A Multitask Representation using Reusable Local Policy Templates

Benjamin Rosman
School of Informatics
University of Edinburgh, UK
Council for Scientific and Industrial Research
South Africa
B.S.Rosman@sms.ed.ac.uk

Subramanian Ramamoorthy
School of Informatics
University of Edinburgh, UK
s.ramamoorthy@ed.ac.uk

Abstract

Constructing robust controllers to perform tasks in large, continually changing worlds is a difficult problem. A long-lived agent placed in such a world could be required to perform a variety of different tasks. For this to be possible, the agent needs to be able to abstract its experiences in a reusable way. This paper addresses the problem of online multitask decision making in such complex worlds, with inherent incompleteness in models of change. A fully general version of this problem is intractable but many interesting domains are rendered manageable by the fact that all instances of tasks may be described using a finite set of *qualitatively meaningful* contexts. We suggest an approach to solving the multitask problem through decomposing the domain into a set of capabilities based on these local contexts. Capabilities resemble the options of hierarchical reinforcement learning, but provide *robust* behaviours capable of achieving some subgoal with the associated guarantee of achieving at least a particular aspiration level of performance. This enables using these policies within a planning framework, and they become a level of abstraction which factorises an otherwise large domain into task-independent sub-problems, with well-defined interfaces between the perception, control and planning problems. This is demonstrated in a stochastic navigation example, where an agent reaches different goals in different world instances without relearning.

1 Introduction

A long-lived autonomous robot in a complex and continuously changing world is expected to make a sequence of nontrivial decisions. These decisions must be made despite incompleteness in available information and limitations on computational resources, and the robot may be required to complete tasks, without much time for exploration.

This combination of issues makes the decision problem difficult to address using standard methods such as those based on optimisation of expected reward in a Markov Decision Process (MDP). In much of the way this theory has been developed, the agent operates in *one* particular environment (at one time), specified – explicitly or implicitly – by a particular combination of transition dynamics, state-action representation and reward process. The problem of extending such work to changing environments is a topic of current research. Some interesting approaches include robust dynamic programming with respect to a bounded family of MDPs (Yu and Mannor 2009), viewing the non-stationarity as arising from mode switches within a set of MDPs (Choi, Yeung, and Zhang 1999) and modifying the value iteration process to accommodate online estimates of transition/reward statistics (Jaulmes, Pineau, and Precup 2005; da Silva et al. 2006). The more practicable of these results view the problem as one of repairing a base policy with respect to changes, which may be inadequate in the sort of domains we are interested in studying. This notion of adaptation is also related to transfer learning (Taylor and Stone 2009). In this setting, there have been successful instances of reusing learned strategies for tasks in closely related worlds, but the larger issue of adapting and constructively developing representations is largely open.

A primary difficulty for the operation of a robot in this type of problem is to make sense of the vast degree of variability it may encounter. An elegant mechanism for posing such problems with hidden or incomplete information is the partially observable Markov decision process (POMDP), or the I-POMDP in the interactive multiagent setting (Gmytrasiewicz and Doshi 2005). These models involve updating belief distributions over state space, but unfortunately the problem of coping with this richness becomes quickly intractable under such models as the state space grows (Lane and Smart 2005). The implications are that an agent attempting to model the real world must develop some succinct representation of experiences in order to quickly approximate and reason about new environments.

People, and animals, when faced with these kinds of decision problems in changing environments, seem to conquer this complexity in a way that suggests that the answer may lie not so much in faster and better algorithms but in a clever encoding of the problem. Indeed, it has long been argued that the representation of problems of reasoning about actions has a dramatic effect on problem solving efficiency (Amarel 1968; Korf 1980). More recently, it has also been shown that when faced with the problem of acting optimally in environments with significant structural uncertainty, the mathematically optimal solution may be to adopt *coarse* strategies that are better adapted to the *family* of tasks (Bookstaber and Langsam 1985). These coarse strategies are more likely to provide reusable components which are of lower

complexity than the global problem, making policy learning and instantiation faster. An excellent example of a multitask encoding that addresses some of these issues is seen in the work of Burridge, et al. (Burridge, Rizzi, and Koditschek 1999), based on the notion of sequential composition by separating local policies from a global strategy. This was a carefully hand-crafted engineering design. In this paper, we propose a way to achieve a similar effect by a method that is more constructive, based on learning from direct experience.

