Hierarchical Linearly-Solvable Markov Decision Problems

Anders Jonsson & Vicenç Gómez

Dept. Information and Communication Technologies
Universitat Pompeu Fabra

Overview

1 Motivation

2 Preliminaries
- Hierarchical Reinforcement Learning
- Linearly-Solvable Markov Decision Processes

3 Contributions
- Transition-dependent rewards
- Hierarchical LMDPs
- Intra-Task Learning
- Experiments

4 Conclusion

Motivation

- Hierarchical reinforcement learning (HRL) simplifies the learning problem by decomposing it into subtasks
- Linearly-solvable Markov decision processes (LMDPs) enable faster learning since the Bellman equation is linear
- **Idea:** Combine the HRL and LMDP frameworks, taking advantage of the benefits of each

Overview

1 Motivation

2 Preliminaries
- Hierarchical Reinforcement Learning
- Linearly-Solvable Markov Decision Processes

3 Contributions
- Transition-dependent rewards
- Hierarchical LMDPs
- Intra-Task Learning
- Experiments

4 Conclusion

Markov Decision Process (MDP)

- Tuple $M = \langle S, A, P, R \rangle$:
  - S: set of states
  - A: set of actions
  - $P : S \times A \times S \to [0, 1]$: transition probabilities
  - $R : S \times A \to \mathbb{R}$: expected reward
- Bellman optimality equation:

$$V(s) = \max_{a \in A} [R(s, a) + \sum_{s'} P(s'|s, a) V(s')]$$

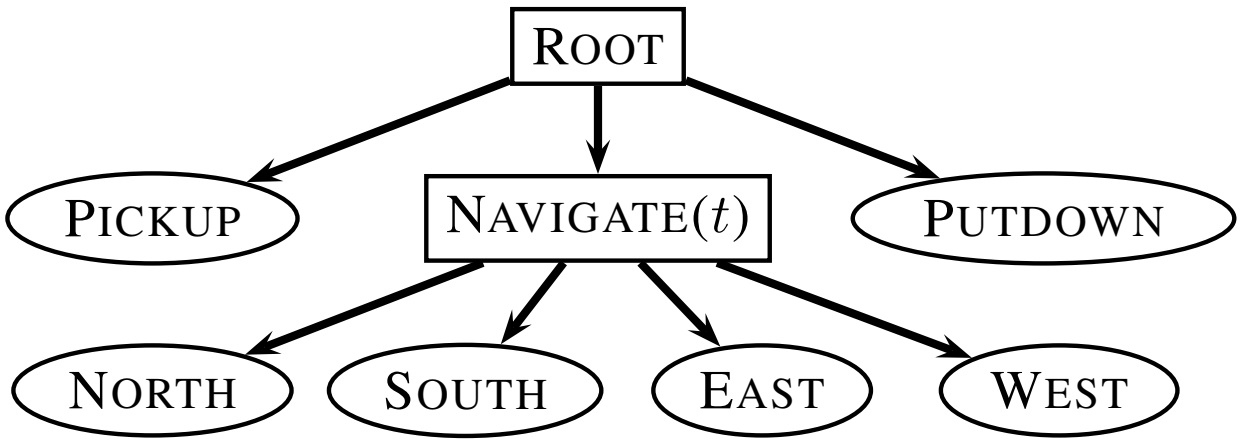
$$Q(s, a) = R(s, a) + \sum_{s'} P(s'|s, a) [\max_{a'} Q(s', a')]$$

Q-learning

- Maintain and update an estimate $\hat{Q}$ of the optimal action-value Q
- Record transition (s_t, a_t, r_t, s_{t+1}) using *any* policy
- Update rule:

