Autonomous Agents 2: Multi-objective decision problems

Diederik Roijers
Informatics Institute
University of Amsterdam

April 30, 2015
A simple problem...

- Let’s say you have a wart on your finger
- Virus, painful, contagious, can in very rare cases lead to skin cancer
- Two treatments:
  - 97.00% probability of being cured
  - 99.99% probability of being cured

A slightly more complex problem...

The Dutch government has been attempting to decrease traffic jams in the Randstad, for a number of decades now. Any solution should balance:

- Percentage of traffic jams decreased
- ...

[Map of the Netherlands]
Why Multi-Objective?

Real-world problems just are Multi-Objective

And AI can help.
Today: MOMDPs and Multi-agent problems
A finite single-objective Markov decision process (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,
- $T : S \times A \times S \to [0, 1]$ is a transition function specifying, for each state, action, and next state, the probability of that next state occurring,
- $R : S \times A \times S \to \mathbb{R}$ is a reward function, specifying, for each state, action, and next state, the expected immediate reward,
- $\mu : S \to [0, 1]$ is a probability distribution over initial states, and
- $\gamma \in [0, 1)$ is a discount factor specifying the relative importance of immediate rewards.
A finite single-objective *multi-objective Markov decision process* (MOMDP), with \( n \) objectives, is a tuple \( \langle S, A, T, R, \mu, \gamma \rangle \) where:

- \( S, A, T, \mu \) and \( \gamma \) are the same as in an MDP, but
- \( R : S \times A \times S \rightarrow \mathbb{R}^n \) is a *reward function*, specifying, for each state, action, and next state, the expected immediate vector-valued reward.
MOMDP equations

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

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

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

Additive vector valued returns.
But wait...

Can’t we just *scalarize* the decision problem?

- Find a function $f$ that translates the multiple objectives to a scalar utility: a scalarization function

\[
V_w^\pi = f(V^\pi, w)
\]

- Use the scalarization function to define an equivalent single-objective problem
- Solve that problem, and we’re done
  
  (http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html)

- ... right?
Scenarios

- We identified 3 scenarios in which scalarization before planning or learning is:
  - Impossible
  - Undesirable
  - Infeasable

- These scenarios are called:
  - (a) unknown weights
  - (b) decision support
  - (c) known weights
The unknown weights scenario

- We know the scalarization function, but not the weights $w$, for an MOMDP.
- Planning (or learning), is expensive, but we have quite a bit of time before we need to act.
- When the weights come in however, we want to act immediate.
- Furthermore, the weights may change quickly.
- Example: resources and costs of varying prices on the market and mining company trying to obtain resources.
Decision support scenario

- We know the MOMDP (or might have a simulator), but it is hard to construct a plan for it.
- Furthermore, the trade-offs between the objectives are hard.
- The people who have to determine the weights, e.g. a committee at a local government, want to be presented with all the alternatives first.

- Example: Changing the traffic situation of a city to improve the flow of traffic, while minimizing noise levels and pollution.
Multiple policies

- The *unknown weights* and *decision support* scenarios, require a solution for all *possible scalarizations*.
- They require *multiple policies* to be computed.
Known weights scenario

- We know the MOMDP (or might have a simulator), and know the scalarization function, and weights, but still cannot scalarize.
- The form of the scalarization function is complex.
- If we try to scalarize the problem before planning or learning, this can lead to undesirably complex scalar valued problems.
- The known weights scenario is a single-policy scenario, but does require special methods.
Scenarios

(a) MOMDP → algorithm → solution set → scalarization → single solution

planning or learning phase

(b) MOMDP → algorithm → solution set → user selection → single solution

planning or learning phase

(c) MOMDP + weights → algorithm → single solution

planning or learning phase

execution phase
Contents

- Why Multi-Objective?
- MOMDPs
- Motivating Scenarios
- Taxonomy
- Methods
- Optimistic linear support
Taxonomy

- So, let’s solve some MOMDPs
- We have seen that:

\[ V_w^\pi = f(V^\pi, w) \]

- Solving for all possible scalarizations gives us the undominated set:

\[ U = \{ \pi : \exists w \forall \pi' \ V_w^\pi \geq V_w^{\pi'} \} \]

- But what is the scalarization function?
- What type of policies do we require/allow?
So, let’s solve some MOMDPs
We have seen that:

\[ V^\pi_w = f(V^\pi, w) \]

Solving for all possible scalarizations gives us the undominated set:

\[ U = \{ \pi : \exists w \forall \pi' V^\pi_w \geq V^\pi'_w \} \]

But what is the scalarization function?
→ a property of the problem!
What type of policies do we require/allow?
→ a property of the problem!
MOMDP Taxonomy

- type of scalarization function (linear, monotonically increasing)
- stochastic policies versus deterministic policies
- single policy versus multiple policies
Scalarization functions

- Linear scalarization function:

\[ V_w = f(V^\pi, w) = w \cdot V^\pi \]

- Some function that is *monotonically increasing* in all objectives
The linear scalarization function

