Context-based Online Policy Instantiation for Multiple Tasks and Changing Environments

Benjamin Rosman
Mobile Intelligent Autonomous Systems
Modelling and Digital Science
Council for Scientific and Industrial Research
South Africa

***Abstract*—This paper addresses the problem of online decision making in continually changing and complex environments, 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 a task can be generated from a finite set of qualitatively meaningful contexts. We present an approach to online decision making that exploits this decomposability in a two part procedure. In a task independent exploratory process, our algorithm running on an autonomous agent learns the set of structural landmark contexts which compose its domain, and reduces this set through the use of the symmetry structure of permutation groups. To each reduced landmark we then associate a set of policies independent of global context. This enables an efficient online policy instantiation process that composes from already learnt policy templates. This is illustrated on a spatial navigation domain where the learning agent is shown to be able to play a pursuit-evasion game in random environments with unknown dynamic obstacles.**

I. Introduction

This paper is motivated by the problem faced by a long-lived agent that finds itself in a continuously changing environment within which there is the need to make a sequence of non-trivial decisions. These decisions must be made despite incompleteness in available information and limitations on computational resources. This general problem pervades a variety of different domains including robotics and autonomous agents in electronic markets. In this paper, we focus attention on one specific domain – spatial navigation in random environments.

Decision making in rich, dynamic environments is difficult to address using traditional methods such as those based on optimisation of expected reward in a Markov Decision Process (MDP) which operate on *one* particular environment (at one time), specified explicitly or implicitly by a particular combination of transition dynamics, state-action representation and reward process. The primary difficulty lies in the fact that the decision maker has both local and global constraints which must be satisfied.

This is compounded by non-stationarity outside of the control of the decision maker. Some interesting approaches to this problem include robust dynamic programming with respect to a bounded family of MDPs [1], viewing the non-stationarity as arising from mode switches within a set of MDPs [2] and modifying the value iteration process to accommodate online estimates of transition/reward statistics [3], [4]. The more practicable of these results view the problem as one of repairing a base policy with respect to changes, which may be quite inadequate in the sort of domains we are interested in studying. This notion of adaptation is also related to transfer learning [5]. In this setting, there have been successful instances of reusing learnt strategies for tasks in closely related environments, but the larger issue of adapting and constructively developing representations is largely open.

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 [6], [7]. 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 a *family* of tasks [8]. An example within the robotics literature of a multi-task encoding that addresses some of these issues is seen in the work of Burridge, et al. [9], based on the notion of sequential composition by separating local policies from a global strategy. Note however that this was a carefully hand-crafted engineering design.

In this paper, we discuss achieving a similar result by a method based on learning from direct experience. Our premise is that although global aspects of the decision problem are subject to continual change, most interesting instances of complex environments are generated from a small number of *qualitatively* meaningful local contexts. Task-independently, each context may be associated with a set of local policy templates. The problem of dealing with the changing environment can then be posed in terms of a combination of offline policy learning, online decomposition and instantiation. The problem of instantiation is posed in a factored manner, separating a global task specification in the form of a game against an abstract adversary from more local aspects of non-stationarity.

We expand on ideas introduced in our earlier work [10] and present an algorithm that consists of two parts. In an offline and task-independent setting, the agent is able to explore the environment for a period of time, summarising the results in a manifold representation of possible contexts. These contexts are further reduced by eliminating symmetries through the use of group operators. With the resulting abstract contexts, one may associate a set of task-independent policy templates, to form *capabilities*. In the online phase, the agent is able to

utilise these capabilities to quickly and efficiently compose a best response to the current context. This response has a local component, akin to a greedy best reaction, and a global component, corresponding to a higher level game against an abstract adversary. We demonstrate this in a spatial navigation domain, involving random environments and agents whose dynamics are *a priori* unknown to the learning agent, and show that these results extend to adversarial settings.

II. Capability Learning

Assume a long-lived agent in some domain D is repeatedly required to perform tasks T_i in some environments, being instantiations of the domain $D_i \in \mathcal{D}$, $i = 1, 2, \ldots$. The agent assumes no prior knowledge of the structure of $\mathcal{D}$. Note that D_i represents the environment dynamics, and T_i the objectives of the agent. Both D_i and T_i may be drawn from non-stationary distributions, and in general are independent. Under this setting, in the naïve case, an agent is required to learn a completely new solution π_i for each environment-task pair (D_i, T_i).

