Graph Laplacian Based Transfer Learning Methods in Reinforcement Learning

Yi-Ting Tsao, Ke-Ting Xiao, Von-Wun Soo and Chung-Cheng Chiu
Department of Computer Science, National Tsing Hua University
HsinChu, Taiwan

1. Introduction

In the real world, people often reuse their knowledge in dealing with daily life problems. They can observe facts in an environment and recall similar experience in the past to deal with new situations. This phenomenon implies that there must be some features for people to compare the similarity between two environments. For example, toilet papers are usually placed nearby cashiers in different marts in Taiwan as shown in Fig. 1. In these two photos, orange ovals represent features for cashiers and red ovals represent features for toilet papers. The features which allow people to recognize the fact "Toilet papers are usually placed nearby cashiers." are the kinds of experience which could be reused.

Fig. 1. Two different marts in Taiwan

One of disadvantages in reinforcement learning (Kimberly & Mahadevan) (Sutton & Barto, 1998) is that two different tasks with different initial states and goal states must be learned to acquire good policies separately. It would waste time to simply learn twice in two different tasks if they share some similar subtasks. Transfer learning is an approach to improve the performance of cross tasks by avoiding redundancy. Some previous work show that transferring knowledge between two tasks could speed up learning (Matthew E. Taylor, Stone, & Liu, 2005). In reinforcement learning, the value function provides a guideline for

action selection in a given state. In other words, the value function could be converted to the corresponding policy, which guides action selection. Therefore, transferring the value function is an intuitive approach in reinforcement learning.

The aim of transfer learning is to reuse learned knowledge from a source task to accelerate learning in a related target task. Many transfer methods which are based on different features, such as the value function or the policy, have been proposed (Hessling & Goel, 2005; Liu & Stone, 2006; Matthew E. Taylor & Stone, 2007; Mattew E. Taylor, Whiteson, & Stone, 2007). Some researchers propose a rule transfer method which is based on case-based reasoning. They acquire some rules by approximating the policy in a source task and then translate them into corresponding rules, which could be used as the policy for a target task (Hessling & Goel, 2005). In more details, they train a decision tree as rules with respect to the value function in a source task and then reuse the decision tree in the target task. In order to transfer, they assume that two tasks have similar descriptions. In addition, some researchers represent the policy as a neural network in a source task and transfer it to a target task (Mattew E. Taylor et al., 2007). However, it requires some hand-coded translation functions. Some researchers represent states and actions as qualitative dynamic Bayes networks (QDBNs) and find their mapping between a source task and a target task (Liu & Stone, 2006). However, finding the mapping needs a lot of efforts. The major problem of the above methods is the use of translation functions that are problem dependent and thus difficult to be defined, even by an expert.

A novel transfer method which is based on proto-value functions has been proposed (Kimberly & Mahadevan, 2006; Mahadevan, 2005; Mahadevan & Maggioni, 2006, 2007). Proto-value functions, which are derived from spectral graph theory, harmonic analysis, and Riemannian manifold, could be used to represent a set of basis functions to approximate a function. This method reuses proto-value functions from a source task and just learns their weights in composing the value function for a target task. Therefore, an advantage of this method is that it transfers from a source task to a target task without any translation function. However, it needs some exploring trials in a target task to acquire accurate weights for proto-value functions.

Reusing learned knowledge could save some time by avoiding redundant learning. Transfer learning is an approach to achieve it. In this chapter, we propose transfer methods to obtain a better prior policy from a source task to reduce learning time in a target task without hand-coded translation functions by graph Laplacian. Graph Laplacian, which is constructed by the topology of the state space, are problem independent, so it is helpful for transfer. In the following sections, we will introduce our transfer methods step-by-step. In section 2, we introduce some background knowledge such as Markov decision process (MDP), reinforcement learning, graph Laplacian, and etc. In section 3, we illustrate our transfer methods in detail. In section 4, we show experimental results on our transfer methods. In section 5, we conclude and discuss future work.