Our premise is that although global aspects of the decision problem are subject to continual change, many interesting instances of complex worlds are generated from a small number of *qualitatively* meaningful contexts. There is reason to believe that there is latent local structure in many domains of interest, and worlds are not completely arbitrary in their formation. A large, and potentially infinite, set of instances of a domain may be generated (at least in an approximate sense) from some finite, and hopefully small, set of prototypical contexts. Particular examples can be seen, for example, in the strategy space of the complex game of Go (Georgeot and Giraud 2011; Liu, Slotine, and Barabási 2011). More general approaches, such as belief space compression (Roy, Gordon, and Thrun 2005), suggest that such structure exists more fundamentally as an underlying manifold in many decision problems.

We introduce the idea of a *capability*, as a behaviour that an agent can instantiate in particular settings to consistently achieve some desired result. This couples the control and perception problems, so the framework can exploit the strengths of both these fields, as well as planning for structuring sequences of capabilities. Capabilities are component policies that may be composed for one-shot tasks, and are used as a basis for flexible policy composition, enabling transfer of skills between environments. We demonstrate this in a spatial navigation domain, involving random environments and stochastic agents with dynamics unknown to the learning agent. This enables a robot to quickly plug into a new task and function sensibly, by estimating coarsely optimal policies (in a component sense) on the fly in previously unexplored regions of state space using local models.

2 A Decomposition of Structure

The key element of this work is the idea of an agent learning capabilities to decompose an environment, and use that to form a set of reusable policy templates that may be composed later. A capability consists of a context and a behaviour. The context is a state abstraction, identified from the observations made by the agent. When the agent is in the appropriate context, that capability becomes active. This allows the agent to invoke the associated behaviour. Formally, define a capability C as

$$C \quad := \quad \langle \chi, \mu, p, \delta \rangle \tag{1}$$

where χ is the context under which C is applicable, μ is the policy associated with C, and p is the predicate achieved with probability $1-\delta$, by executing policy μ in χ. As such, δ is a measure of the robustness, or a performance guarantee, of the capability.

The intuition behind this is that a subset of behaviours are applicable only in certain situations, under specific conditions. When those conditions are met, the associated behaviours become viable. Furthermore, an agent finding itself in an unfamiliar situation may try to invoke behaviours from the closest matching contexts to bootstrap its knowledge of the new scenario, rather than learning from scratch.

An overview of the procedure is presented in Figure 1. This illustrates how the experiences of an agent (a mobile robot in this depiction) can be used to learn a set of prototypical contexts which represent some of the compositional elements of the structure of the world within which it finds itself. Associating learnt policies with these contexts (together with their intended outcomes and robustness measures) define a set of capabilities for the agent, which allow it to achieve some local predicate (such as navigating a local region of a map). The learning of contexts is presented in Section 3 and the association and learning of policies is covered in Section 4.

A capability can be regarded as an abstract policy, which is applicable only under particular local conditions, as defined by the context. This allows for policy reuse and skill transfer between different instances of a domain.

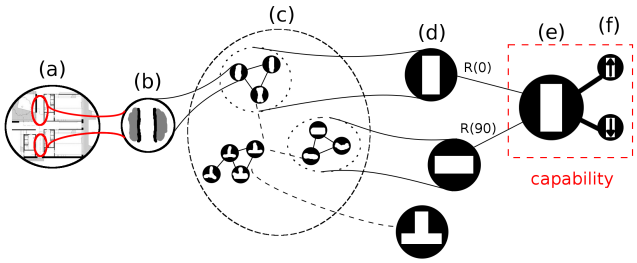

Figure 1: The representation building process. (a) A snippet from the environment containing the agent, where two instances of North-South corridors have been identified (b) State vector (sensor readings) acquired by the agent in random exploratory moves through the environment (c) Representation of contexts encountered by the agent (d) Landmark contexts extracted from the network (e) Symmetry reduced landmark set, with annotated operators (f) Policies learned offline, applicable in that context. Note that landmarks (e) and policies (f) define the basic capabilities of the agent.

