Reasoning and Learning in Real-World Systems for Long-Term Autonomy (LTA)
Papers from the AAAI 2018 Fall Symposium

Policy Networks for Reasoning in Long-Term Autonomy

Kyle Hollins Wray and **Shlomo Zilberstein**
College of Information and Computer Sciences, University of Massachusetts, Amherst, MA, USA

Abstract

Policy networks are graphical models that integrate decision-making models. They allow for multiple Markov decision processes (MDPs) that describe distinct focused aspects of a domain to work in harmony to solve a large-scale problem. This paper presents the formalization of policy networks and their use in modeling reasoning tasks necessary for scalable long-term autonomy. We prove that policy networks generalize a wide array of previous models, such as options and constrained MDPs, which can be equivalently viewed as the integration of multiple models. To illustrate the approach, we apply policy networks to the challenging real world domain of robotic home health care. We demonstrate the benefits of policy networks on a real robot and show how they facilitate scalable integration of multiple decision-making models.

Introduction

Over the past decade, sequential decision-making models have been increasingly deployed in large-scale domains with high societal impact, ranging from aircraft collision avoidance (Kochenderfer 2015) to autonomous vehicles (Wray, Witwicki, and Zilberstein 2017). While these systems have enjoyed rapid growth, they relied on a fragmented collection of specialized approaches that combine either multiple objectives (Altman 1999; Klein et al. 2012) or multiple models by hierarchical abstraction (Sutton, Precup, and Singh 1999; Pineau et al. 2003) or by integrating their actions online (Bai et al. 2015; Wray, Witwicki, and Zilberstein 2017).

Each one of these solutions introduces an important reasoning capability, but to support long-term autonomy in the real world, we increasingly need to integrate multiple capabilities within one system. As Marvin Minsky observed, "the power of intelligence stems from our vast diversity, not from any single, perfect principle" (Minsky 1986). It is unlikely that any single MDP model will suffice. For the sake of scalability and computational efficiency, we need new formal ways to facilitate the integration of multiple models within a single agent. To this end, we propose a novel framework called policy networks that unifies prior approaches, offers new insights, and provides a solid foundation on which to build the next generation of large-scale decision models.

We consider a home healthcare robot domain for household and eldercare scenarios (Pineau et al. 2003). The robot must perform a wide array of helpful tasks (e.g., medicine delivery and cleaning), plan safe paths around the house, and detect falls to call for help as needed (Broadbent et al. 2009) while operating over a long duration. This domain has many subproblems, each complex and nuanced, and they are all interrelated as part of the whole solution. Systems of this scale require an integration of many methods, as they quickly become too large to solve with a single monolithic MDP.

Prior work on integrations of multiple models arose from disparate ideas, each of which extends the MDP in a particular way. For hierarchies, a large problem is decomposed into essentially subtasks (Sutton, Precup, and Singh 1999; Tao et al. 2009). For multiple objectives, which we show is related, solutions typically scalarize the objectives into one or maximize a primary objective subject to constraints (Roijers, Whiteson, and Oliehoek 2014; Klein et al. 2012). For online solutions with multiple models, each update their state spaces simultaneously and recommend actions for each entity in the domain (Kochenderfer 2015; Bai et al. 2015).

While these approaches have been used in modest applications, it is not clear how they relate or how to combine them to solve large-scale problems. This knowledge gap manifests itself by the lack of a unified view across all model forms, which leaves many questions unanswered. For example, how are constrained MDPs (CMDPs) related to options and can they be combined? What does it mean to perform an action if it can induce updates in multiple models? How can multiple models transfer control to one another if their state and/or action spaces are different? More generally, how can we create a principled mathematical framework that enables the integration of multiple models with these concepts?

We propose the notion of a policy network that helps us begin to answer these questions. It is a graph in which the vertices denote a set of policies and the edges denote their dependencies. A set of policies associated with a vertex refers to a state and action space that can be shared or distinct from any other vertices. A policy constraint edge enforces a restriction on a vertex's set of policies from another vertex. A policy transition edge defines a state transition in a vertex's state space or a transfer of control to another vertex. The objective is to maximize expected reward in the induced hierarchy of constrained semi-Markov decision processes following the graph's dependency structure.

Our primary contributions also form the sections of the paper: (1) a formal definition of policy networks and their properties; (2) a theoretical analysis that proves their generality, encapsulating prior models such as CMDPs and options; and (3) an implementation of the approach in a home healthcare robot acting in a real household environment.

Background

A *multi-objective Markov decision process (MOMDP)* is defined by $\langle S, A, T, \mathbf{R} \rangle$. S is a set of states. A is a set of actions. $T : S \times A \times S \to [0,1]$ is a state transition such that $T(s,a,s') = Pr(s'|s,a)$. $\mathbf{R} = [R_0, \dots, R_k]^T$ is a vector of reward functions such that $R_i : S \times A \to \mathbb{R}$. A *policy* may be stochastic $\pi : S \times A \to [0,1]$ or deterministic $\pi : S \to A$. Π refers to any such set of policies. For a stochastic policy π, infinite horizon objectives, with discount factor $\gamma \in [0,1)$, have a *value* $\mathbf{V}^\pi : S \to \mathbb{R}^{k+1}$ for state s defined as:

$$\mathbf{V}^\pi(s) = \sum_{a \in A} \pi(s,a) \Big(\mathbf{R}(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \mathbf{V}(s') \Big).$$

A *constrained MDP (CMDP)* (Altman 1999) is a MOMDP with the objective:

$$\begin{array}{ll} \text{maximize} & V_0^\pi(s^0) \\ \text{subject to} & -V_i^\pi(s^0) \leq c_i, \quad \forall i \in \{1, \dots, k\} \end{array} \tag{1}$$

and given constraints $c_i \in \mathbb{R}$. Rewards are commonly treated as costs $R_i = -C_i$ for each V_i. An *optimal* policy π^* obtains a maximal value $V_0^*(s^0)$.

A *semi-Markov decision process (SMDP)* (Puterman 1994) is an MDP in which control of the system is sojourn, relinquished to a so-called *natural process*, defining distinct *decision epochs* each with a *sojourn time*—duration the agent was not in control. The discrete time SMDP may be defined by $\langle S, A, T, F, R, \rho \rangle$. $T : S \times A \times \mathbb{N} \times S \to [0,1]$ is a state transition that includes the sojourn time τ such that $T(s,a,\tau,s') = Pr(s'|s,a,\tau)$. $F : S \times A \times \mathbb{N} \to [0,1]$ is the probability mass function (PMF) for the sojourn time τ such that $F(s,a,\tau) = Pr(\tau|s,a)$. $\rho : S \times A \times \mathbb{N} \to \mathbb{R}$ is the expected reward rate for the sojourn time. For a deterministic policy π, the *value* $V^\pi : S \to \mathbb{R}$ is defined as:

$$V^\pi(s) = \mathcal{R}(s, \pi(s)) + \sum_{s' \in S} \sum_{\tau=1}^{\infty} \gamma^\tau Pr(\tau, s' | s, \pi(s)) V^\pi(s'),$$

with state transition $Pr(\tau, s'|s,a)$ and:

$$\mathcal{R}(s,a) = R(s,a) + \sum_{\tau'=1}^{\infty} F(s,a,\tau') \sum_{\tau=1}^{\tau'-1} \rho(s,a,\tau).$$

Notationally, continuous time SMDPs define F as the cumulative distribution function (CDF) to facilitate its integral and derivative equations, whereas we equivalently define F as its PMF to facilitate discrete time equations (Howard 1971).