2. Background

2.1 Markov Decision Process

Markov decision process (Puterman, 2005) is a specification of a sequential decision problem with a Markovian transition model and additive rewards. Markov decision process is defined by 4-tuple $(S, A, P^a_{ss'}, R^a_{ss'})$, where S denotes a finite set of states, A denotes a finite

set of actions, $P^a_{ss'}$ denotes the transition probability of taking action a from state s to state s', and $R^a_{ss'}$ denotes the reward for transiting from state s to state s' with action a. A function which determines an agent's action in any state on Markov decision process is called a policy π. In other words, a policy is a mapping from a state to a unique action. A value function V_π maps each state to its expected reward with respect to a policy π as shown in (1), where γ denotes a discount factor and $\pi(s,a)$ denotes the corresponding probability of taking action a in state s. The equation (1) is also called Bellman equation.

$$V_\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'}(R^a_{ss'} + \gamma V_\pi(s')) \tag{1}$$

An optimal policy π^* maps each state to a specific action to maximize the expected total discounted reward and an optimal value function V_{π^*} corresponds to the optimal policy π^* as shown in (2).

$$V_{\pi^*}(s) = \max_a \sum_{s'} P^a_{ss'}(R^a_{ss'} + \gamma V_\pi(s')) \tag{2}$$

The value function could be represented in tabular form with one output for each input tuple. However, some state space in the real world is too huge to memorize the tabular form of the value function. Approximating the value function in terms of a linear combination of a set of basis functions as shown in (3) is an apporach to deal with this problem. For each i, v_i denotes a basis function and w_i denotes a corresponding weight.

$$V_\pi = w_1 v_1 + \ldots + w_n v_n \tag{3}$$

Represneting a function by a linear combination of basis functions could save a lot of memory. However, different sets of basis functions might affect the performance of functional approximation and the preformance directly impacts an agent's behavior. In other words, a suitable set of basis functions plays an important role for an agent's behavior on Markov decision process.

2.2 Reinforcement Learning

Reinforcement learning (Sutton & Barto, 1998) is about learning from interaction to achieve the goal. A reinforcement learning problem is based on Markov decision process. In other words, a reinforcement learning problem which satisfies the Markov property[1] is called Markov decision process. Some reinforcement learning problems do not satisfy the Markov property in the real world, but they still could be approximated by the Markov assumption. Most reinforcement learning methods are based on estimating the value function[2] by

[1] Roughly speaking, if deciding a next state only requires using current information, it satisfies the Markov property.

[2] The value function includes two types: the state-value function and the action-value function. In this paper, we focus on the state-value function.

approximately solving the Bellman equation. Some other learning methods are also based on estimating the value function. A major difference is that the reinforcement learning methods put more efforts into learning to make good decisions for frequently encountered states and less efforts for infrequently encountered states.
Temporal-difference (TD) learning, which combines the Monte Carlo method and dynamic programming, is a central concept in reinforcement learning. Temporal-difference learning estimates the value function of one state from the next state without waiting for an actual final outcome as shown in (4), where V denotes the value function, s denotes the current state, s' denotes the next state, $R_{ss'}^{a}$ denotes the reward for transiting from state s to state s' with action a, α denotes the learning rate, and γ denotes the discount factor.

$$V(s) = V(s) + \alpha[R_{ss'}^{a} + \gamma V(s') - V(s)] \tag{4}$$

The value function guides an agent's behavior on Markov decision process and reinforcement learning learns the value function by continuously updating. Therefore, the updating method plays an important role in an agent's performance.

2.3 Graph Laplacian

The Fourier analysis is to decompose a function in terms of a sum of trigonometric functions with different frequencies. In other words, the trigonometric functions could be combined together to represent the function. In addition, each frequency of trigonometric functions is inversely proportional to its importance as representing more features of the function. If two functions are similar, their trigonometric functions tend to be the same at low frequencies and the difference at high frequencies.
Graph Laplacian can be defined as the combinatorial Laplacian or the normalized Laplacian (Chung, 1997). The combinatorial Laplacian of an undirected unweighted graph G is defined as an operator $L = D - A$, where A is the adjacency matrix and D is a diagonal matrix whose entries are the row sums of A. In other words, the combinatorial Laplacian could represent the connection (undirected) or the transition (directed) between two vertices u and v as shown in (5), where d_v denotes the degree of vertex v without the self loop. In problem solving, states are represented as vertices and connections or transitions between states are represented as edges.

$$L(u,v) = \begin{cases} d_v & \text{if } u = v \\ -1 & \text{if } u \text{ and } v \text{ are adjacent} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Let f denote a function mapping each vertex u in a graph into a real number. The combinatorial Laplacian L acts on a function f as shown in (6), where $u \sim v$ denotes

vertex u and vertex v are adjacent. To minimize the equation $\sum_{u\sim v}(f(u)-f(v))^2$ [3] subject to f with condition, which f is a unit vector, is equivalent to solving the eigenproblem of L as shown in (7), where λ denotes the eigenvalue and f denotes the eigenfunction. By the spectral theorem (Chung, 1997), eigenfunctions with respect to smaller eigenvalues are smoother. In other words, the smoothness of eigenfunctions is inversely proportional to their eigenvalues.