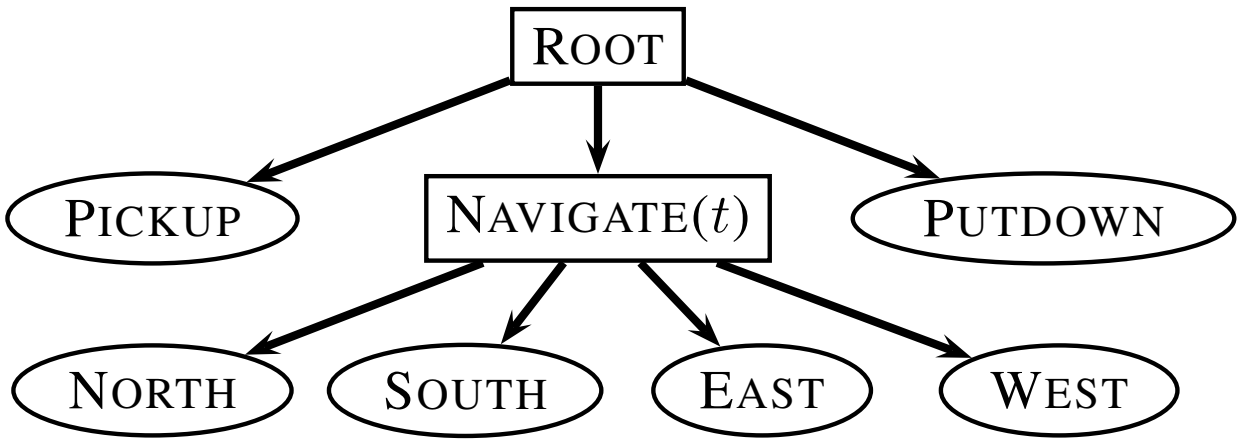
$$\hat{Q}(s_t, a_t) \leftarrow (1 - \alpha)\hat{Q}(s_t, a_t) + \alpha(r_t + \max_a \hat{Q}(s_{t+1}, a))$$

MAXQ Decomposition [Dietterich 2000]



- Decompose an MDP M into a finite set of tasks $\mathcal{M} = \{M_0, \ldots, M_n\}$ with M_0 as root
- Each task M_i has a subtask set $A_i \subseteq \mathcal{M}$

MAXQ Decomposition [Dietterich 2000]



- Decompose an MDP M into a finite set of tasks $\mathcal{M} = \{M_0, \ldots, M_n\}$ with M_0 as root
- Each task M_i has a subtask set $A_i \subseteq \mathcal{M}$
- Bellman optimality equation for M_i decomposes as

$$V_i(s) = \max_{M_j \in A_i} \left\{ V_j(s) + \sum_{s'} P(s'|s, M_j) V_i(s') \right\}.$$

Linearly-Solvable MDP (LMDP) [Todorov 2006]

- Tuple $L = \langle S, P, R \rangle$:
  - S: set of states
  - $P : S \times S \to [0, 1]$: uncontrolled transition probabilities
  - $R : S \to \mathbb{R}$: expected reward
- **Assumption 1**: Controls are transition probabilities $a(s'|s)$
- Control must be consistent with P: $a(s'|s) > 0 \to P(s'|s) > 0$

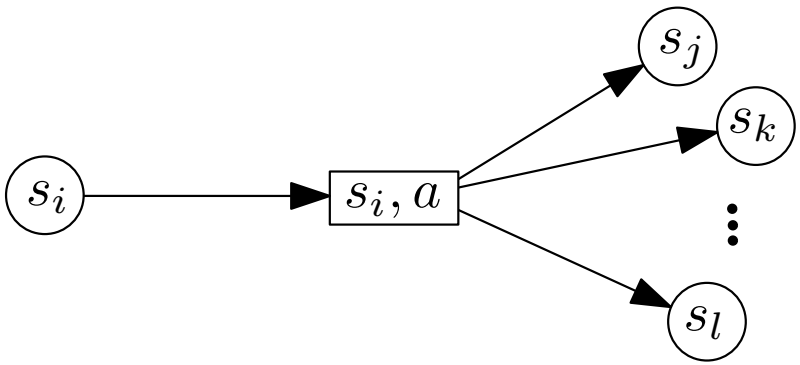
Linearly-Solvable MDP (LMDP) [Todorov 2006]

- Tuple $L = \langle S, P, R \rangle$:
  - S: set of states
  - $P : S \times S \to [0, 1]$: uncontrolled transition probabilities
  - $R : S \to \mathbb{R}$: expected reward
- **Assumption 1**: Controls are transition probabilities $a(s'|s)$
- Control must be consistent with P: $a(s'|s) > 0 \to P(s'|s) > 0$
- **Assumption 2**: Immediate reward has a particular form:

$$\mathcal{R}(s, a) = R(s) - \lambda \cdot \mathrm{KL}(a(\cdot|s) \| P(\cdot|s))$$