As an example, consider a simple navigation domain, such as that presented in Figure 2. In this case, learning the different local structures of navigable space equips the agent with the knowledge needed to *survive* and navigate other similar worlds. The agent can thus acquire policy templates in each context which work up to a performance guarantee δ. An example of this is presented in Section 5. Furthermore, composition of these local structures provides temporal extension and allows for planning, as discussed in Section 6.

Structural context is not limited to navigation domains. Recurring motifs can be seen in other domains such as manipulation tasks, by taking into account topological descriptors of the objects in the robot's workspace (Rosman and Ramamoorthy 2011). Figure 3 provides one such example

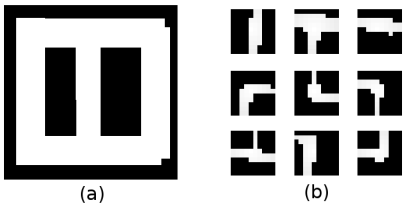

Figure 2: (a) An example navigation world, where the white regions are navigable corridors (b) The local contexts extracted automatically by the method described in Section 3

of the feature of holes in object configurations.

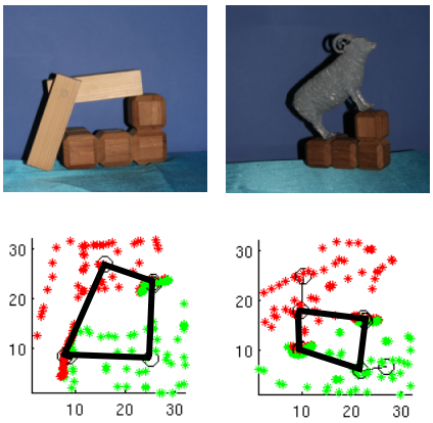

Figure 3: Two examples of topologically equivalent holes detected in scenes with multiple objects. The semantic notion of a "hole" is a contextual feature which describes the structure of the scene

Given a learnt set of capabilities $\mathcal{C}$, the agent operates by using observations o_t at time t to determine its current context χ_t. The current context is identified as the one for which some distance measure $d(\cdot,\cdot)$ from o_t is minimised, i.e.

$$\chi_t \quad = \quad \arg\min_{\chi}\{d(\chi, o_t)\}. \tag{2}$$

Capabilities which can be invoked, $\mathcal{C}_t$, are then any with a context matching χ_t:

$$\mathcal{C}_t = \{C \in \mathcal{C} | \chi_C = \chi_t\}. \tag{3}$$

The agent may then invoke any capability in $\mathcal{C}_t$, depending on which is best suited to the current task. Mechanisms for planning and a representation for composing capabilities in a temporally extended manner are discussed in Section 6.

3 Perceptual Contexts

The function of a context is to ground the behaviours of the agent in its perception, through a set of primitive templates which describe some landmarks used in the latent process which generates worlds in the domain. As a result, contexts can be learned from commonly occurring sets of states as the agent randomly explores worlds generated from the domain, much in the spirit of bootstrap learning (Pierce and Kuipers 1997).

The problem of learning contexts is one of uncovering latent structure in perceptual space in terms of learning a decomposition based on experience of observations. These are then used as a discretisation of state space, where each discrete context has an inherent meaning, represented by a capability, giving a set of policies usable in that context.

A capability is a form of partial model, which can make limited predictions about the evolution of the world. This only happens when a capability is active, which is in turn defined by the context, and when the agent is in a setting which matches that context. In the experiments reported in Section 5, the contexts are defined by particular patterns of free space around the agent. However, structure exists in many domains, such as that shown in Figure 3 where policies may correspond to grasp types.

One way to implement this, the procedure used in our experiment in Section 5, is described as follows:

1. Experience multiple instances of the domain through random exploration,
2. Collect all sensory experiences into a Growing Neural Gas (GNG) network in perception space, which is a form of self-organising map (Fritzke 1995),
3. Skeletonise GNG network into a set of landmark contexts by removing spurious edges which do not affect the topology of the network,
4. Learn and associate policies with the reduced set of contexts (see Section 4).