$$Lf(u)=\sum_{u\sim v}(f(u)-f(v)) \tag{6}$$

$$Lf=\lambda f \tag{7}$$

Furthermore, the normalized Laplacian $\widetilde{L}$ of a graph is defined as $\widetilde{L}=D^{-1/2}LD^{-1/2}$ and each eigenfunction of $\widetilde{L}$ is defined as $g=D^{-1/2}f$, where f denotes each eigenfunction of L. The difference between the combinatorial Laplacian L and the normalized Laplacian $\widetilde{L}$ is that the normalized Laplacian models the degree of a vertex as a local measure.

The spectral analysis of graph Laplacian operator provides an orthonormal set of basis functions that can approximate any square-integrable functions on a graph (Chung, 1997). These basis functions, which are a set of eigenfunctions of L or $\widetilde{L}$, are called as proto-value functions (Kimberly & Mahadevan, 2006; Mahadevan, 2005; Mahadevan & Maggioni, 2006, 2007). Proto-value functions construct a global smooth approximation of a function on a graph. In other words, a function on a graph could be decomposed into a linear combination of proto-value functions.

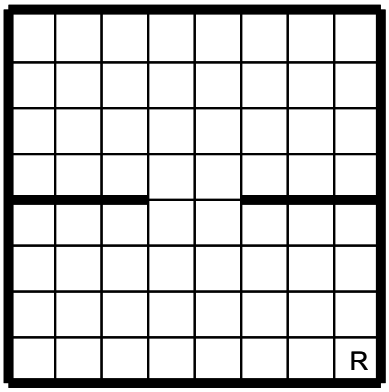
Therefore, the notion of the spectral analysis on graph Laplacian is similar to the Fourier analysis. Basis functions of graph Laplacian corresponding to the smaller eigenvalues represent more features and are more important. It also implies that if two graphs are similar, their features tend to be the same at low-order basis functions and the difference at high-order basis functions.

2.4 Transfer Types

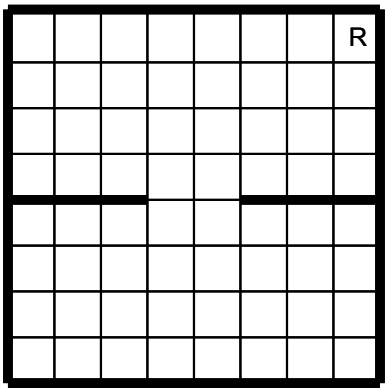
In previous work (Kimberly & Mahadevan, 2006), the authors have proposed three transfer types: task transfer, topological domain transfer, and scaling domain transfer as shown in Fig. 2. The task transfer problem means that the size of states and the transition model does not change but the rewards change. For example, tranferring from Fig. 2(a) to Fig. 2(b) is a task transfer problem and vice versa. The domain transfer problem means that the size of states or the transition model changes but the rewards are still the same. In detail, the scaling domain transfer problem is the change of the size of states and the topological domain transfer problem is the change of a transition model. For example, tranferring from

[3] Minimizing the equation $\sum_{u\sim v}(f(u)-f(v))$ has the same result as minimizing the equation $\sum_{u\sim v}(f(u)-f(v))^2$.

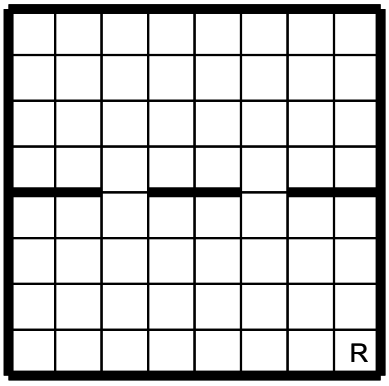
Fig. 2(a) to Fig. 2(c) is a topological domain transfer problem and from Fig. 2(a) to Fig. 2(d) is a scaling domain transfer problem. These three transfer types are symmetric which means that if transferring from graph G^S to graph G^T is one of transfer types, transferring from graph G^T to graph G^S is the same transfer type. Notice that R denotes a reward in a state, but rewards could be gained after any state transition in general case.