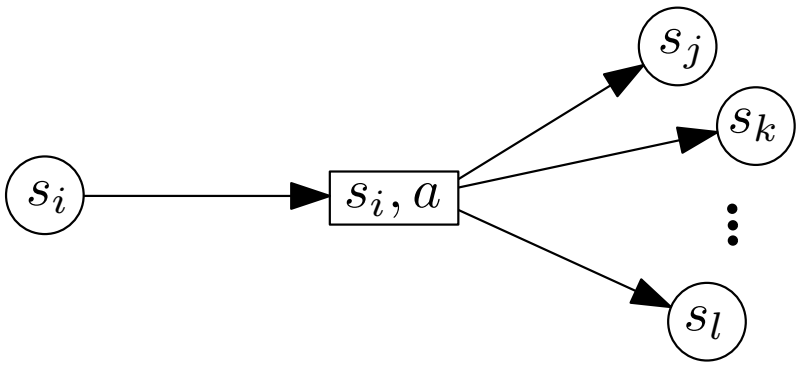
Interpretation of LMDPs

- LMDPs can be interpreted in two alternative ways:
  1. Stochastic and fully-controllable environment, unique control *a*

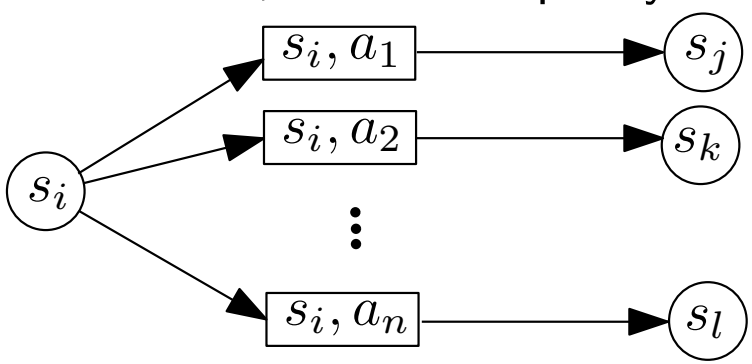


Interpretation of LMDPs

- LMDPs can be interpreted in two alternative ways:
  1. Stochastic and fully-controllable environment, unique control *a*



  2. Deterministic environment, stochastic policy *a*



Solving LMDPs

- Given $Z(s) = e^{V(s)/\lambda}$, Bellman optimality equation is

$$Z(s) = e^{R(s)/\lambda} \sum_{s'} P(s'|s) Z(s') = e^{R(s)/\lambda} \mathcal{G}[Z](s)$$

- Optimal control given by $a^*(s'|s) = P(s'|s) Z(s') / \mathcal{G}[Z](s)$
- Z can be fully computed using the power iteration method

Z-Learning

- Model-free approach to estimating $\hat{Z}$ (inducing control $\hat{a}$)
- Sample transition (s_t, r_t, s_{t+1}) using P, update estimate as

$$\hat{Z}(s_t) \leftarrow (1-\alpha)\hat{Z}(s_t) + \alpha e^{r_t/\lambda}\hat{Z}(s_{t+1})$$

Z-Learning

- Model-free approach to estimating $\hat{Z}$ (inducing control $\hat{a}$)
- Sample transition (s_t, r_t, s_{t+1}) using P, update estimate as

$$\hat{Z}(s_t) \leftarrow (1-\alpha)\hat{Z}(s_t) + \alpha e^{r_t/\lambda}\hat{Z}(s_{t+1})$$

- Importance sampling: sample using $\hat{a}$, modify update as

$$\hat{Z}(s_t) \leftarrow (1-\alpha)\hat{Z}(s_t) + \alpha e^{r_t/\lambda}\hat{Z}(s_{t+1})w_{\hat{a}}(s_t, s_{t+1}),$$
$$w_{\hat{a}}(s_t, s_{t+1}) = \frac{P(s_{t+1}|s_t)}{\hat{a}(s_{t+1}|s_t)}$$

- Note that importance sampling requires access to P

Overview

1 Motivation

2 Preliminaries
- Hierarchical Reinforcement Learning
- Linearly-Solvable Markov Decision Processes

3 Contributions
- Transition-dependent rewards
- Hierarchical LMDPs
- Intra-Task Learning
- Experiments

4 Conclusion

Contributions

- Adapt LMDPs to transition-dependent rewards
- Formulate framework for hierarchical LMDPs based on MAXQ
- Adapt intra-task learning to hierarchical LMDPs
- Validate hierarchical LMDPs in empirical experiments

