Small World Network Based World Representation for Scalable Reinforcement Learning

Seung-Joon Yi SJLEE@BI.SNU.AC.KR
Byoung-Tak Zhang BTZHANG@BI.SNU.AC.KR
School of Computer Science and Engineering, Seoul National University, Seoul 151-742, South Korea

Abstract

The curse of dimensionality plagues practical uses of reinforcement learning. Temporal abstraction approaches have been proposed to overcome this problem, but typically they require a priori design of the hierarchy and lack the compact representation needed for large sized problems, so their practical uses are still limited. Inspired by recent research in complex networks, we present a compact self-organizing, growing network for world representation to scale up reinforcement learning. Continuous state space is represented with a compact self-organizing network, and the network is augmented to have small world property without a priori knowledge. Experimental results with various problem sizes show that the average path length between nodes of this network scales subpolynomially with the size of the network, and the convergence of reinforcement learning is accelerated significantly.

1. Introduction

In the framework of Reinforcement Learning (RL), an agent attempts to learn a policy, i.e. a mapping from state to action, that maximizes some time aggregate of rewards. In the traditional RL framework, the environment is defined as a discrete-time, discrete-state Markov decision process (MDP). Popular RL algorithms such as Q-Learning assume tabular representation of both the state and the action space, and estimates the values of all the state-action pairs to find the optimal policy (Watkins & Dayan, 1992). However, in most real world problems with continuous or high-dimensional state spaces, it is impossible to enumerate all of the state-action pairs. Even with the problems of discrete state space, it is impractical to estimate the value function for all the states when the problem size is large. So it is necessary to use some kind of compact world representation schemes.

A common solution to large or continuous state spaces is using a function approximator such as neural networks (Boyan & Moore, 1995). Using a function approximator with RL has shown good results in some situations. However, it is also known that the number of parameters to be estimated grows exponentially with the size of any compact encoding of a state (Barto & Mahadevan, 2003). Attempts to combat this curse of dimensionality lead to temporal abstraction where decisions are not required to perform every single action. This naturally leads to hierarchical control architectures and thus the associated learning algorithms are called hierarchical reinforcement learning (HRL) algorithms. However it is still hard to use these approaches directly to real world tasks. The first problem is that the structure of hierarchy, subgoals, sub MDPs and subtasks should be decided in advance. The user should 'program' with the problem specific knowledge. Another problem is that these hierarchical RL approaches still assume a tabular representation of states and actions which makes it difficult to apply the HRL algorithms directly to large sized problems that really need them.

Meanwhile, recent studies of complex networks show that many real-world networks, such as web graphs and social networks, show the small world property where most pairs of nodes are linked by short chains of nodes (Watts & Strogatz, 1998). Based on this fact, we propose a new approach of building temporal abstraction using the small world network models. The MDP is augmented with subtasks whose structure is determined by the corresponding small world network models, where two states in the augmented MDP need small number of decision steps between them.

Appearing in *Proceedings of the ICML'05 Workshop on Rich Representations for Reinforcement Learning*, Bonn, Germany, 2005.

And for the practical application to the continuous state real world problems, we also utilize an incremental network that adaptively maps sensory input to actions. By augmenting the network to have the small world property and using appropriate navigation algorithms to select actions, we can build a scalable RL algorithm with compact representation without problem specific knowledge.

This paper is organized as follows. In Section 2, we briefly review the RL framework and the properties of complex networks. In Section 3, we present a practical reinforcement learning algorithm with small world network representation of the environment. In Section 4, we describe the experimental setup and the results. Finally, in Section 5, we conclude with a few future directions.