Using the GNG network means that the topology of the distribution of structures is modelled, and higher density regions are better represented. The procedure for acquiring the structural knowledge of contexts (with associated policies) for the navigation problem is illustrated in Figure 1. The resulting network represents the clusters in sensory space experienced by the agent, with topological connectivity between them. This method, similarly to other approaches such as vector quantisation, results in a clustering of perceptions to provide a basis of landmarks.

As mentioned previously, fitting a capability requires finding an approximate match between the current perceptions of the agent, and the context associated with that capability. If any contexts match the current perceptions, then the agent is in that context, implying the associated capability could be invoked in the current situation. If, at a given point in time, no contexts match the current perceptions of the agent ($\min_{\chi}\{d(\chi, o_t)\} \geq \tau$ for some threshold τ), then the context set is not rich enough to encapsulate the domain. In this case, the closest matching contexts could still be used, although the expected rewards of the capabilities may not be achieved. Alternatively, this indicates a situation when more learning is required. Note that context fitting and detection is based on the observation signal rather than the reward signal (da Silva et al. 2006). This is a faster and more plausible process (an agent can determine this when first entering a new context) and is a direct result of the tighter integration of perception into the framework.

4 Local Policies

Given a decomposition of the structure of the world into a set of basis contexts, the agent learns a set of local control policies associated with each of these contexts, to form capabilities. Each context may have several capabilities based on it, depending on the number of local goals which the agent may try achieve in that context.

There are different ways in which an agent could automatically determine local goals, e.g., through heuristics which identify regions of interest in a context. The approach taken in the experiment in Section 5 is that a "bag of goals" is initially provided to the agent, which is required to determine which of these are possible in each context. This set of locally possible goals are then used for training the policy set.

Learning of the policies can, in the base case, proceed with an A* algorithm or reinforcement learning techniques such as value function learning (Sutton and Barto 1998) over the reduced state space defined by χ. In contexts containing other strategic agents, game theoretic procedures such as regret minimisation or other online learning techniques (Foster and Vohra 1999) may be favourable.

Learning the policy μ occurs in the neighbourhood given by χ and provides a mechanism for the agent to achieve the predicate p from χ. In this way, a policy acts as a funnel, to move the agent from the region of state space represented by χ to a region wherein p is true. As a number of capabilities are usually associated with the same context, there are a corresponding number of funnels starting from the same region but moving the agent towards several potential outcomes (depending on the capability selected).

These policies are learnt offline, once the prototypical contexts of the world have been identified or, as in the case of lifelong learning, whenever a new context is identified. The agent determines which of the possible goals are feasible in that context, and for each local goal, learns a policy to achieve it.

Each capability $C = \langle \chi_C, \mu_C, p_C, \delta_C \rangle$ also has an associated robustness, δ_C. This robustness is estimated empirically, through the proportion of times that executing μ_C in χ_C succeeded in making p_C true,

$$\delta_C \quad = \quad 1 - \frac{\text{\# times } \mu_C \text{ in } \chi_C \Rightarrow p_C = \text{true}}{\text{\# times } \mu_C \text{ in } \chi_C}. \qquad (4)$$

A low robustness ($\delta_C \to 1$) indicates that the policy cannot reliably achieve p_C. This suggests there could be a latent factor which has not been considered in χ_C, and so a more refined model of this situation is required. The execution of a policy μ, once initiated in a context χ, proceeds by attempting to maintain the predicate p. It is desirable to have policies that are locally robust (Nilim and El Ghaoui 2005).

5 An Experiment

We demonstrate the use of this representation in transfer between one-shot tasks in a dynamic two-dimensional navigation domain. Random worlds are generated, with the agent placed at a random initial position, and tasked with reaching a random goal location. This must be done while avoiding an arbitrary number of dynamic obstacles with unknown dynamics. The aim is to illustrate that this method provides an agent with the ability to survive a multitude of widely varying worlds with dynamic components. These are one-shot tasks, in that the agent has only a single attempt at each task, and a new world is generated for each instance.

Worlds are generated as polygons of between four and twelve sides, with no restrictions on convexity. They thus vary considerably in size, and shape. A random number of dynamic obstacles (between 10 and 15) are added to the world. These obstacles move arbitrarily (in our simulation, execute a random walk) over the world, at a speed of 0.6 of the agent's speed. Example worlds are shown in Figure 4.