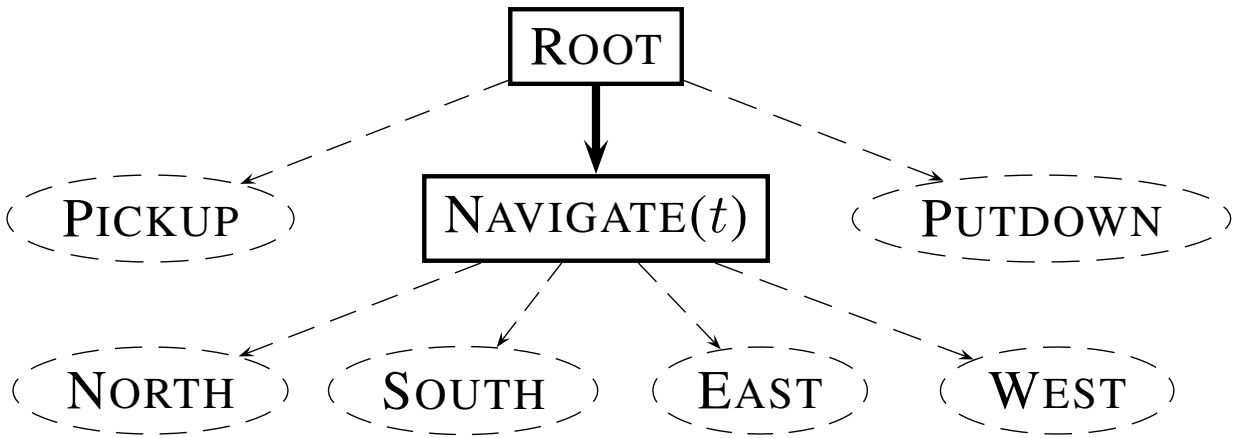
Transition-dependent rewards

- Reward depends on next state, i.e. $R : S \times S \to \mathbb{R}$
- Bellman optimality equation becomes

$$Z(s) = \sum_{s'} P(s'|s) e^{R(s,s')/\lambda} Z(s') = \mathcal{G}[Z](s).$$

- Optimal control is $a^*(s'|s) = P(s'|s) e^{R(s,s')/\lambda} Z(s') / \mathcal{G}[Z](s)$
- Update rules for Z-learning and importance sampling are the same as before

Hierarchical LMDP



- Decompose LMDP L into set of tasks $\mathcal{L} = \{L_0, \ldots, L_n\}$
- Task L_i: LMDP that terminates in unique state $t_i(s)$ $\forall s \in S$
- Transitions P_i include transitions from L and subtasks $A_i \subseteq \mathcal{L}$
- Reward R_i decomposes as $R_i(s, t_j(s)) = V_j(s)$ for each $L_j \in A_i$

Intra-Task Learning

- Each task L_i maintains its own estimate $\hat{Z}_i$ (inducing $\hat{a}_i$)
- Idea: update $\hat{Z}_i$ using transitions sampled from other tasks $\hat{a}_j$:

$$\hat{Z}_i(s_t) \leftarrow (1-\alpha)\hat{Z}_i(s_t) + \alpha e^{r_t/\lambda}\hat{Z}_i(s_{t+1})w_{\hat{a}_i}(s_t, s_{t+1})$$

Intra-Task Learning

- Each task L_i maintains its own estimate $\hat{Z}_i$ (inducing $\hat{a}_i$)
- Idea: update $\hat{Z}_i$ using transitions sampled from other tasks $\hat{a}_j$:

$$\hat{Z}_i(s_t) \leftarrow (1-\alpha)\hat{Z}_i(s_t) + \alpha e^{r_t/\lambda}\hat{Z}_i(s_{t+1})w_{\hat{a}_i}(s_t, s_{t+1})$$