The options framework (Sutton, Precup, and Singh 1999) add special actions to an MDP, called options, that execute a complete policy, performing the actions of the agent until it stochastically returns control. It forms a special type of discrete time SDMP defined by $\mathcal{O} = \{\mathcal{O}_1, \dots, \mathcal{O}_k\}$ with $\mathcal{O}_i = \langle \mathcal{I}_i, \pi_i, \beta_i \rangle$. For *Markov options*, $\mathcal{I}_i \subseteq S$ is a set of admissible initiation states, $\pi_i : S \to A$ is a policy, and $\beta_i : S \to [0,1]$ is the probability of terminating the option at each state. *Semi-Markov options* are instead defined over histories $\bar{H}$—sequences $\bar{h} = \langle s^0, a^0, s^1, a^1, \dots \rangle$—with $\pi_i : \bar{H} \to A$ and $\beta_i : \bar{H} \to [0,1]$.

An *infinite horizon MDP* is generalized by both: (1) a CMDP with no constraints ($k = 0$), and (2) a discrete time SMDP that immediately returns control ($F(s,a,1) = 1$).

Policy Networks

In general, policy networks are graphs in which vertices denote sets of policies for a reward function and edges denote policy dependences among them. The objective is to capture the relations among distinct decision-making components to solve large multi-objective hierarchical problems.

Specifically, a **policy network** is a sequential decision-making model defined by a directed graph $\langle V, E \rangle$:

- V is a finite set of vertices such that each $v \in V$ denotes a set of policies Π_v for reward $R_v : S_v \times A_v \to \mathbb{R}$; and
- E is a finite set of edges such that each $\langle v, w \rangle = e \in E$ is:
  - policy constraint Π_e enforces $\Pi_w \subseteq \Pi_e$; and/or
  - policy transition T_e defines $T_e : S_v \times A_v \times S_w \to [0,1]$ as a partial function equal to $Pr(w, s'_w | s_v, a_v)$.

The execution of a policy network operates over discrete time steps $t \in \mathbb{N}$ as a form of *Markov reward process*. Each vertex v has a state space S_v and action space A_v for its policy and reward. Both, one, or neither space might be shared by any number of other vertices—that is, the same random variable in *dynamic Bayesian networks* or the same uncertainty or decision nodes *influence diagrams*. Each unique state space, say S_v for vertex v, has an initial state $s_v^0 \in S_v$.

As in (PO)MDPs, to **perform** an action is simply the act of conditioning on the action so as to induce an update in the underlying vertex v's stochastic process following the **state transition** distribution $Pr(w, s'_w | s_v, a_v)$. This probability distribution describes the state transition within the state space of v (i.e., $w = v$ and $s'_w = s'_v \in S_v$) as well as across other state spaces used by other vertices (i.e., $w \neq v$ and $s'_w \in S_w$). Policy networks require full specification of $Pr(w, s'_w | s_v, a_v)$, via the collection of functions T_e. In its simplest form, if v only transitions to itself by $T_e = T_{vv}$, then T_e is equivalent to a typical (PO)MDP state transition. Any induced state transition also induces a reward from R_v.

At each time step, any **controller** vertex v performs the action $\pi_v(s_v) \in A_v$ at their current state $s_v \in S_v$ from available policy $\pi_v \in \Pi_v$. Actions performed by v may result in a state transition to a different vertex w's state space. We call this a **transfer of control**, with the controller changing from v to w who now performs the action $\pi_w(s_w) \in A_w$ at their state $s_w \in S_w$ from available policy $\pi_w \in \Pi_w$. Each policy network has an initial controller $v^0 \in V$.

From an initial controller, we derive a graph describing the direct constraint or transfer dependencies among two vertices. Formally, the **dependency graph** $\langle V, E' \rangle$ is a directed acyclic graph (DAG) with $E' \subseteq E$ ensuring all paths from each vertex lead to the initial controller v^0. Generally, we consider any such DAG that contains a maximal number of edges E' from E. A common special case is simply the *shortest path tree*. Following the dependency graph, we can denote the **ancestors** of vertex v as $\mathcal{A}(v) \subset V$, its **parents** as $\mathcal{P}(v) \subseteq \mathcal{A}(v)$, its **descendants** as $\mathcal{D}(v) \subset V$, and its **children** as $\mathcal{C}(v) \subseteq \mathcal{D}(v)$.

Hierarchy of Constrained Semi-MDPs

A policy network induces a dependency graph that induces a hierarchy of constrained semi-Markov decision processes (CSMDP). Each CSMDP is recursively dependent, starting at the furthest vertices and ending at the initial controller.

Relative Times In discrete time SMDPs (Puterman 1994), there are three notions of time which policy networks share. These are relative to a vertex v, but subscripts will be omitted when it is not ambiguous. First, the **natural process time** is denoted by $\tau \in \mathbb{N}$. It refers to the total number of time steps following the state transitions for v's CSMDP. Second, a **decision epoch** is denoted by $t \in \mathbb{N}$. It refers to a time interval within the natural process time for v's CSMDP: $[\tau^1 + \cdots + \tau^t, \tau^1 + \cdots + \tau^{t+1})$. Third, a **sojourn time** is denoted by $\tau^t \in \mathbb{N}$ for decision epoch t. It refers to the duration of a decision epoch. This notation is overloaded, as it refers to both this duration and the random variable that determines this duration (F detailed below). In summary, vertex v's relative decision epochs are when it is a controller, and its relative sojourn times are the duration between being in control.

Relative State Let ancestors that share v's action space be $\overline{\mathcal{A}}(v) = \{w \in An(v) | A_w = A_v\}$. Only these ancestors can directly affect v's state, because if one were to become a controller then it would perform action, which by definition would induce state updates in v. Variables which have this consideration use the same notation, such as sojourn time $\overline{\tau}^t$.

Since each decision epoch's sojourn time reflects a vertex v's perspective on the effects ancestors have on its Markov reward process, its CSMDP omits the time spent by ancestors which do not share its action space. Consequently, the CSMDP state must consider which relevant ancestor is in control $w \in \overline{\mathcal{A}}(v)$, its state $s_w \in S_w$, and the vertex's own current successor state s'_v. Formally, this is defined by $\overline{s}_v = \langle w, s_w, s'_v \rangle$ with all such states denoted as $\overline{S}_v$. This space represents the CSMDP's states while the natural process—ancestors of v—is in control.