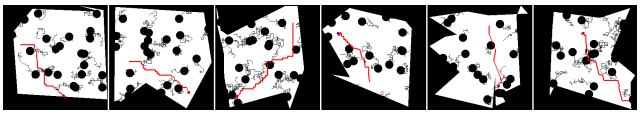

Figure 4: Six sample worlds, with dynamic obstacles and their motion traces, the agent trajectory and a goal location.

In training, the agent moved randomly around 100 worlds, for 300 time steps each, while building a GNG network of contexts in the domain. The agent was equipped with 32 ring-arranged range-sensors, which provide the state information. The resulting GNG network contained 593 nodes, which were reduced to a set of 9 landmarks. Each landmark had up to four policies trained on it: one for moving in each direction, if that was possible from that landmark. These were provided as four predicates which the agent should try and learn to achieve in each context.

A difficulty with evaluation concerns performance optimality on these tasks. To the best of our knowledge, there are no competing methods which would allow successful performance in all these worlds, online. In the absence of comparative experiments, one might look at optimality criteria. However, without knowledge of all dynamic obstacle trajectories (only available post hoc), or a corresponding generative model, optimality isn't well defined. Comparing an online algorithm to an optimal trajectory based on retrospective knowledge is not helpful. Instead, these solutions are compared to those of an optimal A* shortest path, run on stationary snapshots of the world, inspired by the concept of regret in online learning.

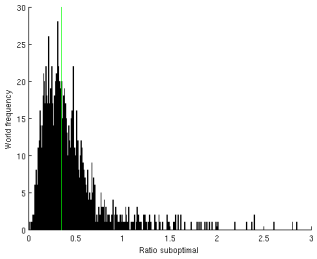

Figure 5: Results of reaching the target in 1549 different *dynamic* worlds. The histogram shows the error in the agent's path length, compared to that of the optimal solution in the stationary task where the obstacles were kept static. The median value is also shown.

Results in Figure 5 show the difference in path lengths of our solution and the optimal solution, as a fraction of the optimal solution, for a total of 1549 different worlds. Bearing in mind that the agent was acting online, in a dynamic environment performing one-shot tasks, these results show that in the majority of instances the solution path was not considerably longer than the optimal solution in the *stationary* world: in fact, generally less than 50% longer. There is a long tail to this distribution, indicating there were cases with which the online agent struggled. These typically amount to situations where the agent was required to navigate tight passageways (static or dynamic) to reach the goal. In these corner cases, improvements could be made by using the above trajectories to jumpstart solutions for an agent which is operating longer in these worlds, as a seed trajectory to be further optimised by other techniques. However, these trajectories still bootstrap performance for the one-shot tasks.

6 Planning Globally

This capability model endows an agent with the ability to act robustly in some local capacity. Using these capabilities, the agent was shown to be able to act myopically by online selection of applicable policies, using a guiding heuristic. This is however insufficient for long-term planning, and achieving goals requiring a sequence of subgoals. What is required in this case is a structuring of multiple behaviours, in order for some to move the agent into states from which other behaviours can be instantiated, bringing the agent towards the desired goal. Capabilities are then compositional elements, in terms of which sequenced and parallel capabilities can be defined, much as in the case of, say, options.

For an agent to be able to achieve a goal state g, it must appear as the output of a capability. If g is not immediately attainable in the current context, the agent leverages the fact that predictions are made in the form of p, with performance guarantees. Each capability guides the agent from one set of states χ to another set p with some probability $1 - \delta$. This probability bounds the performance of each capability. Predictable results with performance bounds enable capabilities to be used as controllers which direct the agent towards desired regions of state space in the spirit of funnels (Burridge, Rizzi, and Koditschek 1999) or skill chaining (Konidaris and Barto 2009). Thus long-term plans are maps between abstract capabilities, with no notion of the time required for the transitions. The abstract capabilities organise policy space as a form of knowledge representation.