2. Related Works

2.1. Reinforcement Learning and Information Propagation on Networks

Here we briefly review the standard reinforcement framework of discrete time, finite MDPs. In this framework, a learning agent interacts with an environment at discrete time scale. On each time t, the agent chooses an action $a_t \in A$ using its policy based on the state $s_t \in S$ it perceives. Then the environment gives the agent a numerical reward $r_{t+1} \in R$ and moves to the next state s_{t+1}. The objective of the agent is to learn a policy, mapping from states to actions, that maximizes the expected discounted future reward defined as

$$R_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \quad (1)$$

To learn the optimal policy, we can get the optimal action value function $Q^*(s,a)$ using the following Q-learning method (Watkins & Dayan, 1992)

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)]. \quad (2)$$

We can generalize the MDP framework to Semi-MDP framework where each action a can take variable amounts of time $k(s,a)$ (Sutton et al., 1999). A slightly different update rule can be used such as

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma^{k(s,a)} \max_{a'} Q(s',a') - Q(s,a)]. \quad (3)$$

In the update rules (2), (3), action value functions of a state s, $Q(s,a)$ are updated using the value functions of its successor state s', $Q(s',a')$. This can be viewed as the propagation of information from state s to its successor state s'. When the agent receives positive reward in a certain state, the information of that reward is propagated to adjacent states, in the form of action value function. When each state collects enough amount of information, their action value function converges and the RL problem is solved. By making the propagation of information faster by adopting temporal abstraction, we can expect better efficiency in solving RL problem.

2.2. Complex Networks

Recent researches on complex networks have showed that most of the real world networks share the following three features. (a) Small world property. There exists a short path between any two nodes, compared to their size (Watts & Strogatz, 1998). (b) High clustering coefficient. Two nodes with a common neighbor has much more likely to be connected than two nodes without one. (c) Scale free degree distribution. The distribution of the degree decays as a power law, which is invariant to scaling. This property is often related to the hierarchical organization of the network.

Various network models with these properties are suggested. Here we present two small network models we will use for our task. First model is the Kleinberg's model (Kleinberg, 2001). In this model, we start with a regular lattice network and random long links (u,w) are added to current network with probability proportional to $d^{-\alpha}$ where d is the lattice distance from u to w. It is known that if we set the value of α to the dimension of underlying lattice, a decentralized greedy algorithm can achieve polylogarthmic search time. Second model is the scale free growing network model (Barabási & Albert, 1999). In this model, when a new node u is added to current network random long links (u,w) are added with probability proportional to degree of w, which is called the linear preferential attachment. This model also shows the scale free degree distribution.

3. Building a Small World Network for World Representation

Temporal abstraction approaches reduce the number of decision steps between states by augmenting the MDP with subtasks. But they require a priori knowledge of the problem in most cases. Instead of using the problem specific knowledge, we propose to use the network model with small world property to determine the structure of hierarchy. By the small world property, the augmented MDP will have small number of decision steps between states, which will help solving

RL problem efficiently.

3.1. The Small World Network-Based World Representation

To apply RL for continuous problem, we have to approximate the state space by discretization or using function approximators such as neural networks. To adopt two small world networks we mentioned above, we need an incremental network model with a regular lattice structure. An example of such a model is the growing neural gas network(Fritzke, 1995). Similar approaches were used in(Gross et al., 1998; Toussaint, 2003) for world representation. Its online extension is the incremental topology preserving map(Millan et al., 2002) which we will use as the base of our algorithm.

Our algorithm is summarized in figure 1. It incrementally adds nodes to the unexplored regions of state space, and uses self-organization rule to modify the connectivities and positions of nodes. Links between edges are modified so that each node is only connected to its neighbors, and the position of nearest node and the positions of all its neighboring nodes are moved closer to the input position. Furthermore, long range edges are added to make the network have the small world property. If we assume that all the state transitions are local, we can use this network model as the discrete MDP which approximates the original problem. Each node represents a corresponding region of state space, and each edge correspond to a subtask of moving to a specific state in original problem. This approach has much in common with topological mapping approaches in robotics (Thrun, 1998) and node graph approaches used in many 3D games (Rabin, 2002).

Our approach has two major differences from other network based approaches. The first difference is the existence of long links. When a new node is added, a long link denoting the corresponding subtask is added with the probability given by a small world network model. This procedure makes the resulting network to have the small-world property. The second difference is the constraint of visibility, which requires that there should be a straight, non-blocked path between two linked nodes in state space. If this requirement is met, the optimal subpolicy for a subtask is trivially given as moving straightly to its subgoal.