Relative State Transitions The relative CSMDP stochastic process follows $Pr(\overline{\tau}, \overline{s} | s_v, a_v)$. It refers to the probability that the next decision epoch for v occurs at or before sojourn time $\overline{\tau}$ and has successor state s'_v, with ancestor w's state s_w, after v had performed action a_v in state s_v.

Additionally, as in discrete time SMDPs, we may write $Pr(\overline{\tau}, \overline{s} | s_v, a_v) = F_v(s_v, a_v, \overline{\tau}) T_v(s_v, a_v, \overline{\tau}, \overline{s})$ given the sojourn time distribution following F_v and the state transition distribution following T_v. Note that as in general SMDPs, there are equivalent representations of $Pr(\overline{\tau}, \overline{s} | s_v, a_v)$ using different definitions of F_v and T_v, which may be used here as well if desired.

We can compute both $F_v(s_v, a_v, \overline{\tau})$ and $Pr(\overline{s} | s_v, a_v, \overline{\tau})$ directly by constructing a Markov chain $M^{|\overline{S}| \times |\overline{S}|}$ from the policy network and its dependency graph. The former is computed by marginalization over the Markov chain's transitions to states of the form $\langle v, s_v, s_v \rangle$, that is, transitioning back to v after sojourn time $\overline{\tau}$ from $M^{\overline{\tau}}$. The latter is computed directly following $M^{\overline{\tau}}$, as the $\overline{\tau}$-th iteration of the Markov chain determines these probabilities. There are two important properties regarding Markov chain M. First, as the state only needs to consider ancestors $\overline{\mathcal{A}}(v)$, all other ancestors $\mathcal{A}(v) - \overline{\mathcal{A}}(v)$ are summarized by their corresponding transfer control probabilities. Second, all policy transitions T_e outside of v's dependency graph are treated as absorbing states in v's CSMDP. This naturally captures the construction of abstractions with collections of subtasks: a subtask's goal or terminal states transfer control back to its parent after it completes the task it was designed to handle.

Relative Rewards The discrete time CSMDP has two rewards: (1) immediate reward R_v, and (2) expected reward gained at rate $\rho_v : S_v \times A_v \times \mathbb{N} \to \mathbb{R}$. Specifically, following SMDPs, $\rho_v(s_v, a_v, \overline{\tau})$ is defined as the reward rate at some sojourn time $\overline{\tau}$ after action a_v was performed in state s_v, but before the next action is performed. Thus, we have:

$$\rho_v(s_v, a_v, \overline{\tau}) = \gamma_v^{\overline{\tau}} \sum_{\overline{s} \in \overline{S}} R_v(s'_v, \pi_w(s_w)) Pr(\overline{s} | s_v, a_v, \overline{\tau}), \quad (2)$$

with a discount factor $\gamma_v \in [0, 1)$. The resulting reward is denoted $\mathcal{R}_v : S_v \times A_v \to \mathbb{R}$ and is written as:

$$\mathcal{R}_v(s_v, a_v) = R_v(s_v, a_v) + \sum_{\overline{\tau}'=1}^{\infty} F_v(s_v, a_v, \overline{\tau}') \sum_{\overline{\tau}=1}^{\overline{\tau}'-1} \rho_v(s_v, a_v, \overline{\tau}). \quad (3)$$

Relative Constraints As this is a *constrained* SMDP, the space of available policies *used to perform action* is restricted by policy constraint edges. Formally, each vertex v refers to a policy space such as $\Pi_v \subseteq \{\pi : S_v \to A_v\}$. This can be all possible policies or any non-empty subset, such as in the options framework or CMDPs. Options define $\Pi_v = \{\pi_v\}$ as a single fixed policy that *is* the option itself π_v (Sutton, Precup, and Singh 1999). CMDPs define $\Pi_v \subset \bigcap_i \Pi_{iv}$ by a restriction in this policy space by each other objective i's policy set Π_{iv} that satisfies constant c_i (Altman 1999).

In general, the set Π_v is defined iteratively, by computing the policy sets for all ancestors in $\mathcal{A}(v)$, ending with v's own edge constraint Π_{vv}, if any exists. Formally, Π_v ensures:

$$\pi_v \in \Pi_v \subseteq \Pi_{vv} \subseteq \bigcap_{w \in \mathcal{P}(v)} \Pi_{wv} \quad (4)$$

with v's final chosen policy being any $\pi_v \in \Pi_v$, for any Π_{vv} and Π_{wv} that exist in E.

To focus our discussion, we consider policy constraint edges Π_e in terms of a bound on regret from expected value (defined in the next section) up to a slack—allowable deviation from optimal. Formally, policy constraint edge $e = \langle v, w \rangle$ has a slack $\delta_{vw} \geq 0$ such that:

$$\Pi_{vw} = \{\pi \in \Pi_v | V_w^*(s_w^0) - V_w^{\pi}(s_w^0) \leq \delta_e\}. \quad (5)$$

While we focus on CMDPs in this paper, the lexicographic MDP (LMDP) (Wray, Zilberstein, and Mouaddib 2015), LPOMDP (Wray and Zilberstein 2015a), and possibly other forms of slack (Wray, Kumar, and Zilberstein 2018) can be generalized by policy networks as well. However, we leave this discussion to future work.

Objective Function

The **infinite horizon** objective of a policy network is defined recursively such that each vertex v's objective to find a policy $\pi_v \in \Pi_v$ that maximizes the expected reward starting from initial state s_v^0 subject to its ancestors:

$$\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma_v^t \mathcal{R}_v(s_v^t, \pi_v(s_v^t))\Big| \pi_v, s_v^0, \forall w \in \mathcal{A}(v), \pi_w, \Pi_w\Big] \quad (6)$$

with s_v^t denoting the random variable for the state of v at its decision epoch t generated following the policy network's dependency graph state transitions $Pr(\overline{\tau}, \overline{s}|s_v, a_v)$, and the stationary ancestor policies π_w and policy sets Π_w. To focus the discussion, we consider an infinite horizon; however, other objectives follow in the natural way.