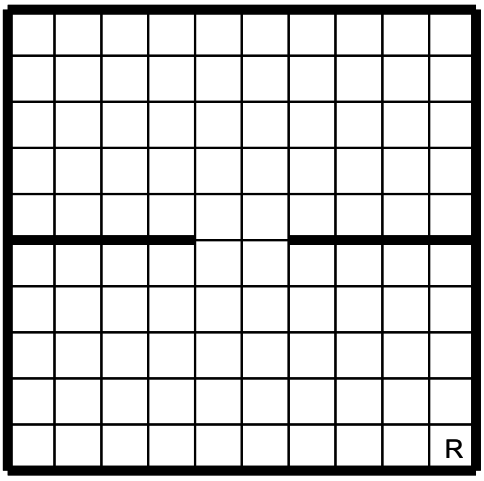
(a) source



(b) the task transfer



(c) the topological domain transfer



(d) the scaling domain transfer

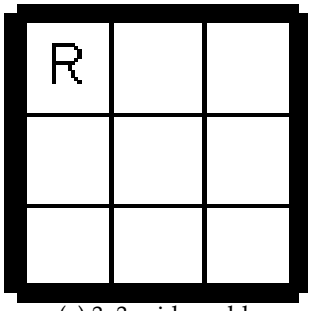
Fig. 2. Examples of transfer types

3. Methodology

3.1 An Example

Before we describe how to transfer, we show a simple example for the combinatorial Laplacian and a simple scenario for the transfer problem. A 3x3 grid world and its corresponding state transitioin graph are shown in Fig. 3(a) and Fig. 3(c). A state is defined

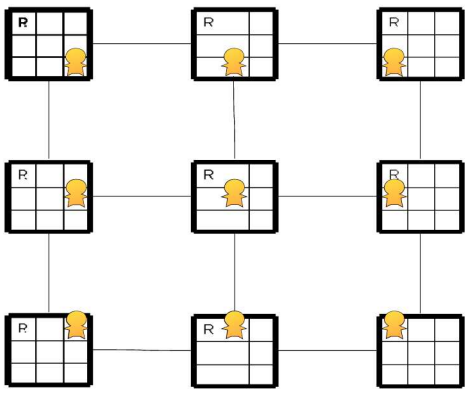
as an agent at one of cells in the grid world and the state transition graph shows the possible transitions from one state to another. By definitioin in section 2.2, we could derive the combinatorial Laplacian as shown in Fig. 3(b). The diagonal terms denote the degree of states and the others denote the connection. Notice that the combinatorial Laplacian does not only describe the grid world problems, but also others. For example, the task of putting on a pair of shoes (Russell & Norvig, 2003) is defined as an agent who wants to put on shoes with a condition of putting on socks before shoes. The state transition graph of this problem is shown in Fig. 3(d). By camparing Fig. 3(c) and Fig. 3(d), we could find that the two graphs are the same. It means that their combinatorial Laplacians are also the same. Therefore, we could do the domain transfer between these two tasks.



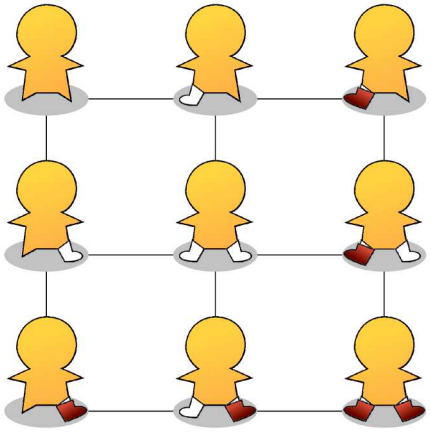
(a) 3x3 grid world

$$\begin{bmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 3 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 3 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 2 \end{bmatrix}$$

(b) the combinatorial Laplacian



(c) the state transition graph of (a)



(d) the state transition graph of the problem of putting on a pair of shoes

Fig. 3. A simple example

3.2 A Simple Transfer Method

In this section, we describe a simple transfer method which is based on transferring the value function. We represent the value function by a linear combination of basis functions and the idea is transferring the weights between two similar tasks whose details are described in Fig. 4. The first step is to collect the knowledge of state transitions in both tasks.

The second step is to construct the normalized Laplacian by the collected state transitions. The third step is to compute the corresponding basis functions of the normalized Laplacians. The fourth step is to obtain the weights of the source basis functions by approximating the source value function. The fifth step is to approximate the target value function in terms of the target basis functions and the obtained weights. The last step is to acquire the target policy through the approximated target value function.