3.2. Reinforcement Learning in a Small World Network

Now we need an appropriate RL algorithm for the small world network based world model. To select actions, the easiest way is using conventional action selection methods such as ϵ-greedy method. Although

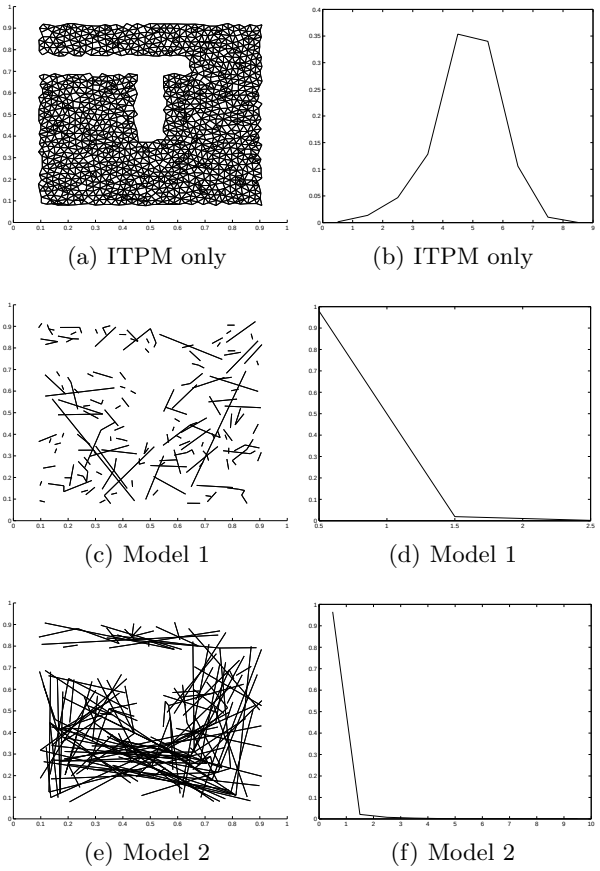
1. Perceive the current position x.
2. Find the nearest node b visible from x
   and second nearest node b' visible from x.
3. If the distance between x and b exceeds the unit radius r, then
   (A) Add a node u at x.
   (B) Create edges from u to b and b',
   (C) Remove any edge between b and b',
   (D) Select nodes v visible from x according to probability proportional to:
   (MODEL 1) $dist(u,v)^{-\alpha}$, where $dist(u,v)$ is the euclidian distance between u and v.
   (MODEL 2) $d(v)$, the number of long range edges starting from v.
   (E) Create long range edges from u to v with the probability p_l.
   Else
   (F) Create an edge between b and b' if b is visible from the position of b'.
4. Move w_b, the position of b, and w_r, the positions of all its neighboring nodes r, toward x if possible:
   $w_b \leftarrow w_b + \delta(x - w_b)$
   $w_r \leftarrow w_r + \delta_r(x' - w_r)$

Figure 1. An algorithm to build a small world network based world model.

this method ignores the network structure, augmenting MDPs with temporal abstraction alone can help reinforcement learning process (Sutton et al., 1999). And it is also known that random walking in a scale free network gravitate towards the high degree nodes, making the search more efficient (Adamic et al., 2001). So for the simplicity, we use a simple ϵ-greedy action selection rule in this work. To update the value functions, we can directly use Semi-MDP value update rule (3) for both of the models we described above. Finally, we need to get a subpolicy for each subtask. For the navigation task we are interested in, finding subpolicies can be trivial. However, for a general RL task, we have to use a local policy learning algorithm such as the Experience Replay procedure(Lin, 1992). We leave this as a future work.