For a policy $\pi_v \in \Pi_v$, the **value** $V_v^\pi : S_v \rightarrow \mathbb{R}$ is the expected reward at state s_v following the Bellman equation:

$$V_v^\pi(s_v) = \mathcal{R}_v(s_v, \pi_v(s_v)) + \sum_{\overline{s}\in\overline{S}} \sum_{\overline{\tau}=1}^{\infty} \gamma^{\overline{\tau}} Pr(\overline{\tau}, \overline{s}|s_v, \pi_v(s_v)) V_v^\pi(s_v'). \quad (7)$$

A policy $\pi_v^* \in \Pi_v$ is **optimal** if it obtains the maximal value V_v^*. This optimal value can be computed by the Bellman optimality equation over each state s_v:

$$V_v^*(s_v) = \max_{a_v \in A_v} \Big(\mathcal{R}_v(s_v, a_v) + \sum_{\overline{s}\in\overline{S}} \sum_{\overline{\tau}=1}^{\infty} \gamma^{\overline{\tau}} Pr(\overline{\tau}, \overline{s}|s_v, a_v) V_v^*(s_v')\Big). \quad (8)$$

Stationarity

For a controller v and its policy set Π_v, the chosen policy $\pi_v^t \in \Pi_v$ for performing actions at time t can remain constant or vary over time. Formally, if $\pi_v^t = \pi_v$ for all time t, then we call the vertex's **policy stationary**, otherwise it is **non-stationary**. In the tradition of MDPs and planning, policies are commonly stationary. However, even solutions computed offline can have time-varying non-stationary policies. The holistic perspective policy networks affords a broader view of stationarity that includes to the behavior of *online* algorithms as well, which vary their policy over time in online planning (Ye et al. 2017) and reinforcement learning (Sutton and Barto 1998). The analysis is specific to the online scenario and therefore we leave this analysis to future work.

Since policy networks relate sets of policies to one another, the set of policies Π_v^t at a time t can also remain constant or vary over time. Formally, if $\Pi_v^t = \Pi_v$ for all time t, then we call the vertex's **policy set stationary**, otherwise it is **non-stationary**. In a simple MDP or POMDP within a policy network, the policy set trivially remains constant. However, in a growing number of online *models*—such as MODIA used in autonomous vehicles (Wray, Witwicki, and Zilberstein 2017)—the set of policies is constantly adjusted online. While this is easily described in a policy network, their formal analysis is nuanced and specific to the assumptions for each online scenario. For this reason, we leave any such extensive analysis to future work, favoring a description of one such online model in terms of policy networks to illustrate the approach.

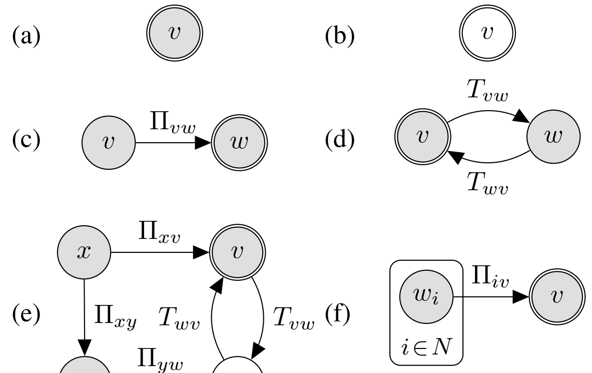


Figure 1: Basic examples of the graphical notation used to represent policy networks, with each $v \in V$ following some $v \sim$ *MDP*(S_v, A_v, T_v, R_v): (a) stationary vertex; (b) non-stationary vertex; (c) constraint edge; (d) a transfer of control; (e) a mixture of these previous concepts; and (f) plate notation denoting a set of N constraints.

Graphical Representation

Policy networks continue the tradition set by probabilistic graphical models (PGMs) with a clean and powerful graphical representation. This allows for rich complex decision-making with many objectives and levels of abstraction to be easily described, analyzed, and implemented.

Figure 1 covers basic policy network notation. Each vertex as $v \in V$ is a circle and each directed edge $e \in E$ as an arrow. Edges are directed and denote their policy dependencies by any relevant variables. For example, policy constraints are denoted by their policy set Π_e and policy transitions are denoted by their function T_e. When it is not ambiguous, it is suffices to denote parameters instead, such as slack δ_e or an options' initiation set I_e or termination function β_e. Vertices and edges are lowercase; their sets are uppercase. The initial controllers v^0 are denoted by double-lined circles. Stationary vertices are filled-in—e.g., solved by *offline* algorithms. In contrast, non-stationary vertices which are *not* filled-in—e.g., solved by *online* algorithms. Plate notation may be used to easily group sets of similarly defined vertices.

Any vertex v which follows a standard MDP, POMDP, etc. model uses the notation $v \sim$ *MDP*$(\cdot)$, $v \sim$ *POMDP*$(\cdot)$, etc. in the tradition of PGMs. Intuitively, the "$\sim$" symbol refers to "selecting" a policy from a policy space. This notation is used for convenience. It completely describes the vertex's policy set Π, reward R, and an implicit self-loop edge with transition T and constraint Π^*—enforcing only optimal policies, as applicable. Formally, this extra notation means $v \sim$ *MDP*(S, A, T, R) defines v with policies $\Pi \subseteq \{\pi : S \rightarrow A\}$ with reward $R : S \times A \rightarrow \mathbb{R}$. Additionally, it defines the *implicit edge*—that is, merely not graphically drawn—self-looping edge $e = \langle v, v\rangle \in E$ with policy constraint $\Pi \subseteq \Pi_e = \Pi^*$ and policy transition $T_e = T$. POMDPs are similarly defined as $v \sim$ *POMDP*(S, A, Ω, T, O, R) since they are a special form of continuous state MDP called a *belief MDP* (Kaelbling, Littman, and Cassandra 1998).

