framework

Interested in solving finite horizon MDP's with stationary policies, *i.e.*

- $H < \infty$,
- $\pi_t(a|s) = \pi(a|s), \qquad t = 1, ..., H.$

In particular we're interested in a **dynamic programming** 'type' solution to this problem class.

Other planning algorithms

EM - slow convergence.

Policy Gradients - susceptible to local optima.

Difficult - Bellman's *principal of optimality* no longer holds.

Problem Framework

Interested in solving finite horizon MDP's with stationary policies, *i.e.*

- $H < \infty$,
- $\pi_t(a|s) = \pi(a|s), \qquad t = 1, ..., H.$

In particular we're interested in a **dynamic programming** 'type' solution to this problem class.

Other planning algorithms

EM - slow convergence.

Policy Gradients - susceptible to local optima.

Difficult - Bellman's *principal of optimality* no longer holds.

Problem Framework

Interested in solving finite horizon MDP's with stationary policies, *i.e.*

- $H < \infty$,
- $\pi_t(a|s) = \pi(a|s), \qquad\qquad t = 1, ..., H.$

In particular we're interested in a **dynamic programming** 'type' solution to this problem class.

Other planning algorithms

EM - slow convergence.

Policy Gradients - susceptible to local optima.

Difficult - Bellman's *principal of optimality* no longer holds.

Problem Framework

Interested in solving finite horizon MDP's with stationary policies, *i.e.*

- $H < \infty$,
- $\pi_t(a|s) = \pi(a|s), \qquad t = 1, ..., H.$

In particular we're interested in a **dynamic programming** 'type' solution to this problem class.

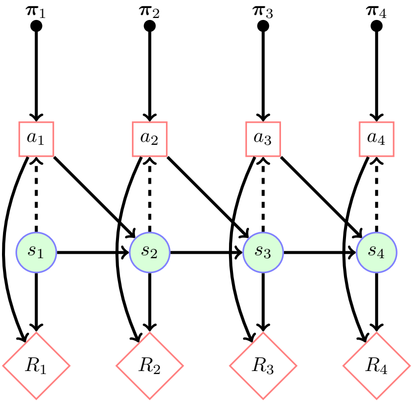
Other planning algorithms

EM - slow convergence.

Policy Gradients - susceptible to local optima.

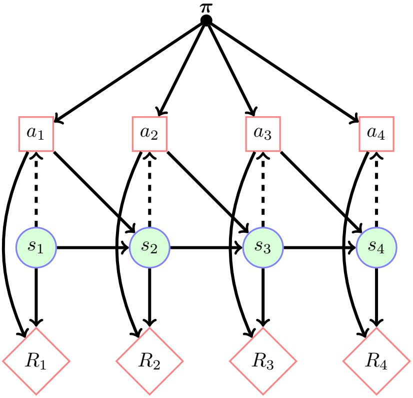
Difficult - Bellman's *principal of optimality* no longer holds.

Influence Diagrams



Non-Stationary Policies

Chain Structured - Easy to Optimise



Stationary Policies

Large Policy Clique - Difficult to Optimise

DUAL DECOMPOSITION

Dual Decomposition

Use idea of **dual decomposition** to exploit the theoretical ease of optimising a finite horizon MDP with non-stationary policies.

Original maximisation problem

$$\max_{\pi} \sum_{t=1}^{H} \sum_{a_t, s_t} R(a_t, s_t) p(a_t, s_t; \pi),$$

can be rewritten as

$$\max_{\substack{\pi, \pi_{1:H} \\ \pi_t = \pi, \forall t}} \sum_{t=1}^{H} \sum_{a_t, s_t} R(a_t, s_t) p(a_t, s_t; \pi_{1:t}).$$

Dual Decomposition

Use idea of **dual decomposition** to exploit the theoretical ease of optimising a finite horizon MDP with non-stationary policies.

Original maximisation problem

$$\max_{\pi} \sum_{t=1}^{H} \sum_{a_t, s_t} R(a_t, s_t) p(a_t, s_t; \pi),$$

can be rewritten as

$$\max_{\substack{\pi, \pi_{1:H} \\ \pi_t = \pi, \forall t}} \sum_{t=1}^{H} \sum_{a_t, s_t} R(a_t, s_t) p(a_t, s_t; \pi_{1:t}).$$

Dual Decomposition

Ordinarily the constraints $\pi_t = \pi$, $t = 1, ..., H$, would be handled by adjoining

$$\sum_{t=1}^{H} \sum_{a,s} \lambda_t(a, s)(\pi_t(a|s) - \pi(a|s)),$$

to the Lagrangian.

Note - this doesn't lead to dynamic programming solution.