4. Experimental Results

In this section we describe two experiments. For both experiments, we first let the agent explore the state space and use our algorithm to generate Semi-MDP network. And we run standard Q-learning algorithm on the network. Two versions of proposed algorithm, each using model 1 and model 2 we discussed above, are tested against standard ITPM algorithm. Common parameter values we use are as follows: movement parameter δ=0.0002, δ_r=0.00002, long link ratio



(a) ITPM only (b) ITPM only

(c) Model 1 (d) Model 1

(e) Model 2 (f) Model 2

Figure 2. Example of networks generated using each algorithms and their degree distributions. Networks are generated with r=0.02 and degree distributions are measured with r=0.00625. Only long links are shown for Models 1 and 2.

p_l=0.1. The lattice dimension p is empirically determined from the average degree of the baseline network. Various values for unit radius r are used to examine the scaling behaviors of each algorithms.

Our experiment is greatly simplified by the visibility constraint, by which the optimal subpolicy for a subtask is trivially given as going straight to their subgoal position. Although this approach is not suitable for general RL tasks where visibility constraint is not applicable, it suits well for navigation task we are interested in. For reinforcement learning part, we use simple epsilon-greedy action selection rule with ϵ=0.1, α=0.5, γ=0.9. we used the Semi-MDP update rule (3), using the length of edges normalized by the unit radius r as the execution time $k(s, a)$. We will discuss each experiments further in following subsections.

4.1. 2D Puddleworld

The first experiment uses a [0,1) by [0,1) continuous state space with a T-shaped obstacle in it. At each

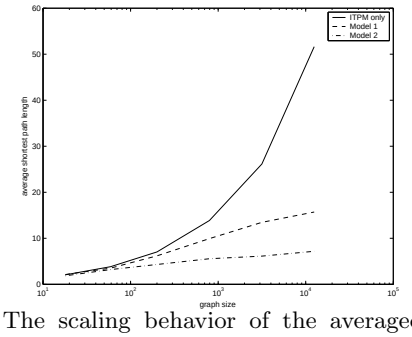
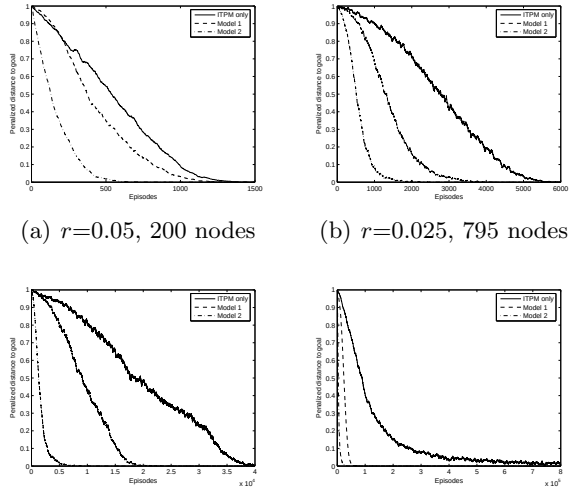


Figure 3. The scaling behavior of the averaged shortest path length. In contrast to baseline ITPM which shows a polynomial growth of shortest length, model 1 and model 2 show a polylogarithimic growth of shortest path length which is shown as the straight line on semi-log plot.

time step the agent can move to any direction, with step size $r/10$. We let the agent do random walk and generate networks for each models using unit radius r=0.2,0.1,0.05,0.025,0.02,0.0125,0.01 and 0.00625. Generated networks from each models and their degree distributions are shown in figure 2. Figure 2a shows the baseline network ITPM generates where each node is connected to only its neighbors. From figure 2b, we can see the degree distribution of the network is peaked at 5, and the lattice dimension is determined as p=$log_2 5$=2.322. Figure 2b and 2c show the additional long links model 1 and 2 add to the baseline network. From figure 2f, we can see the emergence of hub structures and the characteristic power law curve of degree distribution in model 2. The average shortest path length of each network is shown in figure 3. We can see that in contrast to the baseline network where the average shortest path length grows linearly with problem size, the average shortest path length grows polylogarithmically in model 1 and model 2.