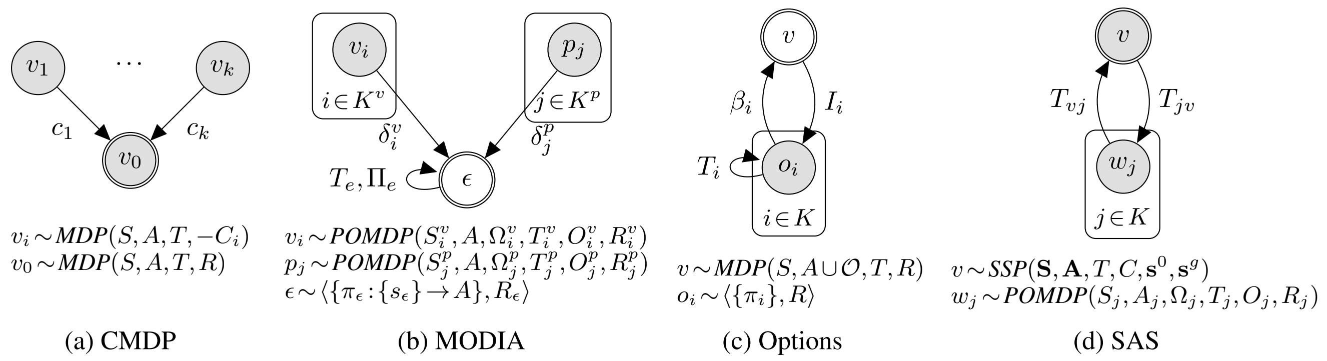

v_1
$\cdots$
v_k
c_1
c_k
v_0
$v_i \sim$ *MDP*$(S,A,T,-C_i)$
$v_0 \sim$ *MDP*(S,A,T,R)
(a) CMDP
v_i
$i \in K^v$
p_j
$j \in K^p$
δ_i^v
δ_j^p
T_e, Π_e
ϵ
$v_i \sim$ *POMDP*$(S_i^v, A, \Omega_i^v, T_i^v, O_i^v, R_i^v)$
$p_j \sim$ *POMDP*$(S_j^p, A, \Omega_j^p, T_j^p, O_j^p, R_j^p)$
$\epsilon \sim \langle \{\pi_\epsilon : \{s_\epsilon\} \to A\}, R_\epsilon \rangle$
(b) MODIA
v
β_i
I_i
T_i
o_i
$i \in K$
$v \sim$ *MDP*$(S, A \cup \mathcal{O}, T, R)$
$o_i \sim \langle \{\pi_i\}, R \rangle$
(c) Options
v
T_{vj}
T_{jv}
w_j
$j \in K$
$v \sim$ *SSP*$(\mathbf{S}, \mathbf{A}, T, C, \mathbf{s}^0, \mathbf{s}^g)$
$w_j \sim$ *POMDP*$(S_j, A_j, \Omega_j, T_j, O_j, R_j)$
(d) SAS

Figure 2: Four policy networks: (a) a constrained MDP; (b) MODIA; (c) the options framework, traditionally for a reinforcement learning agent; and (d) semi-autonomous systems, as a macro-action or subtask example.

Theoretical Analysis

We now show the generality of policy networks by proving that they can encapsulate various models such as the options framework and CMDPs. Additionally, this section also serves as a demonstration of the design of policy networks to provide guidance for how to create them. Proposition 1 begins with a simple but important statement regarding policy networks' generality beyond MDPs and POMDPs.

Proposition 1. *Policy networks generalize (PO)MDPs.*

Proof. For any MDP, we must construct an equivalent policy network. Let $V = \{v\}$ with $v \sim \langle \Pi, R \rangle$. Let $E = \{e = \langle v, v \rangle\}$ with transition $T_e = T$ and $\Pi_e = \{\pi | V^*(s^0) = V^\pi(s^0)\}$. Thus, $Pr(\overline{\tau}, \overline{s}|s,a) > 0$ only if $\overline{\tau} = 1$ and $Pr(1, \overline{s}|s,a) = T(s,a,s')$, also implying $\mathcal{R}(s,a) = R(s,a)$. Thus, Equation 7 becomes the MDP Bellman equation. □

Next, we consider two related cases of *policy constraint edges*: CMDPs and MODIA in Propositions 2 and 3, respectively. Here, constraints limit the space of policies from parent vertices to a child controller vertex. CMDPs represent a policy network with a shared both state and action space, constrained offline with stationary policies. MODIA represents a policy network with a different state space but a shared action space—illustrating how performing action can simultaneously affect many models—constrained online with non-stationary policy sets.

Proposition 2. *Policy networks generalize CMDPs.*

Proof. For any CMDP, we must construct an equivalent policy network. See Figure 2 (a). Let $V = \{v_0, \ldots, v_k\}$ with $v_0 \sim$ *MDP*(S,A,T,R_0), $v_i \sim$ *MDP*$(S,A,T,-C_i)$, $v^0 = v_0$, and stochastic policies. Let $\{\langle v_i, v_0 \rangle\} \subset E$ with $\Pi_{i0} = \{\pi | V_i^*(s^0) - V_i^\pi(s^0) \leq \delta_i\}$ and $\delta_i = V_i^*(s^0) + c_i$. Thus, the policy network has the same objective as the CMDP objective from Equation 1, and Equation 7 equal to the CMDP Bellman equation. □

MODIA (Wray, Witwicki, and Zilberstein 2017) is for autonomous vehicle (AV) decision-making about other entities: vehicles K^v and pedestrians K^p, analyzed a posteriori. Multiple POMDP models (i.e., v_i and p_j) describe each AV-entity pairwise interaction; each model is solved offline in isolate, resulting in stationary polices π_i^v and π_j^p. An *executor* $\epsilon : A^* \to A$ maps any tuple of action recommendations to a final action performed by the executor. This action updates the other models resulting in regret; e.g., for $i \in K^v$ regret is $r_i = V_i^*(b_i) - Q_i^*(b_i, \epsilon(\mathbf{a}))$. Following Wray *et al.* (2017), we consider risk-sensitive MODIA with LEAF, which assumes an ordering exists over actions $\succ$ in terms of safety. If a riskier action is performed $\pi_i(b_i) \succ \epsilon(\mathbf{a})$ then the model i experiences a regret lower bounded by $r_i \geq Q_i^*(b_i) - \underline{Q}$. A so-called LEAF executor that selects the safest action among the recommendations (i.e., $\forall i, \epsilon(\mathbf{a}) \succeq \pi_i(b_i)$ and $\exists j$ s.t. $\epsilon(\mathbf{a}) = \pi_j(b_j)$) minimizes the sum of one step regrets.

Proposition 3. *Policy networks generalize MODIA.*

Proof. For any MODIA, we must construct an equivalent policy network. See Figure 2 (b). Let $V = \{\epsilon\} \cup \{v_i\} \cup \{p_j\}$ with $v^0 = \epsilon$. Let each $v_i \sim$ *POMDP*$(\cdot)$ and $p_j \sim$ *POMDP*$(\cdot)$ as in the figure with shared A. Let $\epsilon \sim \langle \Pi_\epsilon, R_\epsilon \rangle$ for policies $\pi_\epsilon : S_\epsilon \to A$ with trivial state space $S_\epsilon = \{s_\epsilon\}$. Let $R_\epsilon(s_\epsilon, a)$ equal the index of a in the *reverse* of ordering $\succ$ over A. Let $\{\langle \epsilon, \epsilon \rangle\} \cup \{\langle v_i, \epsilon \rangle\} \cup \{\langle p_j, \epsilon \rangle\} \subset E$. Let $T_{\epsilon\epsilon}(s_\epsilon, \cdot, s_\epsilon) = 1$ be a self-loop transition and let $\Pi_{\epsilon\epsilon} = \{\pi | V_\epsilon^*(s_\epsilon) = V_\epsilon^\pi(s_\epsilon)\}$ select optimal executor policies. Let Π_ϵ be a non-stationary policy set that is defined by its parents' constraint edges $\Pi_{i\epsilon}^t = \{\pi : \{s_\epsilon\} \to A\} | V_i^*(b_i^t) - Q_i^*(b_i^t, \pi(s_\epsilon)) < \delta_i^v\}$, with a similarly defined $\Pi_{j\epsilon}$, that only allow executor actions with regret no greater than $\delta_i^v = V_i^*(b_i^t) - \underline{Q}$. By construction, the executor vertex ϵ's selected policy $\pi_\epsilon^* \in \Pi_\epsilon^t$ at time t maps its state s_ϵ to the safest action among recommendations. This is identical to the definition of LEAF, and thus minimizes the sum of one step regrets implicitly through constraints. □

Lastly, we consider two related cases of *policy transition edges*: the options framework and SAS in Propositions 4 and 5, respectively. Here, transfer of control happens between parent and child vertices, both online (options) and offline (SAS). Options represent a policy network with a shared state space and shared action space, learning online with a non-stationary policy. SAS represents a policy network with different state space and different action space—illustrating how how different models can interact—planning offline with a stationary policies.