So we consider the equivalent constraints

$$\sum_{t=1}^{H} \sum_{a,s} \lambda_t(a, s)(\pi_t(a|s) - \pi(a|s))p(s_t = s|\pi_{1:t-1}).$$

Dual Decomposition

Ordinarily the constraints $\pi_t = \pi$, $t = 1, ..., H$, would be handled by adjoining

$$\sum_{t=1}^{H} \sum_{a,s} \lambda_t(a, s)(\pi_t(a|s) - \pi(a|s)),$$

to the Lagrangian.

Note - this doesn't lead to dynamic programming solution.

So we consider the equivalent constraints

$$\sum_{t=1}^{H} \sum_{a,s} \lambda_t(a, s)(\pi_t(a|s) - \pi(a|s))p(s_t = s|\pi_{1:t-1}).$$

Dual Decomposition

Ordinarily the constraints $\pi_t = \pi$, $t = 1, ..., H$, would be handled by adjoining

$$\sum_{t=1}^{H} \sum_{a,s} \lambda_t(a, s)(\pi_t(a|s) - \pi(a|s)),$$

to the Lagrangian.

Note - this doesn't lead to dynamic programming solution.

So we consider the equivalent constraints

$$\sum_{t=1}^{H} \sum_{a,s} \lambda_t(a, s)(\pi_t(a|s) - \pi(a|s))p(s_t = s|\pi_{1:t-1}).$$

Dual Decomposition

This leads to objective function

$$L(\pi, \pi_{1:H}, \lambda_{1:H}) = \sum_{t=1}^{H} \sum_{a_t, s_t} \left\{ \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}) \right.$$
$$\left. - \lambda_t(a_t, s_t) \pi(a_t | s_t) p(s_t | \pi_{1:t-1}) \right\}$$

Can perform optimisation over π.

This gives constraint set $\Lambda(\pi_{1:H})$ over Lagrange multipliers

$$\sum_{t=1}^{H} \lambda_t(a, s) p(s_t = s | \pi_{1:t-1}) = 0, \qquad \forall (a, s) \in \mathcal{S} \times \mathcal{A}.$$

Dual Decomposition

This leads to objective function

$$L(\pi, \pi_{1:H}, \lambda_{1:H}) = \sum_{t=1}^{H} \sum_{a_t, s_t} \Big\{ \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}) \\ - \lambda_t(a_t, s_t) \pi(a_t | s_t) p(s_t | \pi_{1:t-1}) \Big\}$$

Can perform optimisation over π.

This gives constraint set $\Lambda(\pi_{1:H})$ over Lagrange multipliers

$$\sum_{t=1}^{H} \lambda_t(a, s) p(s_t = s | \pi_{1:t-1}) = 0, \qquad \forall (a, s) \in \mathcal{S} \times \mathcal{A}.$$

Dual Decomposition

This leads to objective function

$$L(\pi, \pi_{1:H}, \lambda_{1:H}) = \sum_{t=1}^{H} \sum_{a_t, s_t} \bigg\{ \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}) - \lambda_t(a_t, s_t) \pi(a_t | s_t) p(s_t | \pi_{1:t-1}) \bigg\}$$

Can perform optimisation over π.

This gives constraint set $\Lambda(\pi_{1:H})$ over Lagrange multipliers

$$\sum_{t=1}^{H} \lambda_t(a, s) p(s_t = s | \pi_{1:t-1}) = 0, \qquad \forall (a, s) \in \mathcal{S} \times \mathcal{A}.$$

Dual Objective

Final dual objective function

$$L(\lambda_{1:H}, \pi_{1:H}) = \sum_{t=1}^{H} \sum_{a_t, s_t} \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}).$$

This is optimised iteratively through a sequence of

- **slave** problems
- **master** problems

Dual Objective

Final dual objective function

$$L(\lambda_{1:H}, \pi_{1:H}) = \sum_{t=1}^{H} \sum_{a_t, s_t} \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}).$$

This is optimised iteratively through a sequence of

- **slave** problems
- **master** problems

Slave Problem

For fixed $\lambda_{1:H}$ maximisation over $\pi_{1:H}$ takes the form

$$\underset{\pi_{1:H}}{\operatorname{argmax}} \sum_{t=1}^{H} \sum_{a_t, s_t} \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}) \qquad (1)$$

- Objective (1) an ordinary MDP with non-stationary policies.
- Lagrange multipliers leads to non-stationary rewards.
- Solvable using dynamic programming.