1. Perform N-steps random walk to obtain M trials on a source task and a target task respectively.
2. Construct the normalized Laplacians $\widetilde{L}^S$, by the undirected graphs G^S, G^T which are obtained by the trials.
3. Solve the eigenproblems of $\widetilde{L}^S$, $\widetilde{L}^T$ to obtain the basis functions $\{v_i^S\}$, $\{v_i^T\}$ which are ordered by the ascending eigenvalues.
4. Approximate the source value function $V_{\pi^*}^S$ to obtain the weights $\{w_i^S\}$ corresponding to $\{v_i^S\}$ by the least-square error fit method.
5. Transfer the weight $\{w_i^S\}$ from $\{v_i^S\}$ to the corresponding target basis functions $\{v_i^T\}$.
$$V_{\pi'}^T = \sum_i w_i \cdot v_i^T$$
6. Convert the approximation target value function $V_{\pi'}^T$ to the target policy π'.

Fig. 4. A simple transfer method

The reason why the simple transfer method works is that basis functions of both tasks with the same order play the same important role for both value functions. Therefore, we transfer the obtained weights from a source task to a target task. If two tasks are similar, two sets of basis functions tend to be similar. Notice that it does not imply that numeric values are similar but the structure is similar as shown in Fig. 5. On the one hand, a small difference between two tasks cannot affect the global smooth structure so the both low-order basis functions tend to be the same. On the other hand, the high-order basis functions are affected by a small change so the target policy could obtain from the similar low-order basis functions and the different high-order basis functions. For example, the basis functions in Fig. 5 are the lower-order ones and the basis functions in Fig. 6 are the high-order ones.

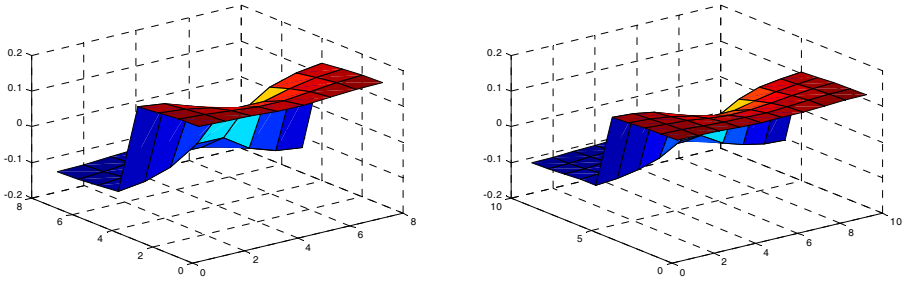


Fig. 5. The similar structure of the basis functions of Fig. 2(a) and Fig. 2(d)

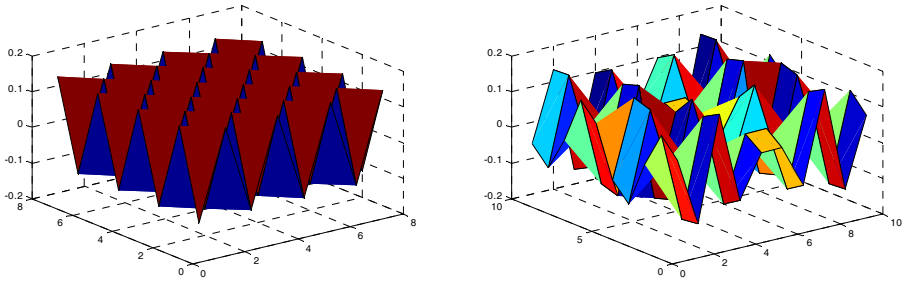


Fig. 6. The different structures of the basis functions of Fig. 2(a) and Fig. 2(d)

3.3 Modified Graph Laplacian

In section 2.3, we introduce the graph Laplacian and the smoothness property of its corresponding eigenfunction. In this section, we assume that each state transition is bidirectional and a positive circular reward does not exist for every task, which means that both edges, $u \sim v$ and $v \sim u$, have positive rewards. Then, the modified graph Laplacian L' of a directed graph is defined in (8), where S_v denotes the entry sum of the v-th row. Roughly speaking, the modified graph Laplacian treats the state with a positive reward as a termination.

$$L'(u,v) = \begin{cases} -S_v & \text{if } u = v \\ -1 & u \sim v \text{ and } v \sim u \text{ without a positive reward} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