- Linear scalarization function:

\[ V_w^{\pi} = f(V^{\pi}, w) = w \cdot V^{\pi} \]

- Common, e.g. prices of different resources
- What happens if we the scalarization function is linear and we know the weights?
The linear scalarization function

- Linear scalarization function:

\[ V_w = f(V^\pi, w) = w \cdot V^\pi \]

- Common, e.g. prices of different resources

- Known weights scenario \(\rightarrow\) trivial (single objective MDP translation)

- Undominated set \(\rightarrow\) Convex Hull

\[ CH = \{ \pi : \exists w \forall \pi' \ w \cdot V^\pi \geq w \cdot V^{\pi'} \} \]
The linear scalarization function

- We need only to consider stationary deterministic policies.
- Why?
The linear scalarization function

- We need only to consider stationary deterministic policies.
- Why? Hint:
  - For single objective MDPs we know that there always is an optimal policy (one with the maximum possible value), that is deterministic and stationary...
  - ...

The linear scalarization function

- We need only to consider stationary deterministic policies.
- Why?
  - For single objective MDPs we know that there always is an optimal policy (one with the maximum possible value), that is deterministic and stationary.
  - For each possible weight vector $w$, an MOMDP with a linear scalarization problem can be translated to an equivalent MDP.
  - Ergo, for all possible weight vectors, there is an optimal policy that is deterministic and stationary.
Exercise

- Three armed bandit (single state), with deterministic direct (2-dimensional) rewards:
  - \( a_1 \rightarrow (3, 0) \)
  - \( a_2 \rightarrow (1, 1) \)
  - \( a_3 \rightarrow (0, 3) \)
  - \( \gamma = 3/4 \)
- Find the Convex Hull

\[
CH = \{ \pi : \exists \mathbf{w} \forall \pi' \mathbf{w} \cdot \mathbf{V}^\pi \geq \mathbf{w} \cdot \mathbf{V}^{\pi'} \}
\]

- What are the (2D) Value(s/ vectors)?
Convex upper surface
Summary Linear Scalarizations

- Single policy
  - One stationary deterministic policy
- Multiple policies
  - Convex Hull of stationary deterministic policies
Monotonically increasing scalarization functions

- What if all we can assume about the scalarization function is that it is monotonically increasing in all objectives?
  - If we keep the values for all objectives but one the same, and increase the value of the other one, the scalarized value cannot go down.

- A Pareto-dominated policy is always worse than a non-dominated policy:
  \[
  \mathbf{V}^\pi \succ_P \mathbf{V}^\pi' \iff \forall i, V_i^\pi \geq V_i^\pi' \land \exists i, V_i^\pi > V_i^\pi'
  \]

- Scalarized(!) returns can become non-additive: e.g. if
  \[
  f(\mathbf{V}^\pi, w) = w \prod_i \max(0, V_i^\pi)
  \]
  then,

  \[
  f(\mathbf{V}^\pi, w) \neq E[\sum_{k=0}^{\infty} \gamma^k f(r_{t+k+1}, w)].
  \]
Exercise revisited

- Three armed bandit (single state), with deterministic direct (2 dimensional) rewards:
  - $a_1 \rightarrow (3, 0)$
  - $a_2 \rightarrow (1, 1)$
  - $a_3 \rightarrow (0, 3)$
  - $\gamma = 3/4$
- The scalarization function is: $f(\textbf{V}^\pi, w) = w \prod_i \max(0, V_{i}^\pi)$ if all, where $w$ is a positive constant.
- What is the optimal policy?
Exercise revisited

- Three armed bandit (single state), with deterministic direct (2 dimensional) rewards:
  - $a_1 \rightarrow (3, 0)$
  - $a_2 \rightarrow (1, 1)$
  - $a_3 \rightarrow (0, 3)$
  - $\gamma = 3/4$
- The scalarization function is: $f(\mathbf{V}^\pi, w) = w \prod_i \max(0, V_i^\pi)$, where $w$ is a constant.
- What is the optimal policy?
  - Stochastic policy?
  - If we allow only deterministic policies, can we suffice with stationary policies?
Monotonically increasing scalarization functions

- If we do not allow stochastic policies,
  - E.g. in medical decisions it is not exceptable to take actions stochastically
  - It is not acceptable to treat patients stochastically and look only at the *expected* (read average) returns.

- We may have to resort to non-stationary policies, i.e. policies that condition their actions on time.

- This does not occur in single objective MDPs!