Slave Problem

For fixed $\lambda_{1:H}$ maximisation over $\pi_{1:H}$ takes the form

$$\underset{\pi_{1:H}}{\text{argmax}} \sum_{t=1}^{H} \sum_{a_t, s_t} \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}) \qquad (1)$$

- Objective (1) an ordinary MDP with non-stationary policies.
- Lagrange multipliers leads to non-stationary rewards.
- Solvable using dynamic programming.

Slave Problem

For fixed $\lambda_{1:H}$ maximisation over $\pi_{1:H}$ takes the form

$$\operatorname*{argmax}_{\pi_{1:H}} \sum_{t=1}^{H} \sum_{a_t, s_t} \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}) \qquad (1)$$

- Objective (1) an ordinary MDP with non-stationary policies.
- Lagrange multipliers leads to non-stationary rewards.
- Solvable using dynamic programming.

Slave Problem

For fixed $\lambda_{1:H}$ maximisation over $\pi_{1:H}$ takes the form

$$\underset{\pi_{1:H}}{\operatorname{argmax}} \sum_{t=1}^{H} \sum_{a_t, s_t} \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}) \qquad (1)$$

- Objective (1) an ordinary MDP with non-stationary policies.
- Lagrange multipliers leads to non-stationary rewards.
- Solvable using dynamic programming.

Master Problem

For fixed $\pi_{1:H}$ minimisation over $\lambda_{1:H}$ takes the form

$$\underset{\lambda_{1:H} \in \Lambda}{\operatorname{argmin}} \sum_{t=1}^{H} \sum_{a_t, s_t} \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}).$$

Minimisation done using a **projected sub-gradient step.**

Gradient Step - take step in direction of anti-gradient

$$\lambda_t^i \leftarrow \lambda_t^{i-1} - \eta_{i-1} \pi_t^{i-1}.$$

Projection Step - project $\lambda_{1:H}$ back down into constraint set Λ

$$\lambda_t^i(s, a) \leftarrow \lambda_t^i(s, a) - \sum_{\tau=1}^{H} \rho_\tau(s) \lambda_\tau^i(s, a).$$

Master Problem

For fixed $\pi_{1:H}$ minimisation over $\lambda_{1:H}$ takes the form

$$\underset{\lambda_{1:H}\in\Lambda}{\operatorname{argmin}} \sum_{t=1}^{H} \sum_{a_t,s_t} \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}).$$

Minimisation done using a **projected sub-gradient step.**

Gradient Step - take step in direction of anti-gradient

$$\lambda_t^i \leftarrow \lambda_t^{i-1} - \eta_{i-1} \pi_t^{i-1}.$$

Projection Step - project $\lambda_{1:H}$ back down into constraint set Λ

$$\lambda_t^i(s, a) \leftarrow \lambda_t^i(s, a) - \sum_{\tau=1}^{H} \rho_\tau(s) \lambda_\tau^i(s, a).$$

Master Problem

For fixed $\pi_{1:H}$ minimisation over $\lambda_{1:H}$ takes the form

$$\operatorname*{argmin}_{\lambda_{1:H}\in\Lambda} \sum_{t=1}^{H} \sum_{a_t,s_t} \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t|\pi_{1:t}).$$

Minimisation done using a **projected sub-gradient step.**

Gradient Step - take step in direction of anti-gradient

$$\lambda_t^i \leftarrow \lambda_t^{i-1} - \eta_{i-1}\pi_t^{i-1}.$$

Projection Step - project $\lambda_{1:H}$ back down into constraint set Λ

$$\lambda_t^i(s,a) \leftarrow \lambda_t^i(s,a) - \sum_{\tau=1}^{H} \rho_\tau(s)\lambda_\tau^i(s,a).$$

Master Problem

For fixed $\pi_{1:H}$ minimisation over $\lambda_{1:H}$ takes the form

$$\underset{\lambda_{1:H}\in\Lambda}{\operatorname{argmin}} \sum_{t=1}^{H} \sum_{a_t,s_t} \Big(R(a_t, s_t) + \lambda_t(a_t, s_t) \Big) p(a_t, s_t | \pi_{1:t}).$$

Minimisation done using a **projected sub-gradient step.**

Gradient Step - take step in direction of anti-gradient

$$\lambda_t^i \leftarrow \lambda_t^{i-1} - \eta_{i-1}\pi_t^{i-1}.$$

Projection Step - project $\lambda_{1:H}$ back down into constraint set Λ

$$\lambda_t^i(s,a) \leftarrow \lambda_t^i(s,a) - \sum_{\tau=1}^{H} \rho_\tau(s)\lambda_\tau^i(s,a).$$

Algorithm Summary

Summary - dual decomposition solution iterates between **slave problem** and the **master problem** until convergence.