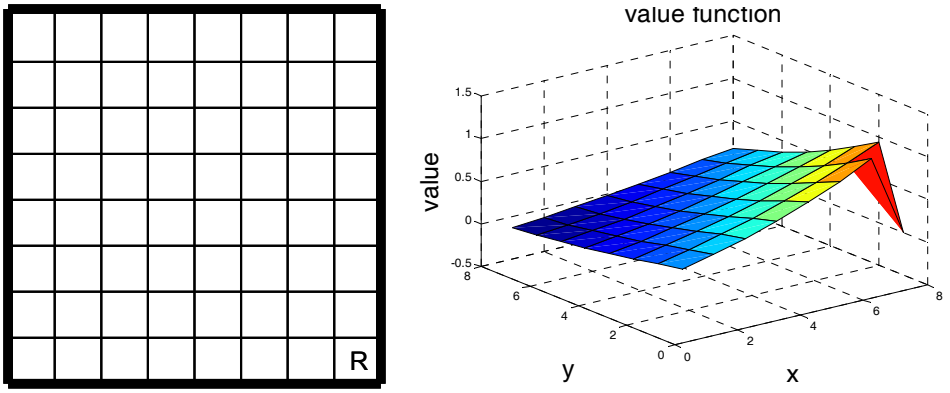
Let f denote a function mapping each vertex u in a graph into a real number and the modified graph Laplacian L' acts on f as shown in (9), where $u \approx v$ denotes $u \sim v$ and $v \sim u$ without a positive reward. To minimize the equation (9) subject to f with the condition which f is a unit vector is equivalent to solving the eigenproblem of L'. It is similar to the graph Laplacian case.

$$L'f(u) = \sum_{u \approx v} (f(u) - f(v)) \tag{9}$$

Because the graph Laplacian L is a positive semidefinite matrix, the eigenvalues of L are non-negative real numbers. To analyze the eigenvalues of the modified graph Laplacian L' we observe the characteristic equation of the modified graph Laplacian L' as shown in (10), where $\hat{L}$ denotes the combinatorial Laplacian L without i-th row and column, which $L'(i,i)$ is equivalent to zero. By the definition, $\hat{L}$ is a possible graph Laplacian. Therfore, $\hat{L}$ is a positive semidefinite matrix and its eigenvalues are non-negtaive numbers. Furthermore, we could derive that the eigenvalues of L' are still non-negative and the normalized version $\widetilde{L}' = D'^{-1/2} L' D'^{-1/2}$, where D' denotes a matrix with diagonal terms of L'.

$$\det(L' - \lambda I) = (-\lambda)^k \cdot \det(\hat{L} - \lambda I) \tag{10}$$

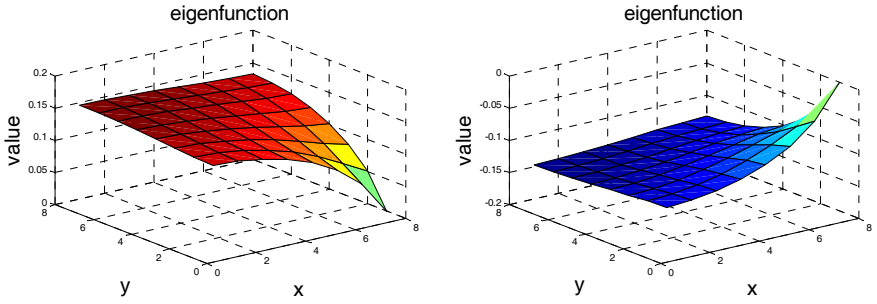
The eigenfunctions with respect to different eigenvalues represent different levels of smoothness. Therefore, the eigenfunction with respect to the first nonzero eigenvalue on the modified graph Laplacian is the smoothest. In most cases, the value function tends to be smooth. By the observation, we find the eigenfunction with respect to the first nonzero eigenvalue have the similar behavior tendency as its value function. An simple task and its value function are shown in Fig. 7, where R denotes a reward to illustrate the tendency. In this grid world task, an agent in each cell represents a state and its topology represents the possible state transitions. An agent reaches the state with R to obtain a reward $+1$ and terminate, otherwise a penalty -0.04 . By the definition, we construct the modified graph Laplacian as shown in (11). Then, we compute the eigenfunction with respect to the first nonzero eigenvalue as shown in Fig. 8. Because the eigenfunction is a vector, it have two possible directions. For convenience, if all values are non-negative, it is called the positive eigenfunction, otherwise the negative eigenfunction. By the definition (Sutton & Barto, 1998), the value of a terminal state in value function is zero and the value of the state which is adjacent to a positive reward is close to the value of the reward. Therefore, we could expect the value function and the negative eigenfunction with respect to the first nonzero eigenvalue to be similar and thus we could transfer the value function by the negative eigenfunction with respect to the first nonzero eigenvalue.