Most interesting domains are not completely arbitrary, and instead admit latent structure, in that every instantiation of the domain is constructed from a finite set of generators, or prototypical contexts $\mathcal{X}_\mathcal{D} = \{\chi\}_\mathcal{D}$ encountered in that domain. This property is evident in the structure of rooms in buildings, and common patterns observed in games such as Go [11], [12] and chess [13]. These local contexts are composed by stitching them together sequentially in small neighbourhoods, having transformed them using a finite set of operators $\{\circ\}_\mathcal{D}$ which describe symmetries, e.g. in spatial problems; rotation, mirroring and permutations. The goal of the learning agent is then to discover these elements which allow for a decomposition of any environment in the domain. The representation $(\{\chi\}_\mathcal{D}, \{\circ\}_\mathcal{D})$ provides the agent with an alphabet from which environments can be constructed. Furthermore, if for each context χ, the agent can associate a set of control policies $\{\pi\}_\chi$ which enable it to perform any number of subtasks in that context (e.g. through local value functions), then the agent could be said to have an *understanding* of that context, through a set of capabilities which are applicable in χ. If these capabilities are learnt in a task-independent manner, then the agent is equipped with a behavioural structure, which can be transferred between tasks and environments in the domain $\mathcal{D}$.

Our model thus equips the agent with a notion of context-specific capabilities in a domain, as a representation of a family of tasks and environments. These capabilities provide a description language for the domain which can be considered as decision making affordances [14], owing to the fact that they are learnt in an unsupervised, task-independent manner. Reformulating a domain in this way allows for fast and efficient behaviour in the domain, on which a whole suite of decision-making algorithms can be built. The complete algorithm for extracting this structure from the environment is outlined in Figure 1, and described in the remainder of this section.

A. Context Manifold Learning

The first phase of building capabilities is learning perceptual contexts. At time t, an agent receives observations as a state vector s_t from a state space $\mathcal{S}$. Any inherent exploitable structure in the domain must be observable by the agent through s_t. Learning contexts requires that the agent learn the decomposition of environments generated under the domain. To do so, the agent builds a *cognitive map* of its experiences as it explores randomly generated instances of the domain.

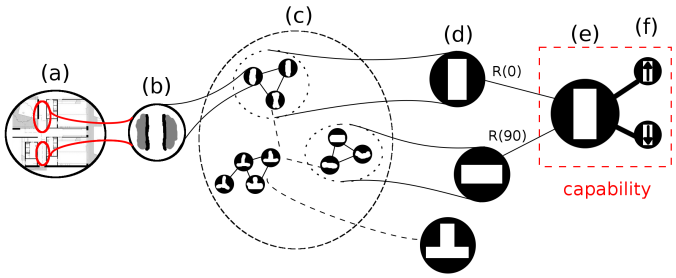

Fig. 1. The representation building process. (a) A snippet from the environment containing the agent, where two different instances of North-South corridors have been identified (b) The state vector (sensor readings) acquired by the agent by random exploratory moves through the environment (c) The growing neural gas representation of the contexts encountered by the agent (Section II-A) (d) The landmark contexts extracted from the network (Section II-B) (e) Symmetry reduced landmark set, with annotated operators (Section II-C) (f) Policies learnt offline which are applicable in that context (Section II-D). Note that the landmarks (e) and policies (f) define capabilities of the agent.

The experiences are gathered into a growing neural gas (GNG) network [15] – a form of self-organising map that has been shown to be effective for representation learning [16]. This is represented as a graph $G = (\mathcal{X}, E)$, $\mathcal{X} \subset \mathcal{S}$, $E \subset \mathcal{X} \times \mathcal{X}$ embedded in the state space $\mathcal{S}$, that grows to model the topology and distribution of sample points in $\mathcal{S}$. The network reflects the types of perceptions the agent may encounter in other instances of the domain. While alternate manifold representations may instead be used, the GNG models the topological structure of the observation manifold, as well as any dimensionality changes in different regions of the manifold. It is also a discrete model, which is useful for efficient skeletonisation (see Section II-B), and the distances between nodes can be described in terms of path lengths as a convenient metric. Furthermore, epochs of learning and acting using the network may be alternated, and so the network could grow continually throughout the lifetime of the agent.