- **Slave Problem** - Update $\pi_{1:H}$ by solving a finite horizon MDP with
  - non-stationary policies.
  - non-stationary rewards - $\hat{R}_t = R + \lambda_t$.
- **Master Problem** - Update $\lambda_{1:H}$ using a projected sub-gradient step.

Resource Allocation

Dual decomposition algorithm adjusts non-stationary rewards (*i.e.* Lagrange multipliers) to obtain stationary policies.

Question - How are $\lambda_{1:H}$ updated?

We show the following relation

$$\lambda_t^{i+1}(s,a) \begin{cases} \leq \lambda_t^i(s,a) & \text{if } a = \underset{a}{\text{argmax}}\, \pi_t^i(a|s), \\ \geq \lambda_t^i(s,a) & \text{if otherwise.} \end{cases}$$

Additionally, the difference obeys the relation

$$|\lambda_t^{i+1}(s,a) - \lambda_t^i(s,a)| = \mathcal{O}(H - N_i(s,a)),$$

where $N_i(s,a)$

$$N_i(s,a) = \left| \left\{ t \in \{1, ..., H\} \middle| \pi_t(a|s) = 1 \right\} \right|$$

Dual decomposition algorithm adjusts non-stationary rewards (*i.e.* Lagrange multipliers) to obtain stationary policies.

Question - How are $\lambda_{1:H}$ updated?

We show the following relation

$$\lambda_t^{i+1}(s,a) \begin{cases} \leq \lambda_t^i(s,a) & \text{if } a = \underset{a}{\operatorname{argmax}}\, \pi_t^i(a|s), \\ \geq \lambda_t^i(s,a) & \text{if otherwise.} \end{cases}$$

Additionally, the difference obeys the relation

$$|\lambda_t^{i+1}(s,a) - \lambda_t^i(s,a)| = \mathcal{O}(H - N_i(s,a)),$$

where $N_i(s,a)$

$$N_i(s,a) = \left| \left\{ t \in \{1, ..., H\} \middle| \pi_t(a|s) = 1 \right\} \right|$$

Dual decomposition algorithm adjusts non-stationary rewards (*i.e.* Lagrange multipliers) to obtain stationary policies.

Question - How are $\lambda_{1:H}$ updated?

We show the following relation

$$\lambda_t^{i+1}(s,a) \begin{cases} \leq \lambda_t^i(s,a) & \text{if } a = \underset{a}{\operatorname{argmax}}\ \pi_t^i(a|s), \\ \geq \lambda_t^i(s,a) & \text{if otherwise.} \end{cases}$$

Additionally, the difference obeys the relation

$$|\lambda_t^{i+1}(s,a) - \lambda_t^i(s,a)| = \mathcal{O}(H - N_i(s,a)),$$

where $N_i(s,a)$

$$N_i(s,a) = \left| \left\{ t \in \{1, ..., H\} \middle| \pi_t(a|s) = 1 \right\} \right|$$

Resource Allocation

Dual decomposition algorithm adjusts non-stationary rewards (*i.e.* Lagrange multipliers) to obtain stationary policies.

Question - How are $\lambda_{1:H}$ updated?

We show the following relation

$$\lambda_t^{i+1}(s,a) \begin{cases} \leq \lambda_t^i(s,a) & \text{if } a = \underset{a}{\text{argmax}}\ \pi_t^i(a|s), \\ \geq \lambda_t^i(s,a) & \text{if otherwise.} \end{cases}$$

Additionally, the difference obeys the relation

$$|\lambda_t^{i+1}(s,a) - \lambda_t^i(s,a)| = \mathcal{O}(H - N_i(s,a)),$$

where $N_i(s,a)$

$$N_i(s,a) = \left| \left\{ t \in \{1, ..., H\} \middle| \pi_t(a|s) = 1 \right\} \right|$$

Example - Consider an MDP with 2 actions.

If in a given a state, s, the previous slave problem found

- action a_1 was optimal for a **large** number of time points,
- while action a_2 was optimal for only a **few** time points,

then

- for time-points where a_1 was **optimal**

  $\lambda_t(a_1, s)$ - would **decrease only slightly**
  $\lambda_t(a_2, s)$ - would **increase only slightly**

- for time-points where a_2 was **optimal**

  $\lambda_t(a_1, s)$ - would **increase more dramatically**
  $\lambda_t(a_2, s)$ - would **decrease more dramatically**

Example - Consider an MDP with 2 actions.