(a) (b)

Fig. 7. A 8x8 grid world task and its optimal value function

$$\begin{matrix} & & & & & \downarrow \text{state with } R \\ \begin{bmatrix} 2 & -1 & \cdots & 0 & 0 \\ -1 & 3 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 3 & -1 \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix} & & & & & \leftarrow \text{state with } R \end{matrix} \tag{11}$$



(a) the positive eigenfunction (b) the negative eigenfunction

Fig. 8. The eigenfunctions of the 8x8 grid world in Fig. 7(a) with respect to the first nonzero eigenvalue

3.4 Transfer Method

In this section, we describe a transfer method which is based on the tendency of the eigenfunction with respect to the first nonzero eigenvalue of the modified graph Laplacian. The detail of the transfer method is shown in Fig. 9. The first step is to collect the knowledge of state transitions in both tasks. The second step is to construct the normalized modified Laplacian by the collected state transitions. The third step is to compute the corresponding negative eigenfunctions with respect to the first nonzero eigenvalue of the normalized modified Laplacians. The fourth step is to sort the eigenfunctions in descending order respectively to obtain the one-to-one state mappings which map states in the source task to the corresponding ones in the target task. The fifth step is to map the values of states in the source task to the corresponding ones in the target task. The last step applies only for the case with different state sizes. If the number of states in the target task is bigger than in the source task, some states do not obtain the mapping states in the step 4. Therefore, the extrapolated method is used to estimate their value in terms of the negative eigenfunction in the target task and the value function in the source task. If the number of states in the target task is smaller than in the source task, all states in the target task can find the mapping states in the source task and some states in the source task are useless.

1. Perform N-steps random walk to obtain M trials on a source task and a target task respectively.
2. Construct the normalized modified Laplacians $\widetilde{L}'^{S}$, $\widetilde{L}'^{T}$ by the directed graphs G^{S}, G^{T}, which are obtained by the trials.
3. Solve the eigenproblems of $\widetilde{L}^{S}$, $\widetilde{L}^{T}$ to obtain the negative eigenfunctions with respect the first nonzero eigenvalue v_1^{S}, v_1^{T}.
4. Sort the negative eigenfunctions v_1^{S}, v_1^{T} in descending order respectively to obtain the one-to-one state mappings.
5. Map the values of the source value function to the values of the corresponding states in the target task.
6. (optional) If the number of states in the target task is bigger than that in the source task, an extrapolated method is used to estimate the rest of states.

Fig. 9. The transfer method

4. Experiments

4.1 Setting

These experiments investigate the effects of the simpler transfer method and the transfer methods by three transfer types. The transition model is shown in Fig. 10. It means that when an agent takes an action in a state, the consequence is not deterministic. For example, if an agent goes forward in a state, the possible next states can be the forward state, the left state and the right state. Notice that the symbol R denotes a terminal state with reward $+1$ and any state transitions could not reach the terminal state with a penalty -0.04. We compare the results by an ε-greedy TD learning agent which means that the agent takes an action which is not according to the policy with probability ε. We set $\varepsilon = 0.1$, the learning rate $\alpha = 0.1$ and the discount factor $\gamma = 0.9$.

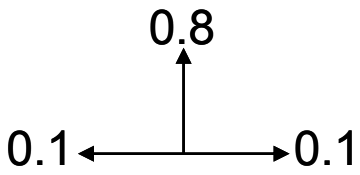
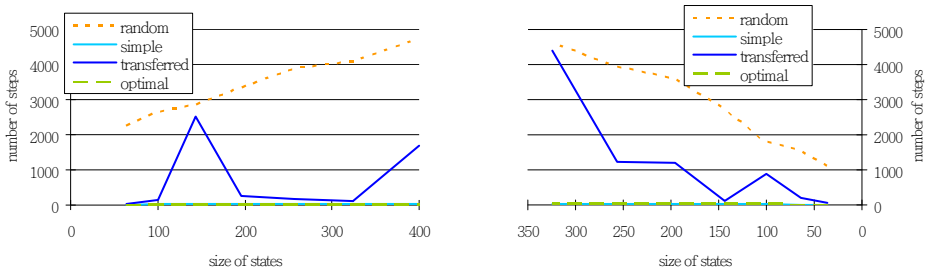


Fig. 10. The transition model in the experiments

The goal of these experiments is to understand the performance and the accelerated effects. To calculate the steps we assume that the upper left corner is the start state. In the domain transfer cases, we compare the steps of reaching the reward of a random policy, the simple transferred policy, the transferred policy and the optimal one as the performance evaluations. In the task transfer cases, we compare the steps of reaching the reward of a random policy and the transferred policy to evaluate the performance. In addition, to show the accelerated effects, we compare the convergence using a random initial policy and the transferred initial policy for all cases.