Finally we run a Q-learning algorithm on generated networks. At each episode, the agent starts at a random start point and move up to 100 steps. In an absorbing goal area positioned upper right corner of the state space, a reward of 100 is given. The number of episodes in a run is empirically determined to fully cover its convergence phase. The total distance from start point to goal is measured at each episode. To penalize the episodes that fail to reach the goal within 100 unit distance, we assign 1000 unit distance as the penalized distance to the goal of that episode. We run a number of runs and average the penalized distances to goal. Figure 4 shows the convergence of penalized distance to goal for various network sizes. We can see that using model 1 and 2 improves the convergence speed of RL, and this effect is more apparent in bigger problem. To see the scaling behavior of RL performance over problem size, we measure the number of episodes needed to reach 50% convergence from each



(a) r=0.05, 200 nodes (b) r=0.025, 795 nodes

(c) r=0.0125, 3157 nodes (d) r=0.00625, 12501 nodes

Figure 4. The penalized distances to the goal at each episode, averaged after 100 runs. 1000 unit distance is used to penalize the failed episodes. Each graph is normalized to fit in the range of 0 to 1.

learning curves. To reduce the effect of variance, we apply a smoothing filter which averages outcomes within a given window size. Figure 5 shows the scaling behaviors of RL using each models. Though they do not scale polylogarithmically like shortest path length, we can accelerate the convergence of reinforcement learning algorithm significantly.

4.2. 3D Gameworld

To show that our algorithm can handle a complex problem with a continuous state space, we now consider a 3D gameworld task. For our experiment platform, we use a popular 3D game Half-Life 2 (Hodgson, 2004) which enables the real-time simulation of continuous 3D world. We use a map named dm_lockdown whose size is approximately 50*150*10 m. Learning agent has size 1*1*1 m and has maximum movement speed of 7 m/s (27k mph). The position of agent is checked every 1/20 second to update the network and get a new action. To explore the state space efficiently, we use a RL based multi agent exploration algorithm, which we do not cover in this paper.

We generated three networks using unit radius r=0.5,0.75,1.0 m. Generated networks consist of 4594, 6403, 13252 nodes and 6933, 11447, 13866, 23341 links, respectively. Long links generated by model 1 and 2 account for about 5% of total links. Generated network using r=0.75 meter is shown in Figure 6. With the same setting we use for the first experiment, we run Q-learning on each networks. A reward of 100 is given at the single goal located in one of the rooms. Figure 7

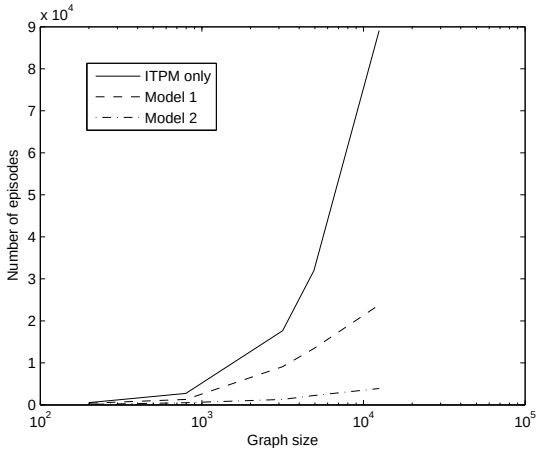


Figure 5. The scaling behavior of the number of episodes to reach 50% convergence.

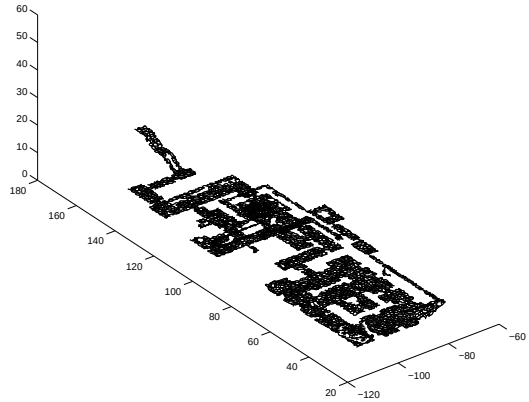
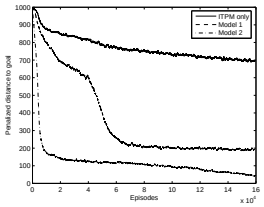


Figure 6. The 3D view of generated baseline network using r=0.75 m. Total number of nodes is 6403.