Proposition 4. *Policy networks generalize options.*

Proof. For any set of options, we must construct an equivalent policy network. See Figure 2 (c). First, we consider Markov options. Let $V = \{v\} \cup \{o_i\}$ with $v^0 = v$. Let $v \sim$ *MDP*$(S, A \cup \mathcal{O}, T, R)$ and for each option $\mathcal{O}_i = \langle \mathcal{I}_i, \pi_i, \beta_i \rangle$ let $o_i \sim \langle \Pi_i, R \rangle$ with only the option policy $\Pi_i = \{\pi_i\}$. As v is traditionally a reinforcement learning agent, we simply represent the vertex's policy as non-stationary. Let R include the option reward $R(s, \mathcal{O}_i) = R(s, \pi_i(s))$. Let $\{\langle v, o_i \rangle\} \cup \{\langle o_i, v \rangle\} \cup \{\langle o_i, o_i \rangle\} \subset E$. Let $T_{vi}(s_v, \mathcal{O}_i, s'_i) = T(s_v, \pi_i(s_v), s'_i)[s_v \in \mathcal{I}_i]$ transfer control to the option, as allowed by $\mathcal{I}_i$, using Iverson bracket $[\cdot]$. Without loss of generality in MDPs, actions can be defined for each state $A(s_v)$, handling any invalid execution of options in states. Let $T_{iv}(s_i, a_i, s_v) = \beta_i(s_i) T(s_i, \pi_i(s_i), s_v)$ transfer control back to v stochastically following β_i; consequently, $T_i(s_i, a_i, s'_i) = (1 - \beta_i(s_i)) T(s_i, a_i, s'_i)$ accounts for β_i, too.

Semi-Markov options have a similar structure, except the shared state space is $\bar{H}$ instead of S, with $v \sim$ *MDP*$(\bar{H}, A \cup \mathcal{O}, T_v, R_v)$ and $o_i \sim \langle \Pi_i, R_v \rangle$. When v is in control, its transitions T remain at zero time states: $T_v(\bar{h}_v, a, \bar{h}'_v) = T(s_v, a, s'_v)[\bar{h}' = \langle s'_v \rangle]$ and $R_v(\bar{h}, a) = R(s^t_v, a)$ for any $\bar{h} = \langle s^0_v, a^0, \ldots, a^{t-1}, s^t_v \rangle$, again with $R_v(\bar{h}, \mathcal{O}_i) = R(s^t_v, \pi_i(s^t_v))$. When options are in control, they transition over time by simply recording its action $\pi_i(s^t_i)$ and stochastic state s^t_i to iteratively construct a history embedded in its state space $\bar{H}$: $T_i(\bar{h}_i, a, \bar{h}'_i) = (1 - \beta_i(s^t_i)) T(s^t_i, a, s'_i)[\bar{h}' = \langle s^0_i, \ldots, s^{t-1}_i, a, s'_i \rangle]$. Lastly, transfer of control is similarly defined. Let $T_{vi}(\bar{h}_v, \mathcal{O}_i, \bar{h}'_i) = T(s^t_v, \pi_i(s^t_v), s'_i)[s^t_v \in \mathcal{I}_i]$ simply transfer to the option following $\mathcal{I}_i$. Let $T_{iv}(\bar{h}_i, a, \bar{h}_v) = \beta_i(s^t_i) T(s^t_i, a, s^0_v)$ transfer back to v following β_i, with it resetting time encoded in the state $\bar{h}_v$ for v.

In both cases, the value equation follows the option framework's discrete time SMDP Bellman equation. □

Semi-autonomous systems (SAS) (Wray, Pineda, and Zilberstein 2016) model the transfer of control of a single agent among a group of *actors* I that control it, such as transferring control between an AV and a human driver. It is built on a two-level hierarchy with a stochastic shortest path (SSP) problem (Bertsekas and Tsitsiklis 1991) reasons about transfer of control success and failure by executing a POMDP. The SAS state space $\mathbf{S} = S \times I$ includes the current actor, and the action space $\mathbf{A} = A \times I$ includes the desired next actor. The state transition $T : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \to [0, 1]$ follows the current actor's state transition $T_i : S \times A \times S \to [0, 1]$. However, if a transfer is attempted at $\mathbf{s} = \langle s, i \rangle$, we multiply by $\rho : \mathbf{S} \times \mathcal{I} \times \mathcal{I} \to [0, 1]$ as $\rho(\mathbf{s}, \hat{i}, i') = Pr(i' | \mathbf{s}, \hat{i})$ denotes the probability that the next actor is i' given an attempt to transfer from i to $\hat{i}$. For each state-action pair, ρ is computed by a POMDP in a completely different state and action space that considers communication messages and belief about the state of the actor's preparedness to take control. Its execution ends in a collapsed absorbing belief state: success b^s, failure b^f, or abort b^a. A mapping $f : \{b^s, b^f, b^a\} \to I$ must be provided from these POMDP result states to the next SSP actor. Given its policy, the probability of reaching these three absorbing states is computed as ρ for use by the SSP.