It is important to note that the GNG network does not model temporal evolution of state, but instead models types of local scenarios which may be encountered. Generated environments in the domain could consist of any finite subset of these contexts, arranged in many ways. One would thus require a separate map of a particular instance of the domain to encode causality in that environment.

B. Cognitive Map Skeletonisation

The GNG network is a discrete manifold in the observation space $\mathcal{S}$ of the agent. The contexts represented by the nodes of the network can be composed to describe a large family of possible environments, on which many different objectives could be defined. We use properties of this manifold to select representative landmarks and simplify the structure of the graph, such that each node of the original graph is represented by a nearest landmark.

Given a graph $G = (V, E)$, compute a refined graph $G' = (V', E')$ as follows. Let $e = (u, v) \in E \subset V \times V$ be the edge in E of minimum Euclidean length in $\mathcal{S}$. Define a new node $w = (u+v)/2$. Then $V' = (V \cup \{w\}) \setminus \{u \cup v\}$. Let $\sigma_a(E, b)$ be a relabelling of all nodes a to b in E, then $E' = \sigma_{u,v}(E \setminus e, w)$.

Iteratively refine the constructed GNG network $G = G^0$, giving a sequence of graph refinements $G^0, G^1, G^2, \ldots$. For each refinement G^k, compute the Hausdorff distance

$$d_H(G, G^k) = \max\{\sup_{\chi} \inf_{\chi^k} d(\chi, \chi^k), \sup_{\chi^k} \inf_{\chi} d(\chi, \chi^k)\} \quad (1)$$

for $\chi \in \mathcal{X}$, $\chi^k \in \mathcal{X}^k$, and $d(\cdot, \cdot)$ the Euclidean distance metric considered between nodes of the two graphs. Now define $H_G(G^k) = d_H(G, G^k)$. We choose the skeletonised GNG network $G^K = G^k$, where G^k is the refined network maximising the curvature of $H_G(G^k)$.

Denote the landmark set corresponding to the nodes in this refined graph G^K by $\hat{\mathcal{X}}$. These context landmarks $\hat{\mathcal{X}}$ are points-of-interest on the context manifold, to which the agent can associate behavioural knowledge to be generalised to the rest of the manifold.

C. Symmetry Reduction

The GNG network with identified landmarks provide some notion of invariance to noise and some metrical variations to the contexts. However, there are additional topological invariances which may be preserved in the domain, but not reflected in the landmark network.

Symmetries in the observation space of an agent indicate equivalent states, for which completely new observation and behavioural models need not be learnt if the transformations are known. An effective way of modelling these is with groups. For the spatial navigation domain, we focus on permutation groups. This is more general than rotations, in that we allow mirroring, partial rotation, etc.

For an N-element state vector s, denote a permutation as an index-reordering bijective mapping $\circ : \{1, \ldots, N\} \to \{1, \ldots, N\}$, which imposes a finite group structure on the elements of s [17]. The group of permutations generated by repeated application of this operator must contain the identity permutation. This structure has two particularly advantageous properties: closure, meaning the transformations are finite and computable, and invertibility, meaning a policy associated with a landmark can be mapped back to the original space.

Note that the question of operator identification is outside the scope of this paper. The operators are however a property of the domain, and as a result, could be given to the agent as prior domain knowledge by an external source. Alternatively, they could be inferred from data [18], [19], [20].

These operators define equivalence classes over the elements of the state vector, i.e. every permutation of a single element belongs to the same equivalence class. These classes identify symmetric structure in the data. For each operator $\circ$, its order $|\circ|$, is defined as the number of times $\circ$ must be applied to s such that the result is again s. The function of these operators is to reduce the set of landmark contexts based on symmetries. The question we wish to answer is: given two landmarks χ_1 and χ_2, is there a finite sequence of operators $\circ^* = \circ_1 \circ_2 \cdots \circ_P$ such that $\chi_1 \simeq \circ^* \chi_2$?