If in a given a state, s, the previous slave problem found

- action a_1 was optimal for a **large** number of time points,
- while action a_2 was optimal for only a **few** time points,

then

- for time-points where a_1 was **optimal**

  $\lambda_t(a_1, s)$ - would **decrease only slightly**
  $\lambda_t(a_2, s)$ - would **increase only slightly**

- for time-points where a_2 was **optimal**

  $\lambda_t(a_1, s)$ - would **increase more dramatically**
  $\lambda_t(a_2, s)$ - would **decrease more dramatically**

Example - Consider an MDP with 2 actions.

If in a given a state, s, the previous slave problem found

- action a_1 was optimal for a **large** number of time points,
- while action a_2 was optimal for only a **few** time points,

then

- for time-points where a_1 was **optimal**

  $\lambda_t(a_1, s)$ - would **decrease only slightly**

  $\lambda_t(a_2, s)$ - would **increase only slightly**

- for time-points where a_2 was **optimal**

  $\lambda_t(a_1, s)$ - would **increase more dramatically**

  $\lambda_t(a_2, s)$ - would **decrease more dramatically**

Resource Allocation - An Example

Example - Consider an MDP with 2 actions.

If in a given a state, *s*, the previous slave problem found

- action a_1 was optimal for a **large** number of time points,
- while action a_2 was optimal for only a **few** time points,

then

- for time-points where a_1 was **optimal**

  $\lambda_t(a_1, s)$ - would **decrease only slightly**
  $\lambda_t(a_2, s)$ - would **increase only slightly**

- for time-points where a_2 was **optimal**

  $\lambda_t(a_1, s)$ - would **increase more dramatically**
  $\lambda_t(a_2, s)$ - would **decrease more dramatically**

EXPERIMENTS

Comparisons

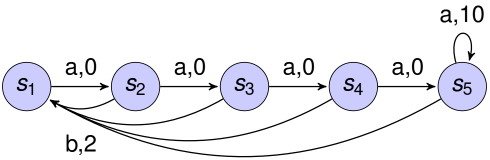
We compare our Dual Decomposition Dynamic Programming (DD DP) algorithm against;

- Expectation Maximisation (EM)
- Policy Gradients (PG)
    - Fixed Step Size
    - Line Search
- Expectation Maximisation - Policy Gradients (EM-PG)

Chain Problem

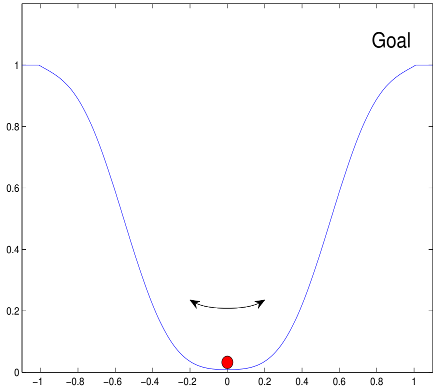
Objective - For $H = 25$ it is optimal to manoeuvre the agent to the right-most end of the chain.

- $|\mathcal{S}| = 5$.
- $|\mathcal{A}| = 2$.
- H = 25.



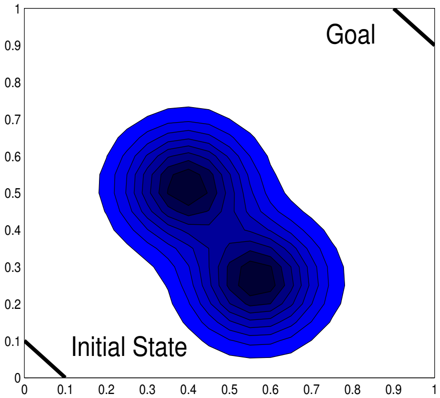
Objective - Manoeuvre the agent to the goal region at the right-most peak of the valley.

- $|\mathcal{S}| = 231$.
- $|\mathcal{A}| = 3$.
- H = 25.



Objective - Manoeuvre the agent to the goal region whilst avoiding the puddles, which cause a negative reward.

- $|\mathcal{S}| = 441$.
- $|\mathcal{A}| = 4$.
- H = 50.



Results

		Algorithms				
		DD DP	EM	F-PG	LS-PG	EM-PG
Chain Problem	$U(\pi^*)$	86	85	75	65	86
	Iterations	3	100	100	3	100
Mountain Car	$U(\pi^*)$	19	19	16	14	19
	Iterations	7	100	100	3	100
Puddle World	$U(\pi^*)$	42	39	N/A	0	N/A
	Iterations	30	1000	N/A	10	N/A

SUMMARY
&
FUTURE WORK

Summary & Future Work

Summary
We have presented that dual decomposition algorithm for finite horizon MDP's with stationary policies.

Future work

- Extend to continuous state-action domains.
- Extend to more complex domains, such as *partially observable Markov decision processes*.