Proposition 5. *Policy networks generalize SAS.*

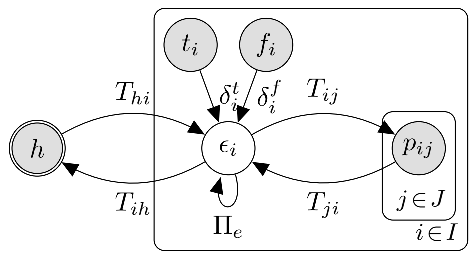
Proof. For any SAS, we must construct an equivalent policy network. See Figure 2. Let $V = \{v\} \cup \{w_j\}$ with $v \sim$ *SSP*$(\cdot)$ and $w_j \sim$ *POMDP*$(\cdot)$ as in the figure with distinct state and action. Without loss of generality and to remain inline with SAS, an SSP is used, which is akin to an MDP with discount $\gamma = 1$, initial state $\mathbf{s}^0$, and goal state $\mathbf{s}^g$. For each SSP state-action pair there is a distinct POMDP w_j, with $K = \{j \in \mathbf{S} \times \mathbf{A} | j = \langle \mathbf{s}, \mathbf{a} \rangle \wedge i \neq \hat{i}\}$. Let $\{\langle v, w_j \rangle\} \cup \{\langle w_j, v \rangle\} \subset E$. For T_{vj} and T_{jv}, we have the notation $j = \langle \mathbf{s}, \mathbf{a} \rangle$, $\mathbf{s} = \langle s, i \rangle$, $\mathbf{a} = \langle a, \hat{i} \rangle$, and $\mathbf{s}' = \langle s', i' \rangle$. First, let $T_{vj}(\mathbf{s}, \mathbf{a}, b^0_j) = 1$ be defined for any $i \neq \hat{i}$, always executing the corresponding POMDP starting at its b^0_j. Second, T_{jv} has three cases: for each $b \in \{b^s_j, b^f_j, b^a_j\}$ we have $T_{jv}(b, a_j, \mathbf{s}') = T_i(s, a, s')[f(b) = i']$. Since the action spaces are different ($\mathbf{A} \neq A_j$), v's CSMDP summarizes the state transitions of each w_j, resulting in v's Bellman equation producing a ρ for each w_j. This is identical to the SAS model. Thus, this policy network produces the same Bellman equations for the SAS's SSP and POMDPs. □

Evaluation: Home Healthcare Robot

Home healthcare robots serve in household and eldercare scenarios, providing solutions to a wide array of helpful tasks ranging from cleaning to medicine delivery (Robinson, MacDonald, and Broadbent 2014). Surveys conducted by Broadbent *et al.* (2009) analyzed and ranked the desired tasks the robot could do to help improve the lives of the elderly. Both elderly people and healthcare staff were surveyed. The top ranked needs include: (1) medicine notification and delivery; (2) forms of cleaning the house; and (3) monitoring, detecting, and helping with falls. We focus on a robot solution that captures all three.

The few mobile healthcare robots that exist tend towards hand-engineered decision-making systems that work well for their specific implementation (Graf, Hans, and Schraft 2004). One of the notable exceptions of a general model-based approach involves an early form of hierarchical POMDP (Pineau et al. 2003). They partitioned a single POMDP's action space into smaller groups, solving a collection of identical POMDPs with differing reduced action spaces. While successful, this early seminal work lacked the generality of a policy network—such handling multiple objectives in CMDPs, use different models (state and action spaces), or leverage grounding in SMDPs such as options.

Problem Description Consider a healthcare robot with a set of *high-level tasks* it must continuously complete. The *medicate task* is selected to complete by the high-level and requires navigating to the bathroom, retrieving medicine, finding the patient, and delivering it to them. The *clean task* is also selected and requires moving any out-of-place objects back in place while vacuuming. The *monitor task* must operate at all times, reactively interrupting any other task, and requires monitoring and detecting a fall of an elderly person. If confident in the detection, the robot should check on the person and autonomously call for the help of a healthcare professional. The *low-level path planning* must take special care to avoid obstacles to safely traverse the house.



$h \sim \mathit{MDP}(S^h, A^h, T^h, R^h)$
$t_i \sim \mathit{POMDP}(S_i^t, A_i^t, \Omega_i^t, T_i^t, O_i^t, R_i^t)$
$f_i \sim \mathit{POMDP}(S^f, A^f, \Omega^f, T^f, O^f, R^f)$
$\epsilon_i \sim \langle \{\pi_i^\epsilon : \{s_\epsilon, s_c\} \rightarrow A_i^t \cup A^f\}, R_i^\epsilon \rangle$
$p_{ij} \sim \mathit{Harmonic}(S^p, A^p, O^p, G_{ij}^p)$

Figure 3: The policy network for the home healthcare robot.

Name	V	$\lvert S\rvert$	$\lvert A\rvert$	$\lvert\Omega\rvert$	$\lvert\Gamma\rvert$	Time
High-Level Task Selector	h	16	4	—	—	<0.1
Medicate Task	t_1	289	13	5	1224	159.8
Clean Task	t_2	145	13	3	612	14.0
Fall Monitor/Assist Task	f_i	2	3	2	22	<0.1
Low-Level Path Planner	p_{ij}	17766	9	—	—	0.92

Table 1: The problem sizes and run times for each model.

Policy Network Solution Figure 3 shows the policy network for the healthcare robot. We provide a description of each vertex below. Also, Table 1 shows the problem sizes and results of solving these problems using *nova* (Wray and Zilberstein 2015b), with value iteration (VI), point-based VI (PBVI) (Pineau, Gordon, and Thrun 2006), and harmonic functions (Wray et al. 2016), for MDP, POMDP, and path planning models, respectively. PBVI has a policy size denoted as $|\Gamma|$. Harmonic function path planning ($\mathit{Harmonic}(\cdot)$ above) is equivalent to a special class of SSP with uniform state transitions, goals $G_{ij}^p \subset S^p$, and cost of 1 for obstacles $O^p \subset S^p$. All source code will be provided for complete problem descriptions and reproducibility.

The high-level task h handles issues $I = \{t_1, t_2, f_i, \emptyset\}$ with $S^h = 2^I$ and $A^h = I$. Let $\emptyset$ denote a "complete" or "no-op" state and action here. The high-level h transfers control by T_{hi} to the start state of the corresponding task when selected as an action. Let a set of regions $\mathcal{R}$ (e.g., kitchen, bathroom, and bedroom) be given for the map. The medicate task t_1 has $S_1^t = \mathcal{R} \times \mathcal{R} \times \{Y, N\} \cup \{\emptyset\}$, denoting region locations for the robot and the person, as well as if the medicine is carried or not. $A_1^t = \mathcal{R}$ refer to navigation to a region by the path planner by T_{ij} and T_{ji}. $\Omega_2^t = \{Y, N\} \times \{Y, N\} \cup \{\emptyset\}$ refers to detection of a person or not, holding medicine or not, and completion. The clean task t_2 is similarly defined with $S_2^t = \mathcal{R} \times \mathcal{R} \cup \{\emptyset\}$ and $A_2 = \mathcal{R}$ as it searches for a location to clean. $\Omega_2^t = \{Y, N, \emptyset\}$ refers to detection of a person or not and completion. The monitor task f_i has state space $S^f = \{Y, N\}$ for if the person has fallen and needs help. $A^f = \{\mathit{call}, \mathit{ask}, \emptyset\}$ denotes calling for help, asking if the person needs help, and no-op. $\Omega^f = \{Y, N\}$ refers to detecting a fall or not. The executor ϵ follows as in a MODIA, with a preference for call and ask over region navigation actions $\mathcal{R}$. T_{ih} transfers control back to h when in a task complete state s_c. The path planner p_{ij} navigates between regions $J = \mathcal{R} \times \mathcal{R}$ in the occupancy grid map in Figure 4.