These permutation groups are finitely generated, and as such are finite-automaton presentable [21], meaning the group operation can be recognised by a finite automaton (FA). We can thus construct a FA from the definition of the domain operators to recognise permutations.

Construct the FA F as follows: For each index i of the N-element state vector, create a state in F labelled i. Now, for each operator $\circ_j$ of order M_j, consider the enumeration of the M_j repeated group permutations from that operator. The composition of all J operators thus describes an enumeration of up to $M = M_1 \times M_2 \times \cdots \times M_J$ permutations. For each of these permutations $P = [p_1, p_2, \cdots, p_N]$, create a directed transition (i, j) for each consecutive pair (p_i, p_j) in P. Label this transition with the sequence of operators used in P.

Figure 2 provides an example of how an automaton may be constructed to identify a composition of operators in a toy domain.

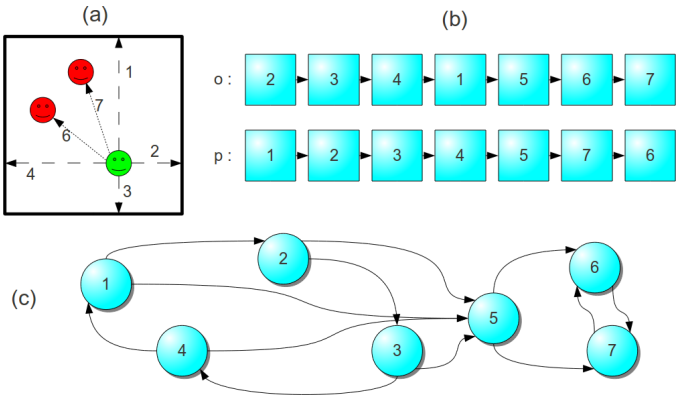

Fig. 2. An example of permutation groups. (a) An agent's state vector has 7 elements: 1-4 are distances to the walls of the room, 5 is speed, 6-7 are distances to targets (b) Two operators define rotation in elements 1-4 (o) and a permutation in 6-7 (p) (c) The automaton created from the permutation set to recognise elements of the group.

F is constructed once for the set of operators. It is then instantiated for a particular set of values by labelling the transitions of F with the values from χ_1, giving F_{χ_1}. F is then presented with χ_2, starts at the state corresponding to the first element of χ_2, and transitions to new states as each element of χ_2 is read. F accepts the vector if all transitions made are defined in F, and if all states are visited exactly once. If F_{χ_1} accepts χ_2 (or an approximation thereof, as the values may be continuous) then these two landmarks are equivalent under the operator set. Furthermore, the intersection of the annotated labels on the transitions of F_{χ_1} required to accept χ_2 provides the sequence of operators required to transform χ_2 into χ_1.

Using the set of symmetry operators and the constructed automaton F, we reduce the set of landmarks $\hat{\mathcal{X}}$ to a smaller set, to eliminate symmetry-equivalent members of $\hat{\mathcal{X}}$. Let the reduced landmark set be denoted $\mathcal{X}_< \subseteq \hat{\mathcal{X}}$.

D. Policy Association

The symmetry reduced landmark set $\mathcal{X}_<$ with the operators $\{\circ\}$ provide a decomposition of the domain into metrically

and topologically invariant contexts. The goal of learning this reduction is to facilitate policy learning and selection, which is faster in a smaller candidate state space. Policy learning happens offline, as a form of introspection over each landmark context.

For each landmark $\tilde{\chi} \in \mathcal{X}_{<}$, a set of policies $\{\pi\}_{\tilde{\chi}}$ are learnt, enumerating the set of all behaviours which the agent can accomplish in that context. The policies may be learnt through any of a number of techniques, e.g. learning by demonstration from an expert being a human or another agent [22], mixing between behaviours [23], through a form of stochastic optimal control such as reinforcement learning [24], or a heuristic approach like a shortest path A* algorithm. This equips the agent with a set of locally applicable behaviours.