- We need the Pareto-front of deterministic non-stationary policies:

$$PF = \{ \pi : \neg \exists \pi', V^{\pi'} \succeq_{p} V^{\pi} \}.$$
Pareto front vs. convex hull
Monotonically increasing scalarization functions

- If we do allow stochastic policies,
  - There is a nice trick to combine 2 or more stationary deterministic Convex Hull policies, to the optimal undominated policies:
  - Consider a mixture policy: stochastically select one of the deterministic policies to follow.
  - e.g. a mixture policy $\pi_m$, of a policy $\pi_1$ with value $(3, 0)$, and another policy $\pi_2$ with value $(0, 3)$, will yield the following value:

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

depending on the value of $p_1$. 
Values of mixture policies?
Monotonically increasing scalarization: single policy

- One stochastic and/or non-stationary policy
- If stochasticity is allowed: one mixture policy
Monotonically increasing scalarization: summary

- Stochastic policies: convex hull of deterministic stationary policies + mixture policies
- Deterministic policies: Pareto front of deterministic non-stationary policies
### Summary

<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Stochastic</th>
<th>Deterministic</th>
<th>Stochastic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single policy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(known weights)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Linear scalarization</td>
<td>One deterministic stationary policy (1)</td>
<td></td>
<td>Convex hull of deterministic stationary policies (2)</td>
<td></td>
</tr>
<tr>
<td>Monotonically increasing</td>
<td>One deterministic policy (3)</td>
<td>One mixture policy of two or more deterministic stationary policies (4)</td>
<td>Pareto front of deterministic non-stationary policies (5)</td>
<td>Convex hull of deterministic stationary policies (6)</td>
</tr>
<tr>
<td>Scalarization</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Autonomous Agents 2: Multi-objective decision problems*
Methods: inner loop versus outer loop

- **Inner loop**
  - Perform a series of multi-objective operations (e.g. Bellman backups)
  - Typically by adapting operators of a single objective method (e.g., value iteration)

- **Outer loop**
  - Use a single objective method as a subroutine
  - Solve as a series of single-objective problems
(Inner loop) Methods

- Planning Multiple policies
  - Convex Hull value iteration (Barret and Narayanan (2008))
  - CON-MOMDP (deterministic stationary Pareto Front) (Wiering and De Jong (2007))
  - (hypervolume) Monte-Carlo Tree Search (PF) (Wang and Sebag (2012))
  - Various LP methods

- Learning
  - Model-based (Lizotte et al. (2010) and Lizotte et al. (2012))
  - Policy search (e.g. Policy gradient (SP), evolutionary methods (PF))
  - TD methods (e.g. Hiraoka et al. (2009), Mukai et al. (2012))
  - Pareto Q-learning (Van Moffaert and Nowé (2014)) (model-based?)
Contents

- Why Multi-Objective?
- MOMDPs
- Motivating Scenarios
- Taxonomy
- Methods
  - Optimistic linear support
    - Relation with POMDPs
    - Linear support
    - Optimistic CCS
    - $\varepsilon$-CCSs
Optimistic Linear Support

- Outer loop for finding a CCS (lossless subset of the CH)
- Generic multi-objective method
- Repeatedly calls a single-objective solver
- Inherits quality bounds from single-objective method
Relation with POMDPs

Piece-wise linear and convex scalarized value function:

$$V_{CCS}^*(w) = \max_{\pi \in CCS} w \cdot V^\pi$$
Linear Support

- Algorithm from POMDP literature
- Can be adapted to multi-objective setting
  - Beliefs $\rightarrow$ weight vectors
  - $\alpha$-vectors $\rightarrow$ value vectors
- Find the CCS without enumerating all policies, by solving scalarized instances at specific weight vectors $\mathbf{w}$
Linear Support

Autonomous Agents 2: Multi-objective decision problems
Find the CCS without enumerating all policies

1. Start with an empty set of value vectors $S$
2. Put the extrema of the weight simplex, $(0, 1)$ and $(1, 0)$, on a queue $Q$
3. While $Q$ is not empty
   - Solve scalarized instances at every weight vector in $Q$
   - Add solutions to $S$
   - Calculate new corner weights (where the values intersect) and add them to $Q$
Optimistic CCS
Use a priority queue with $\Delta$ as priority.
Theorem

During execution of OLS, S is an $\varepsilon$-CCS with $\varepsilon \leq \Delta(w_1)$, where $w_1$ is the corner weight with the highest priority in Q.
Reusing value functions found at earlier iterations

- Observation: when two weights are close, the scalarized values are probably close
- Observation: if we can just check whether the value at a given weight is still optimal, we might be done immediately
- Optimistic Linear Support with Alpha Reuse (OLSAR) for MOPOMDPs
OLS

- OLS is a generic multi-objective method
  - MOMDPs
  - MOPOMDPs
  - Multi-objective coordination graphs
- Can produce the CCS without enumerating all possible policies
- Can produce an $\varepsilon$-CCS (much faster)
- Works with any exact single-objective solver
- Extension for approximate solver (AOLS)
- Reusing value functions from earlier iterations makes OLS much faster (OLSAR)
Concluding

- Thank you for your attention,
- If you are interested in doing a project and/or master thesis on the subject of multi-objective decision problems, please contact us.
Further reading

- **MOMDPs:**

- **OLS**
  - Diederik M. Roijers, Shimon Whiteson, and Frans Oliehoek — Point-Based Planning for Multi-Objective POMDPs. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015. To Appear.