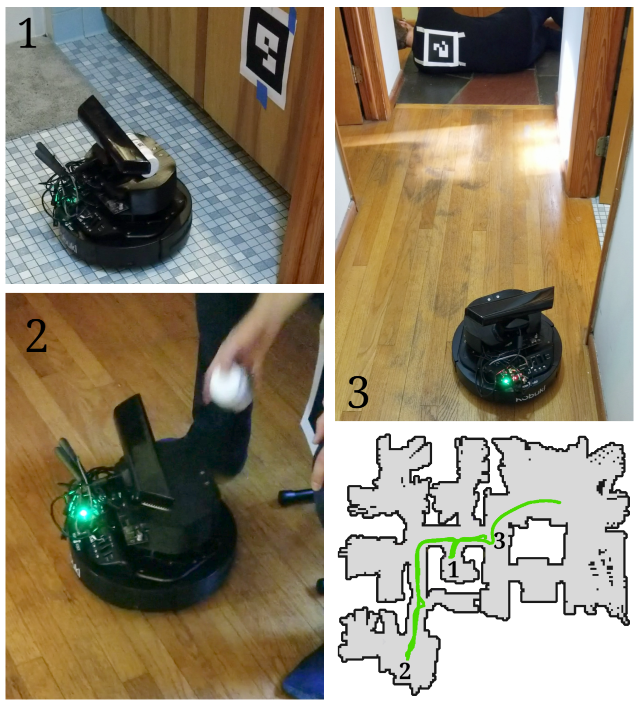


Figure 4: Experiments with the home healthcare robot using this policy network in the real household shown above. Three highlights are shown: (1) medicine retrieval for task t_1, (2) medicine delivery completion with transfer $t_1 \rightarrow h \rightarrow t_2$, and (3) interruption of cleaning task t_2 by detecting a fall with task f_i and calling for assistance.

Conclusion

We now revisit the questions posed in the introduction. First, how are CMDPs related to options and can these two models be combined? In policy networks, they are different types of edges between collections of distinct models: policy constraints and policy transitions. By simply adding any desired vertices and corresponding edges, we can easily combine both ideas, as evident by the previous section. Second, what does it mean to perform an action? In policy networks, it means conditioning on an action so as to induce a state update in any models that share the same action space. Third, how do these operate when the state and/or action spaces are different? Following the definition of performing an action, any shared action space induces state updates in the collection of models, as in options or more generally SMDPs. With different action spaces, the policies can still affect one another through transfer of control, treated as an abstraction or macro-action.

Finally, is there a principled mathematical model that enables the integrated design of multiple models with these concepts? Policy networks serve as an answer to this. They provide a theoretical model that generalizes select state-of-the-art models. The implementation shown here demonstrates they can successfully model and solve a challenging home healthcare robot domain. In summary, policy networks provide a general model for the reasoning component in real-world systems for long-term autonomy.

Acknowledgments

We thank the reviewers for their helpful comments. This research was supported by the National Science Foundation grant number IIS-1724101.

References

Altman, E. 1999. *Constrained Markov decision processes*. England: Chapman & Hall/CRC Press.

Bai, H.; Cai, S.; Ye, N.; Hsu, D.; and Lee, W. S. 2015. Intention-aware online POMDP planning for autonomous driving in a crowd. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 454–460.

Bertsekas, D. P., and Tsitsiklis, J. N. 1991. An analysis of stochastic shortest path problems. *Mathematics of Operations Research* 16(3):580–595.

Broadbent, E.; Tamagawa, R.; Kerse, N.; Knock, B.; Patience, A.; and MacDonald, B. 2009. Retirement home staff and residents' preferences for healthcare robots. In *Proceedings of the 18th IEEE International Symposium on Robot and Human Interactive Communication*, 645–650.

Graf, B.; Hans, M.; and Schraft, R. D. 2004. Care-o-bot ii—development of a next generation robotic home assistant. *Autonomous Robots* 16(2):193–205.

Howard, R. A. 1971. *Dynamic Probabilistic Systems: Semi-Markov and Decision Processes*. New York, NY: Wiley.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1):99–134.

Klein, L.; young Kwak, J.; Kavulya, G.; Jazizadeh, F.; Becerik-Gerber, B.; Varakantham, P.; and Tambe, M. 2012. Coordinating occupant behavior for building energy and comfort management using multi-agent systems. *Automation in Construction* 22:525–536.

Kochenderfer, M. J. 2015. *Decision Making Under Uncertainty: Theory and Application*. MIT Press.

Minsky, M. 1986. *The Society of Mind*. New York, NY: Simon and Schuster.

Pineau, J.; Montemerlo, M.; Pollack, M.; Roy, N.; and Thrun, S. 2003. Towards robotic assistants in nursing homes: Challenges and results. *Robotics and Autonomous Systems* 42(3):271–281.

Pineau, J.; Gordon, G.; and Thrun, S. 2006. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research* 27:335–380.

Puterman, M. L. 1994. *Markov decision processes: Discrete stochastic dynamic programming*. New York, NY: John Wiley & Sons.

Robinson, H.; MacDonald, B.; and Broadbent, E. 2014. The role of healthcare robots for older people at home: A review. *International Journal of Social Robotics* 6(4):575–591.

Roijers, D. M.; Whiteson, S.; and Oliehoek, F. A. 2014. Linear support for multi-objective coordination graphs. In *Proceedings of the 13th International Conference on Autonomous Agents and Multi-Agent Systems*, 1297–1304.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*. MIT Press.

Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence* 112(1-2):181–211.

Tao, Y.; Wang, T.; Wei, H.; and Chen, D. 2009. A behavior control method based on hierarchical POMDP for intelligent wheelchair. In *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, 893–898.

Wray, K. H., and Zilberstein, S. 2015a. Multi-objective POMDPs with lexicographic reward preferences. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 1719–1725.

Wray, K. H., and Zilberstein, S. 2015b. A parallel point-based POMDP algorithm leveraging GPUs. In *Proceedings of the AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 95–96.

Wray, K. H.; Ruiken, D.; Grupen, R. A.; and Zilberstein, S. 2016. Log-space harmonic function path planning. In *Proceedings of the 29th IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1511–1516.

Wray, K. H.; Kumar, A.; and Zilberstein, S. 2018. Integrated cooperation and competition in multi-agent decision-making. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 4751–4758.

Wray, K. H.; Pineda, L.; and Zilberstein, S. 2016. Hierarchical approach to transfer of control in semi-autonomous systems. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 517–523.

Wray, K. H.; Witwicki, S. J.; and Zilberstein, S. 2017. Online decision-making for scalable autonomous systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 4768–4774.

Wray, K. H.; Zilberstein, S.; and Mouaddib, A.-I. 2015. Multi-objective MDPs with conditional lexicographic reward preferences. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 3418–3424.

Ye, N.; Somani, A.; Hsu, D.; and Lee, W. S. 2017. DESPOT: Online POMDP planning with regularization. *Journal of Artificial Intelligence Research* 58:231–266.