To learn policies in the reduced landmark set, the agent considers each in turn, in an offline manner. In learning a policy, the landmark context is treated as a stationary, isolated and restricted environment. The required local set of behaviours may well need to be prescribed by an external source, but the trajectories may be optimally learnt. For example, in our navigation experiments policies were learnt by an A* algorithm, although a reinforcement learning algorithm could have been used instead to learn value functions.

Each landmark having an associated policy set in this way could be broadly interpreted as the agent having an 'understanding' of that context – the agent could be said to understand some small local environment if that environment could be identified and the agent is equipped with a number of different activities that could be performed locally, irrespective of global context.

As an alternative to learning different policies for each context, we may assume that the policy set is fixed across the domain. In this case, we may instead learn *action priors* over the policy set for each context, being task-independent probabilistic action selection preferences in that situation [25].

III. Online Policy Instantiation

Having learnt the GNG context manifold and associated policies, the agent is equipped for online decision making. Given a task and domain instance, at each time step t the agent acquires some information s_t about its current state in the environment. This is used to perform a lookup into the GNG network to find the context χ_t such that $\chi_t = \arg\min_{\chi \in \mathcal{X}} d_S(s_t, \chi)$, for $d_S(\cdot,\cdot)$ a distance in S. This returns the landmark context most similar to the current state. Note that if the distance between χ_t and s_t exceeds some threshold, e.g. some percentage of the total distance spanned by the GNG, this is a state with which the agent is not familiar, and so the agent may reinstate the GNG learning temporarily.

Find the best matching template $\hat{\chi}_t$ from the full landmark set $\hat{\mathcal{X}}$ to the context χ_t, such that $\hat{\chi}_t = \arg\min_{\hat{\chi} \in \hat{\mathcal{X}}} d_G(\chi_t, \hat{\chi})$, for $d_G(\cdot,\cdot)$ the graph distance measured by the number of links between two nodes in G. $\hat{\chi}_t$ is thus the landmark policy which best matches s_t.

Now, for each landmark context $\hat{\chi}_i$, there is a group transformation $\circ_i$ which maps it to some symmetry reduced landmark $\tilde{\chi}_i$. Let $(\tilde{\chi}_t, \circ_t)$ be the transformation-landmark pair associated with $\hat{\chi}_t$. $\tilde{\chi}_t$ is then a topologically and metrically invariant version of s_t. This affords the agent a set of behaviours $\{\pi\}_{\tilde{\chi}_t}$ which could be instantiated at time t. If the chosen policy π_t describes a behaviour in state space relative to $\tilde{\chi}_t$, then $\circ_i^{-1}\pi_t$ describes the behaviour relative to s_t.

The aforementioned procedure, depicted in Figure 1, describes a generalisable representation scheme of the environments which could be generated in some domain. As such, this is not a causal model of the domain. To predict the results of selecting a particular policy, one must learn a causal map (specified in the language of landmark contexts and associated policies) of a particular environment in the domain. Inference can then be done over this causal model. For example, it could be learnt that in a *particular instance* of the domain, executing some policy from a specific template leads to other templates with some probability. This causal understanding of that environment gives rise autonomously to dynamics models [26], for which plans can be computed using methods such as dynamic Bayesian networks [27].

This representation and policy instantiation method allows for a coupling of local reactions to uncertainty, with a higher level global game associated with incomplete knowledge of a different kind, e.g. strategies of an adversary. This is illustrated in our second experiment in Figure 5. Such a factored approach makes it possible to separate concerns and acknowledge incompleteness of knowledge at a global level.

IV. Experimental Results

We illustrate our method in a two-dimensional navigation domain. Random environments are generated, in which the agent is placed at a random initial position, and is tasked with reaching a random goal location. This must be done while avoiding an arbitrary number of dynamic obstacles whose dynamics are unknown. The aim is to illustrate that our algorithm provides an agent with the ability to survive a multitude of widely varying environments with many dynamic components.

Environments are generated as polygons of between four and twelve sides, and thus vary considerably in size, shape and convexity. A random number of dynamic obstacles (between 10 and 15) are added to the environment. These obstacles execute a random walk, at a speed of 0.6 of the agent's speed. Example environments are shown in Figure 3.

