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

- **action-state space**
  
  \begin{align*}
  \text{action space} & \quad a \in \mathcal{A} \text{ (discrete).} \\
  \text{state space} & \quad s \in \mathcal{S} \text{ (discrete).}
  \end{align*}

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

- **policy**
  
  \begin{align*}
  \text{non-stationary} & \quad \pi_t(a|s, t) = p(a|s, t; \pi). \\
  \text{stationary} & \quad \pi(a|s) = p(a|s; \pi).
  \end{align*}

- **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).
We consider the problem of Markov Decision Processes, which are given by

- **action-state space**
  - action space - \( a \in A \) (discrete).
  - state space - \( s \in S \) (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).
We consider the problem of Markov Decision Processes, which are given by

- action-state space
  - action space - $a \in A$ (discrete).
  - state space - $s \in S$ (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).
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).
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).
Objective - Optimise $\pi$ 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

$$p(s_{1:H}, a_{1:H}; \pi) = p(a_H|s_H; \pi)p_0(s_1) \prod_{t=1}^{H-1} p(s_{t+1}|s_t, a_t)p(a_t|s_t; \pi).$$
Interested in solving finite horizon MDP’s with stationary policies, \( i.e. \)

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

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.
Interested in solving finite horizon MDP’s with stationary policies, \( i.e. \):

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

In particular we’re interested in a \textit{dynamic programming} ‘type’ solution to this problem class.

Other planning algorithms

- \text{EM} - slow convergence.
- \text{Policy Gradients} - susceptible to local optima.

\textbf{Difficult} - Bellman’s \textit{principal of optimality} no longer holds.
Interested in solving finite horizon MDP’s with stationary policies, i.e.

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

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

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

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.
Non-Stationary Policies
Chain Structured - Easy to Optimise

Stationary Policies
Large Policy Clique - Difficult to Optimise
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

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

\[
\max_{\pi, \pi_1:H} \sum_{\pi_t=\pi} \sum_{\forall t=1} \sum_{a_t,s_t} R(a_t, s_t) p(a_t, s_t; \pi_1:t).
\]
Ordinarily the constraints $\pi_t = \pi$, $t = 1, \ldots, H$, 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

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

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

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

Can perform optimisation over \( \pi \).

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

\[
\sum_{t=1}^{H} \lambda_t(a, s) p(s_t = s | \pi_{1:t-1}) = 0, \quad \forall (a, s) \in S \times A.
\]
This leads to objective function

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

Can perform optimisation over \( \pi \).

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

\[
\sum_{t=1}^{H} \lambda_t(a,s) p(s_t = s|\pi_{1:t-1}) = 0, \quad \forall (a,s) \in S \times A.
\]
This leads to objective function

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

\[ - \lambda_t(a_t, s_t) \pi(a_t | s_t) p(s_t | \pi_{1:t-1}) \left. \right\} \]

Can perform optimisation over \( \pi \).

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

\[ \sum_{t=1}^{H} \lambda_t(a, s) p(s_t = s | \pi_{1:t-1}) = 0, \quad \forall (a, s) \in S \times A. \]
Final dual objective function

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

This is optimised iteratively through a sequence of

- **slave** problems
- **master** problems
Final dual objective function

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

This is optimised iteratively through a sequence of

- **slave** problems
- **master** problems
For fixed $\lambda_{1:H}$ maximisation over $\pi_{1:H}$ takes the form

$$\arg\max_{\pi_{1:H}} \sum_{t=1}^{H} \sum_{a_t,s_t} \left( R(a_t, s_t) + \lambda_t(a_t, s_t) \right) p(a_t, s_t | \pi_{1:t})$$

(1)

- Objective (1) an ordinary MDP with non-stationary policies.
- Lagrange multipliers leads to non-stationary rewards.
- Solvable using dynamic programming.
For fixed $\lambda_{1:H}$ maximisation over $\pi_{1:H}$ takes the form

$$\arg\max_{\pi_{1:H}} \sum_{t=1}^{H} \sum_{a_t, s_t} \left( R(a_t, s_t) + \lambda_t(a_t, s_t) \right) p(a_t, s_t | \pi_{1:t})$$  \hspace{1cm} (1)

- Objective (1) an ordinary MDP with non-stationary policies.
- Lagrange multipliers leads to non-stationary rewards.
- Solvable using dynamic programming.
For fixed $\lambda_{1:H}$ maximisation over $\pi_{1:H}$ takes the form

$$\arg\max_{\pi_{1:H}} \sum_{t=1}^{H} \sum_{a_t, s_t} \left( R(a_t, s_t) + \lambda_t(a_t, s_t) \right) p(a_t, s_t | \pi_{1:t})$$

(1)