To achieve a long-term goal, the agent reasons iteratively backwards from the goal through causal chains to determine courses of action with the required consequences. However, in any dynamic environment the world will change, and so planning should be done in terms of *weak* preconditions, which needs abstraction. It is also possible that a plan could fail, if an unpredicted event transpires during plan execution. This would push the agent's state into a different, unexpected context. To mitigate these effects, it makes sense for the agent to plan while keeping its options open. We thus require a data structure for planning with these requirements.

One proposal for a general purpose structuring of these capabilities is to organise them as a simplicial complex, which is a higher-order generalisation of a graph. In the semantics of this structure, each capability is represented by a single vertex. An edge appears between two vertices C_a and C_b if the predicate $p_a \wedge p_b$ can be held in the region defined by $\chi_a \cap \chi_b$. This means that the two capabilities can be achieved simultaneously. Similarly, faces can be defined between sets of three vertices, tetrahedra between four, etc.

These higher-order substructures within the capability simplicial complex can be learnt through exploration, incrementally, as the agent discovers that several of these predicates may be held simultaneously. These represent local tasks which the agent can perform in parallel, which provide faster trajectories towards global goals.

In the paradigm of the capability simplicial complex, a global plan becomes a mapping between different substructures on the complex. This provides the agent with a formalism for encoding the probabilistic transitions between capabilities, as these transitions are labelled by their likelihood, which is learnt from experience. This knowledge is also transferrable between instances of the same domain, or tasks in those instances. In current work, we are trying to show that these transitions speed up the planning process, by identifying capability transitions that may or may not be particularly reliable. If the agent chooses policies to always keep it on the substructure of highest dimensionality possible, then it is essentially keeping options open for responding to changes in the world, by making a least commitment to specific states, despite limited knowledge about the exact stochastic process of future evolution of the world.

For purposes of planning globally, this capability simplicial complex provides a concise alphabet of domain-specific subproblems, with simultaneous capabilities as well as temporal mappings on the same structure. This provides the agent with a topological encoding of the entire domain in terms of capabilities, and the knowledge required to address the combination of local uncertainty and a global objective.

7 Related Work

The capability model is closely related to hierarchical reinforcement learning with options (Precup, Sutton, and Singh 1998), being temporally extended actions defined as $O = \langle I, \mu, \beta \rangle$, which are tuples of initial states, policies and distribution over termination states respectively. They allow for subcomponents of solution trajectories to be learned independently and reused. The question of learning options remains in general an open question, but has been approached from many different angles (Şimşek, Wolfe, and Barto 2005; Pickett and Barto 2002). Extensions to the options model have improved generalisability, such as using homomorphisms to define mappings between relativised options (Ravindran and Barto 2003). The relation between this and our capabilities model is that options have no notion of robustness with respect to a set of contexts.

Other related approaches to problem decomposition include using libraries of behaviours matched with nearest neighbours (Stolle and Atkeson 2007). They differ from the capability model in that capabilities are grounded in perceptual contexts and are coupled to sets of robust behaviours. More symbolic approaches to task decomposition include

a complete abstraction of actions in terms of their preconditions and postconditions (Pasula, Zettlemoyer, and Kaelbling 2007), which allows for planning with the effects of these actions, efficiently as a dynamic Bayesian network (Toussaint 2009). The distinction is that rather than abstracting using a logical state about which one can reason using traditional inference methods (logical or probabilistic), we chain abstract sub-problems as a composition technique.

The capability model is also related to collections of partial models (Talvitie 2010), which have partial models making restricted predictions in restricted settings. Our model extends this by coupling these models with policies, local goals and a notion of robustness.

8 Conclusion

We present a representation for re-describing worlds generated in a domain, to associate abstract capabilities with local contexts that are useful across many task instances. This equips the agent to deal with changing worlds and tasks, enabling flexible online decision making through coarsely optimal modular policies, with local scope and reduced complexity. This capability model allows for decomposing the entire domain, rather than a single task. This provides a representation useful for multitask scenarios, and transferring knowledge learnt in one task to others in the same domain.