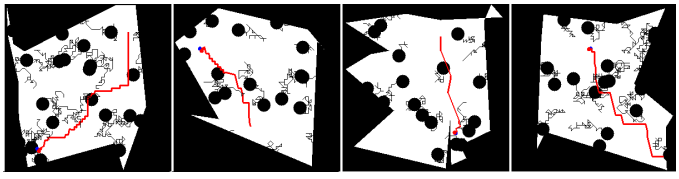

Fig. 3. Four sample environments, showing dynamic obstacles with their traces over time, the agent trajectory (in red) and a goal location.

In training, the agent moved randomly about 100 environments, for 300 time steps each, while building a GNG network of contexts in the domain. State information was constructed from a set of 32 ring-arranged range-sensors. The resulting network contained 593 nodes, which were reduced to a set of 9 landmarks. Each landmark had a maximum of four policies

trained on it: one for moving in each direction if possible in that landmark.

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 environments, 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*), optimality isn't well defined. Comparing an online algorithm to an optimal trajectory based on retrospective knowledge is meaningless. Instead, we compare our solutions to those of an optimal A* shortest path, run on stationary snapshots of the environment, inspired by the concept of regret in online learning.

Results are reported in Figure 4 showing the difference in the path lengths of our solution and the optimal solution as a fraction of the optimal solution, for a total of 1549 different environments. 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* environment: in fact, generally less than 50% longer. There is a long tail to this distribution, which indicates that there were cases with which the online algorithm struggled. These typically amount to situations where the agent is required to navigate tight passageways (static or dynamic) to reach the goal. In these corner cases, improvements could be made by using these trajectories as jumpstart solutions for an agent which is operating longer in any of these environments, as a seed which could be further optimised by other techniques.

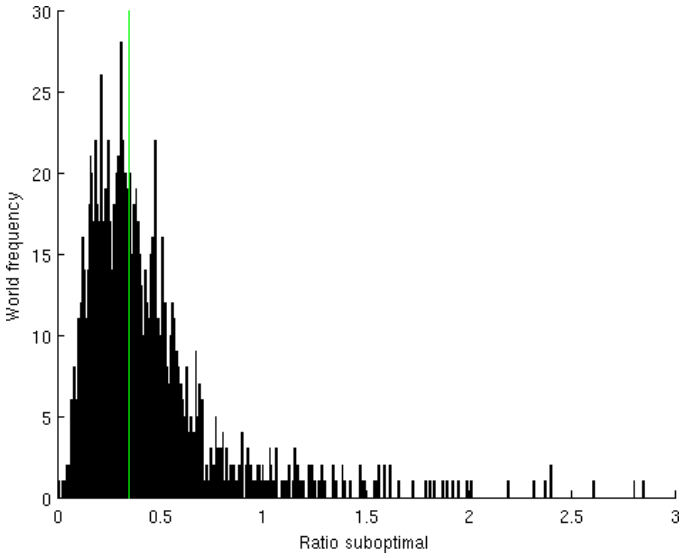

Fig. 4. Results of reaching the target in 1549 different *dynamic* environments. The histogram shows the error in our agent's path length, compared to that of the optimal solution in the *stationary* task where the obstacles were kept static. The median value is shown in green.

Once we have this ability to locally respond to environmental change, we may couple this with different global task requirements. A second experiment was aimed at demonstrating this capability of the algorithm and representation. We use the previously learnt capabilities to play a pursuit-evasion game known as the "homicidal chauffeur" [28], [29] which involves a slower but more manoeuvrable evader fleeing a slightly faster and less agile pursuer. The relative position of the evader (x, y) to the pursuer is

$$\begin{aligned} \dot{x} &= V_e \sin\phi - V_p \frac{y}{R} u, \\ \dot{y} &= V_e \cos\phi - V_p + V_p \frac{x}{R} u, \\ u &\in [-1, 1], \end{aligned}$$

for V_e the constant speed of the evader, V_p the constant speed of the pursuer, R the minimum pursuer turn radius, ϕ the evader's control and u the pursuer's control. Results are shown in Figure 5, and indicate that our algorithm is still able to handle non-stationary environments while performing a closed-loop task which depends on the behaviour of the agent itself.