- Objective (1) an ordinary MDP with non-stationary policies.
- Lagrange multipliers leads to non-stationary rewards.
- Solvable using dynamic programming.
For fixed $\lambda_{1:H}$ maximisation over $\pi_{1:H}$ takes the form

$$\arg\max_{\pi_{1:H}} \sum_{t=1}^{H} \sum_{a_t, s_t} \left( R(a_t, s_t) + \lambda_t(a_t, s_t) \right) p(a_t, s_t | \pi_{1:t})$$

(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

$$\arg\min_{\lambda_{1:H}\in\Lambda} \sum_{t=1}^{H} \sum_{a_t,s_t} \left( R(a_t, s_t) + \lambda_t(a_t, s_t) \right) p(a_t, s_t|\pi_{1:t}).$$

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 $\Lambda$

$$\lambda_t^i(s, a) \leftarrow \lambda_t^i(s, a) - \sum_{\tau=1}^{H} \rho_{\tau}(s) \lambda_{\tau}^i(s, a).$$
For fixed $\pi_{1:H}$ minimisation over $\lambda_{1:H}$ takes the form

$$\arg\min_{\lambda_{1:H} \in \Lambda} \sum_{t=1}^{H} \sum_{a_t, s_t} \left( R(a_t, s_t) + \lambda_t(a_t, s_t) \right) p(a_t, s_t | \pi_{1:t}).$$

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

**Gradient Step** - take step in direction of anti-gradient

$$\lambda^i_t \leftarrow \lambda^i_{t-1} - \eta_{i-1} \pi^i_{t-1}.$$

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

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

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

$$\arg\min_{\lambda_{1:H}\in\Lambda} \sum_{t=1}^{H} \sum_{a_t,s_t} \left( R(a_t, s_t) + \lambda_t(a_t, s_t) \right) p(a_t, s_t | \pi_{1:t}).$$

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

**Gradient Step** - take step in direction of anti-gradient

$$\lambda^i_t \leftarrow \lambda^{i-1}_t - \eta_{i-1} \pi^{i-1}_t.$$

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

$$\lambda^i_t(s, a) \leftarrow \lambda^i_t(s, a) - \sum_{\tau=1}^{H} \rho_{\tau}(s) \lambda^i_{\tau}(s, a).$$
Master Problem

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

$$\arg\min_{\lambda_{1:H} \in \Lambda} \sum_{t=1}^{H} \sum_{a_t, s_t} \left( R(a_t, s_t) + \lambda_t(a_t, s_t) \right) p(a_t, s_t | \pi_{1:t}).$$

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 $\Lambda$

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

$$
\begin{align*}
\lambda^{i+1}_t(s, a) & \leq \lambda^i_t(s, a) \quad \text{if } a = \arg\max_a \pi^i_t(a|s), \\
\lambda^{i+1}_t(s, a) & \geq \lambda^i_t(s, a) \quad \text{if otherwise}.
\end{align*}
$$

Additionally, the difference obeys the relation

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

where $N_i(s, a)$

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

$$
\lambda^{i+1}_t(s, a) \begin{cases} 
\leq \lambda^i_t(s, a) & \text{if } a = \arg\max_a \pi^i_t(a|s), \\
\geq \lambda^i_t(s, a) & \text{if otherwise.}
\end{cases}
$$

Additionally, the difference obeys the relation

$$
|\lambda^{i+1}_t(s, a) - \lambda^i_t(s, a)| = O(H - N_i(s, a)),
$$

where $N_i(s, a)$

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

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

Additionally, the difference obeys the relation

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

where $N_i(s, a)$

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

\[
\lambda^{i+1}_t(s, a) \begin{cases} 
\leq \lambda^i_t(s, a) & \text{if } a = \arg\max_a \pi^i_t(a|s), \\
\geq \lambda^i_t(s, a) & \text{if otherwise.}
\end{cases}
\]

Additionally, the difference obeys the relation

\[
|\lambda^{i+1}_t(s, a) - \lambda^i_t(s, a)| = O(H - N_i(s, a)),
\]

where \(N_i(s, a)\)

\[
N_i(s, a) = \left\{ t \in \{1, ..., H\} \mid \pi_t(a|s) = 1 \right\}
\]
Example - Consider an MDP with 2 actions.

If in a given 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

Thomas Furmston, David Barber
Dual Decomposition of Finite Horizon MDP's
**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
  \[ \lambda_t(a_1, s) \text{ - would decrease only slightly} \]
  \[ \lambda_t(a_2, s) \text{ - would increase only slightly} \]

- for time-points where $a_2$ was optimal
  \[ \lambda_t(a_1, s) \text{ - would increase more dramatically} \]
  \[ \lambda_t(a_2, s) \text{ - would decrease more dramatically} \]
EXPERIMENTS
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)
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.

- $|S| = 441$.
- $|A| = 4$.
- $H = 50$. 
### Results

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