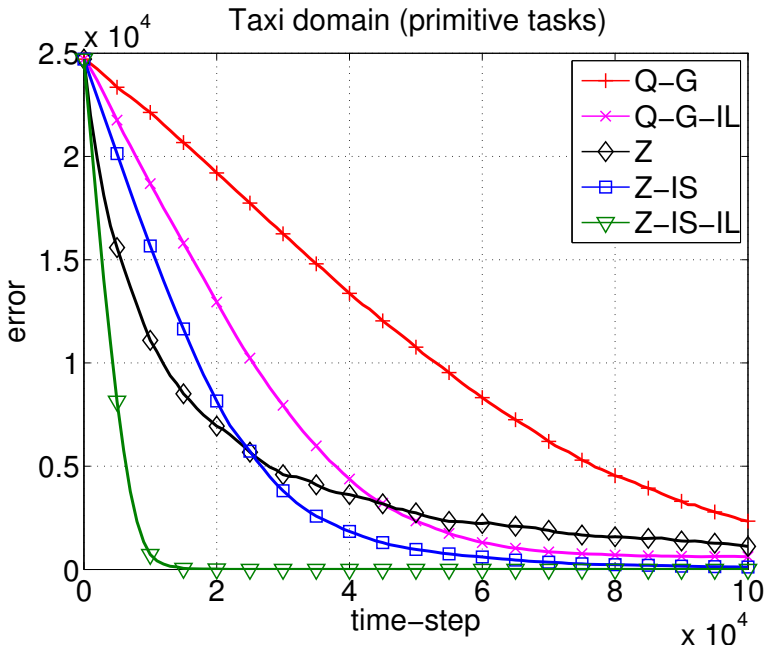
- Wrong correction $w_{\hat{a}_i}(s_t, s_{t+1})$! However, expanding yields

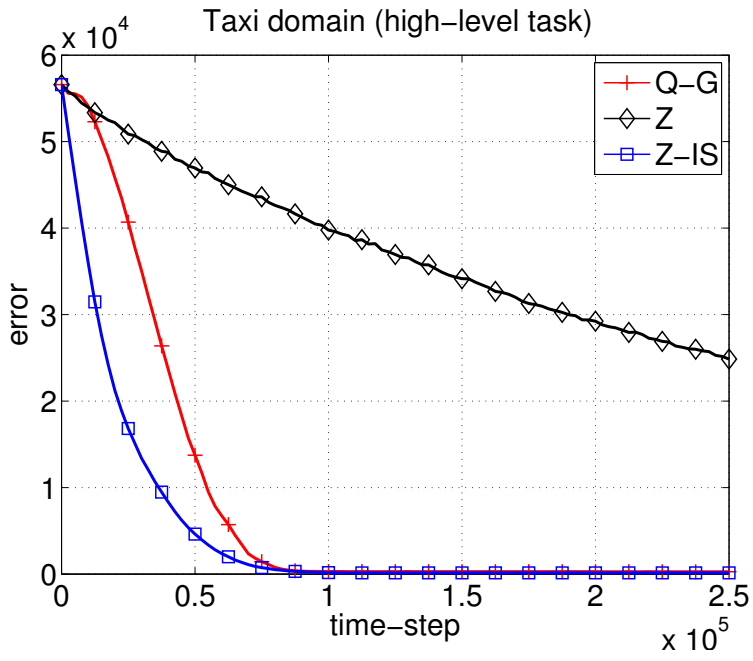
$$\begin{aligned}
\hat{Z}_i(s_t) \leftarrow & \ (1-\alpha)\hat{Z}_i(s_t) + \alpha e^{r_t/\lambda}\hat{Z}_i(s_{t+1})\frac{P(s_{t+1}|s_t)}{\hat{a}_i(s_{t+1}|s_t)} \\
= & \ (1-\alpha)\hat{Z}_i(s_t) + \alpha e^{r_t/\lambda}\frac{P(s_{t+1}|s_t)\hat{Z}_i(s_{t+1})}{P(s_{t+1}|s_t)\hat{Z}_i(s_{t+1})}G[\hat{Z}_i](s_t) \\
= & \ (1-\alpha)\hat{Z}_i(s_t) + \alpha e^{r_t/\lambda}G[\hat{Z}_i](s_t)
\end{aligned}$$

Experiments

- Two domains: Taxi domain [Dietterich 2000] and autonomous guided vehicle (AGV) domain [Ghavamzadeh et al. 2007]
- Construct MDP and LMDP with the same optimal value function
- Compare Z-learning with Q-learning
- *State abstraction*: ignore irrelevant state variables