Note that the agent did not learn the solution to the game. Instead, the focus here is to demonstrate that policy instantiation is not restricted to myopic decisions but can be part of longer-term reasoning. In this example, it would be infeasible to model the entire environment to the extent demanded by traditional approaches to solving the differential game. Our approach reduces the problem to a more manageable one.

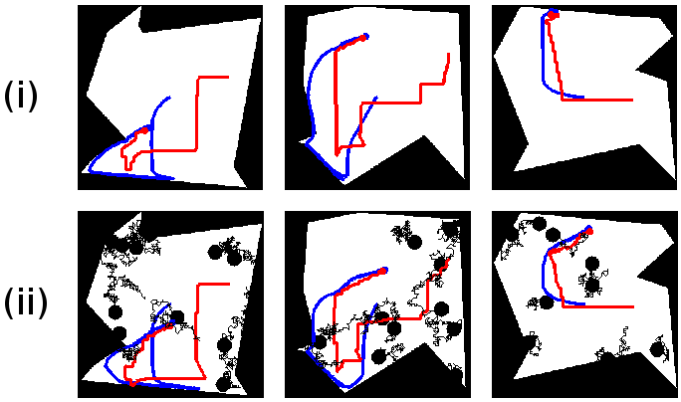

Fig. 5. Result trajectories for three instances of the pursuit-evasion games. The pursuer trajectory is shown in red, and the evader in blue. Row (i) is the solution in a stationary environment and row (ii) is the same problem with added dynamic obstacles.

This is just one example of a global game. The point is that non-stationarity and change need not only be approached via expensive repair of a single comprehensive policy, instead it can be dealt with compositionally by formulating the decision problem in an essentially different way; designed for a changing environment from the outset.

V. Related Work

Our method learns particular sub-policies which are applicable in various local contexts. As such, they are closely related to the options framework of hierarchical reinforcement learning. The identification and discovery of options is still an open question which has received considerable attention [30], [31], [32]. These methods focus on finding common subsets or generalisations of solutions to particular environments in a domain. Our method instead focuses on commonly occurring features in the environment, over a large number of different environments, and applying these to approximate

context matches. The local policies we learn can also be used as options, but the association of contexts allows us to further factorise the domain. Other approaches to transforming policies are relativised options [33] and SMDP homomorphisms [34]. We adopt a related notion of abstraction through our group formalism, but our complete formulation allows for local models to be used dynamically within any online decision procedure.

Our approach is also inspired by the idea of analysing the space of value functions to find repeated structure [35], but we couple observations and behaviours. Our method is most similar to the notion of learning local features for behaviour reuse [36], [37] but we explicitly model the context manifold using a GNG network, and reduce this with group operators.

VI. Conclusion

We present an algorithm for re-describing environments generated in a domain to associate generic capabilities with local contexts that are qualitatively meaningful across many task instances. This equips the agent to deal with changing environments and tasks, enabling flexible online decision making. This factors the problem by separating the global issues having to do with the larger scale structure of the task from the local issues that can be dealt with by correspondingly local policies that are essentially independent of other higher level considerations. So, for instance, the agent could treat the global problem as a low-dimensional differential game while the local problem requires a higher dimensional policy, but one that can be learnt offline and tuned across multiple environments.

References

[1] J. Y. Yu and S. Mannor, "Arbitrarily Modulated Markov Decision Processes," *IEEE Conference on Decision and Control*, pp. 2946–2216, December 2009.

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

[3] R. Jaulmes, J. Pineau, and D. Precup, "Learning in Non-Stationary Partially Observable Markov Decision Processes," *ECML Workshop on Reinforcement Learning in Non-Stationary Environments*, 2005.

[4] B. da Silva, E. Basso, A. Bazzan, and P. Engel, "Dealing with Non-Stationary Environments using Context Detection," *International Conference on Machine Learning*, pp. 217–224, 2006.

[5] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains: A Survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[6] S. Amarel, "On representations of problems of reasoning about actions," *Machine Intelligence*, vol. 3, pp. 131–171, 1968.

[7] R. E. Korf, "Toward a model of representation changes," *Artificial Intelligence*, vol. 14, no. 1, pp. 41–78, August 1980.