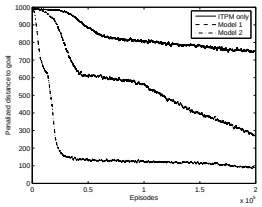
shows the convergence properties of penalized length to goal for each models. In contrast to the baseline ITPM algorithm which shows very slow convergence speed, our algorithm quickly start to converge.

5. Conclusion and Future Work

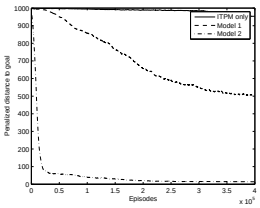
We propose a novel network based world representation to cope with the curse of dimensionality in reinforcement learning. By augmenting the network with a small number of additional links based on small world network models, we demonstrate that we can keep the growth of the number of decision steps polylogarithmically, which can accelerate the convergence of RL as the problem size grows bigger. Experimental result with 3D game environment shows that our algorithm can learn usable policy in reasonably short time, even with a huge problem with state size exceeding 10,000. Although still preliminary, our approach is promising



(a) r=1, 4594 nodes



(b) r=0.75, 6403 nodes



(c) r=0.5, 13252 nodes

Figure 7. The penalized distances to goal at each episode for a 3D gameworld task, averaged after 10 runs. The 1000 unit distance is used as the penalized distance for failed episodes.

in several aspects: it is scalable, takes continuous state and action space, has a compact representation and does not need problem specific knowledge.

There are many interesting directions for future work. The most interesting one is extending our approach to general continuous state space RL problems by adopting a subpolicy learning algorithm. Using larger subtasks can help reducing the number of decision steps between states, but it may add overhead of learning subpolicies for larger subspaces. Finding the right tradeoff between these two will be a challenging problem.

References

Adamic, L. A., Lukose, R. M., Puniyani, A. R., & Huberman, B. A. (2001). Search in power-law networks. *Phys. Rev. E*, *64*, 46135–46143.

Barabási, A.-L., & Albert, R. (1999). Emergence of scaling in random networks. *Science*, *286*, 509–512.

Barto, A. G., & Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Systems journal*, *13*, 41–77.

Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. *Advances in Neural Information Processing Systems 7* (pp. 369–376). Cambridge, MA: The MIT Press.

Fritzke, B. (1995). A growing neural gas network learns topologies. In G. Tesauro, D. S. Touretzky and T. K. Leen (Eds.), *Advances in neural information processing systems 7*. Cambridge MA: MIT Press.

Gross, H., Stephan, V., & Krabbes, M. (1998). A neural field approach to topological reinforcement learning in continuous action spaces. *IEEE World Congress on Computational Intelligence, WCCI'98 and International Joint Conference on Neural Networks, IJCNN'98.*

Hodgson, D. (2004). *Half-life 2:raising the bar*. Prima Games.

Kleinberg, J. (2001). Small-world phenomena and the dynamics of information. *Advances in Neural Information Processing Systems 14*. Cambridge MA: The MIT Press.

Lin, L. G. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*, 293–321.

Millan, J. D. R., Posenato, D., & Dedieu, E. (2002). Continuous-action q-learning. *Machine Learning*, *49*, 241–265.

Rabin, S. (2002). *Ai game programming wisdom*. Charles River Media, Inc.

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

Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. *Artificial intelligence*, *99(1)*, 21–71.

Toussaint, M. (2003). Learning a world model and planning with a self-organizing, dynamic neural system. *Advances in Neural Information Processing Systems 16*. Cambridge, MA: The MIT Press.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279–292.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*, 404–407.