9 Acknowledgements

This work has taken place in the Robust Autonomy and Decisions group within the Institute of Perception, Action and Behaviour, School of Informatics. Research of the RAD Group is supported in part by grants from the UK Engineering and Physical Sciences Research Council (grant number EP/H012338/1) and the European Commission (TOMSY Grant Agreement 270436, under FP7-ICT-2009.2.1 Call 6). The authors gratefully acknowledge the anonymous reviewers for their insightful comments and suggestions.

References

Amarel, S. 1968. On representations of problems of reasoning about actions. *Machine Intelligence* 3:131–171.

Bookstaber, R., and Langsam, J. 1985. On the optimality of coarse behavior rules. *Journal of Theoretical Biology* 116(2):161–193.

Burridge, R. R.; Rizzi, A. A.; and Koditschek, D. E. 1999. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research* 18(6):534–555.

Choi, S. P. M.; Yeung, D.-Y.; and Zhang, N. L. 1999. Hidden-mode markov decision processes. *International Joint Conference on Artificial Intelligence, Workshop on Neural Symbolic, and Reinforcement Methods for Sequence Learning* 9–14.

Şimşek, Ö.; Wolfe, A. P.; and Barto, A. G. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. *International Conference on Machine Learning* 817–824.

da Silva, B.; Basso, E.; Bazzan, A.; and Engel, P. 2006. Dealing with non-stationary environments using context detection. *International Conference on Machine Learning* 217–224.

Foster, D. P., and Vohra, R. 1999. Regret in the on-line decision problem. *Games and Economic Behavior* 29:7–35.

Fritzke, B. 1995. A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems* 625–632.

Georgeot, B., and Giraud, O. 2011. The game of go as a complex network. *arXiv:1105.2470*.

Gmytrasiewicz, P. J., and Doshi, P. 2005. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research* 24:49–79.

Jaulmes, R.; Pineau, J.; and Precup, D. 2005. Learning in non-stationary partially observable markov decision processes. *ECML Workshop on Reinforcement Learning in Non-Stationary Environments*.

Konidaris, G. D., and Barto, A. G. 2009. Skill discovery in continuous reinforcement learning domains using skill chaining. *Advances in Neural Information Processing Systems* 22:1015–1023.

Korf, R. E. 1980. Toward a model of representation changes. *Artificial Intelligence* 14(1):41–78.

Lane, T., and Smart, W. D. 2005. Why (po)mdps lose for spatial tasks and what to do about it. *International Conference on Machine Learning*.

Liu, Y.-Y.; Slotine, J.-J.; and Barabási, A.-L. 2011. Controllability of complex networks. *Nature* 473:167–173.

Nilim, A., and El Ghaoui, L. 2005. Robust control of markov decision processes with uncertain transition matrices. *Operations Research* 53(5):780–798.

Pasula, H. M.; Zettlemoyer, L. S.; and Kaelbling, L. P. 2007. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research* 29(1):309–352.

Pickett, M., and Barto, A. G. 2002. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. *International Conference on Machine Learning* 506–513.

Pierce, D., and Kuipers, B. J. 1997. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence* 92:169–227.

Precup, D.; Sutton, R. S.; and Singh, S. 1998. Theoretical results on reinforcement learning with temporally abstract options. *European Conference on Machine Learning*.

Ravindran, B., and Barto, A. G. 2003. Relativized options: Choosing the right transformation. *Proceedings of the Twentieth International Conference on Machine Learning*.

Rosman, B. S., and Ramamoorthy, S. 2011. Learning spatial relationships between objects. *International Journal of Robotics Research* 30(11):1328–1342.

Roy, N.; Gordon, G.; and Thrun, S. 2005. Finding approximate pomdp solutions through belief compression. *Journal of Artificial Intelligence Research* 23:1–40.

Stolle, M., and Atkeson, C. G. 2007. Knowledge transfer using local features. *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning* 26–31.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. The MIT Press.

Talvitie, E. N. 2010. *Simple Partial Models for Complex Dynamical Systems*. Ph.D. Dissertation, The University of Michigan.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10:1633–1685.

Toussaint, M. 2009. Probabilistic inference as a model of planned behavior. *German Artificial Intelligence Journal* 3.

Yu, J. Y., and Mannor, S. 2009. Arbitrarily modulated markov decision processes. *IEEE Conference on Decision and Control* 2946–2216.