[8] R. Bookstaber and J. Langsam, "On the optimality of coarse behavior rules," *Journal of Theoretical Biology*, vol. 116, no. 2, pp. 161–193, September 1985.

[9] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential Composition of Dynamically Dexterous Robot Behaviors," *The International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, June 1999.

[10] B. S. Rosman and S. Ramamoorthy, "A Multitask Representation using Reusable Local Policy Templates," *AAAI Spring Symposium Series on Designing Intelligent Robots: Reintegrating AI*, 2012.

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

[12] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabási, "Controllability of complex networks," *Nature*, vol. 473, pp. 167–173, May 2011.

[13] H. A. Simon and W. G. Chase, "Skill in Chess: Experiments with chess-playing tasks and computer simulation of skilled performance throw light on some human perceptual and memory processes," *American Scientist*, vol. 61, no. 4, pp. 394–403, July-August 1973.

[14] J. J. Gibson, *The Ecological Approach to Visual Perception*, 2nd ed. Lawrence Erlbaum Associates, Inc., 1986.

[15] B. Fritzke, "A Growing Neural Gas Network Learns Topologies," *Advances in Neural Information Processing Systems*, pp. 625–632, 1995.

[16] J. Provost, "Reinforcement Learning in High-Diameter, Continuous Environments," Ph.D. dissertation, The University of Texas at Austin, August 2007.

[17] C. C. Sims, *Computation with Finitely Presented Groups*, ser. Encyclopedia of mathematics and its applications. Cambridge University Press, 1994, vol. 48.

[18] X. Miao and R. P. N. Rao, "Learning the Lie Groups of Visual Invariance," *Neural Computation*, vol. 19, pp. 2665–2693, 2007.

[19] G. Bartók, C. Szepesvári, and S. Zilles, "Models of active learning in group-structured state spaces," *Information and Computation*, vol. 208, no. 4, pp. 364–384, April 2010.

[20] D. Pierce and B. J. Kuipers, "Map learning with uninterpreted sensors and effectors," *Artificial Intelligence*, vol. 92, pp. 169–227, 1997.

[21] A. Nies, "Describing Groups," *The Bulletin of Symbolic Logic*, vol. 13, no. 3, pp. 305–339, September 2007.

[22] P. Abbeel and A. Y. Ng, "Apprenticeship Learning via Inverse Reinforcement Learning," *International Conference on Machine Learning*, 2004.

[23] B. S. Rosman and S. Ramamoorthy, "A Game-Theoretic Procedure for Learning Hierarchically Structured Strategies," *IEEE International Conference on Robotics and Automation*, 2010.

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

[25] B. S. Rosman and S. Ramamoorthy, "What good are actions? Accelerating learning using learned action priors," *International Conference on Development and Learning and Epigenetic Robotics*, November 2012.

[26] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *Journal of Artificial Intelligence Research*, vol. 29, no. 1, pp. 309–352, May 2007.

[27] M. Toussaint, "Probabilistic inference as a model of planned behavior," *German Artificial Intelligence Journal*, vol. 3, 2009.

[28] T. Başar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed. Society for Industrial and Applied Mathematics, 1999.

[29] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere Publishing Corporation, 1975.

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

[31] M. Pickett and A. G. Barto, "PolicyBlocks: An Algorithm for Creating Useful Macro-Actions in Reinforcement Learning," *International Conference on Machine Learning*, pp. 506–513, 2002.

[32] G. D. Konidaris and A. G. Barto, "Skill Discovery in Continuous Reinforcement Learning Domains using Skill Chaining," *Advances in Neural Information Processing Systems*, vol. 22, pp. 1015–1023, December 2009.

[33] B. Ravindran and A. G. Barto, "Relativized Options: Choosing the Right Transformation," *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[34] ——, "SMDP homomorphisms: an algebraic approach to abstraction in semi-Markov decision processes," *Proceedings of the 18th international joint conference on Artificial intelligence*, pp. 1011–1016, 2003.

[35] D. Foster and P. Dayan, "Structure in the Space of Value Functions," *Machine Learning*, vol. 49, pp. 325–346, 2002.

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

[37] ——, "Finding and transferring policies using stored behaviors," *Autonomous Robots*, vol. 29, pp. 169–200, 2010.