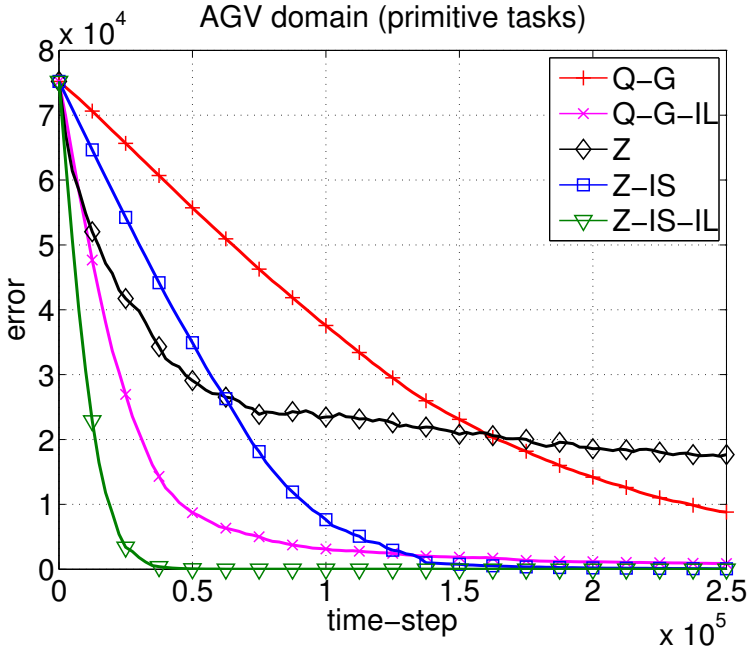
Results in Taxi Domain

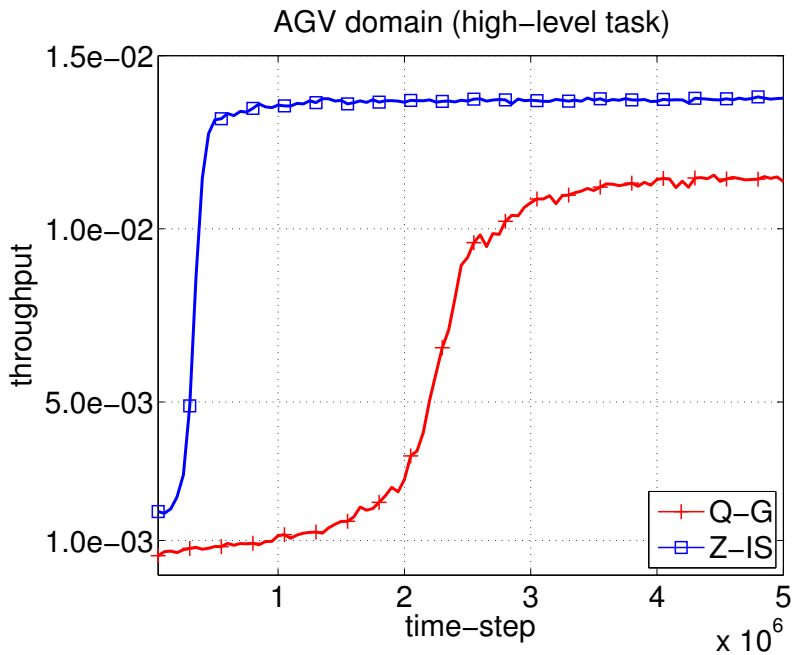




- Q-G: ϵ-greedy Q-learning
- Q-G-IL: ϵ-greedy Q-learning with intra-task learning
- Z: Regular Z-learning
- Z-IS: Z-learning with importance sampling
- Z-IS-IL: Z-learning, importance sampling & intra-task learning

Results in AGV Domain





Overview

1 Motivation

2 Preliminaries
- Hierarchical Reinforcement Learning
- Linearly-Solvable Markov Decision Processes

3 Contributions
- Transition-dependent rewards
- Hierarchical LMDPs
- Intra-Task Learning
- Experiments

4 Conclusion

Conclusion

- Novel framework that combines MAXQ decomposition and linearly-solvable MDPs, forming hierarchical LMDPs
- Benefits from the same properties that LMDPs enjoy
- Addresses the curse of dimensionality for LMDPs
- Hierarchical, intra-task Z-learning outperforms state-of-the-art HRL methods