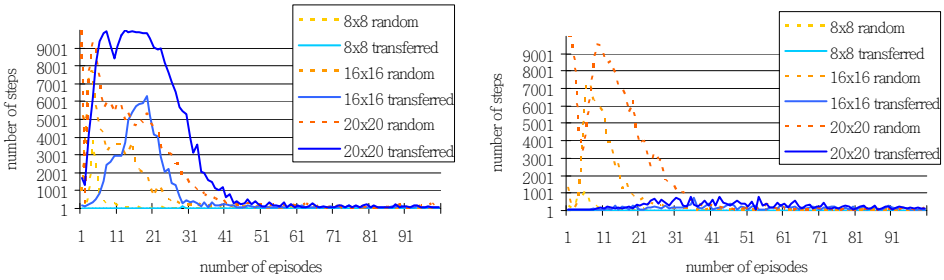
4.2 Scaling Domain Transfer

To investigate the performance of the simple transferred policy we separate the scaling domain transfer into two cases: the up-scaling case and the down-scaling case. The topology of the task is the same as Fig. 2(a). In the up-scaling case, we choose the 6x6 grid world as a source task and 8x8, 10x10, 12x12, 14x14, 16x16, 18x18, and 20x20 grid world as target tasks. In down-scaling case, we choose the 20x20 grid world as a source task and 6x6, 8x8, 10x10, 12x12, 14x14, 16x16, and 18x18 grid world as target tasks. The results are shown in Fig. 11, where the simple transferred policy is derived from the simple transfer method and the transferred policy is derived from the transfer method. We could discover that regardless of the size is changed in a target task, the simple transferred policy still performs very close to the optimal policy and the transferred policy doe not always perform well. Therefore, we investigate the accelerated effect of the transfer method with different topologies in the up-scaling case as shown in Fig. 2(a) and Fig. 7(a). The results are shown in Fig. 12. We could discover that different topologies have different effects and the transfer method is not always good for the scaling domain transfer.



(a) the up-scaling case (b) the down-scaling case

Fig. 11. The performance of transferred policies in the scaling domain transfer

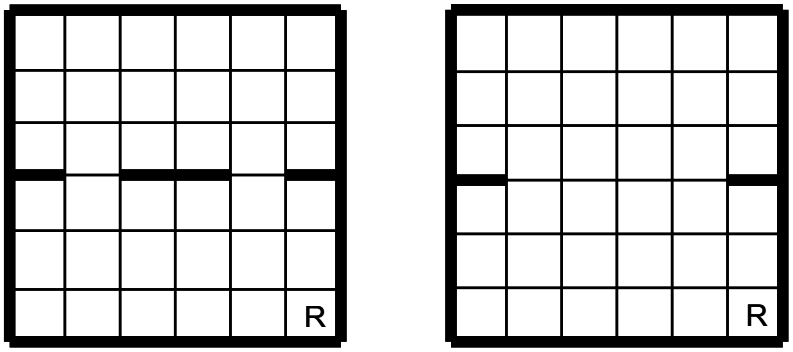


(a) corresponding to Fig. 2(a) (b) corresponding to Fig. 7(a)

Fig. 12. The accelerated effect of the transfer method in the scaling domain transfer

4.3 Topological Domain Transfer

The source task is shown in Fig. 2(a) and the target tasks are shown in Fig. 13. Fig. 13(a) represents that the door is separated into two doors and the distances between each door and the center is equal to a unit. Fig. 13(b) represents that the size of the door is increased. We investigate the topological domain transfer in different sizes as follows: 6x6, 8x8, 10x10, 12x12, 14x14, 16x16, 18x18, 20x20. The performance of the transferred policies is shown in Fig. 14. We could discover that different transfer methods are good for different topological domain transfer tasks. Although sometimes the transferred policy is not as good as the optimal policy, if the convergence is good enough, it is still a pretty good transfer. That is one of reasons why we take the accelerated effect into consideration. Another reason is that even though a policy is acceptable so far, it is possible to have a bad performance in a bigger task. The accelerated effect of the transfer method is shown in Fig. 15. We discover that the convergence of the transferred policy is faster than the random policy in both cases.



(a) (b)

Fig. 13. The target tasks of the topological domain transfer

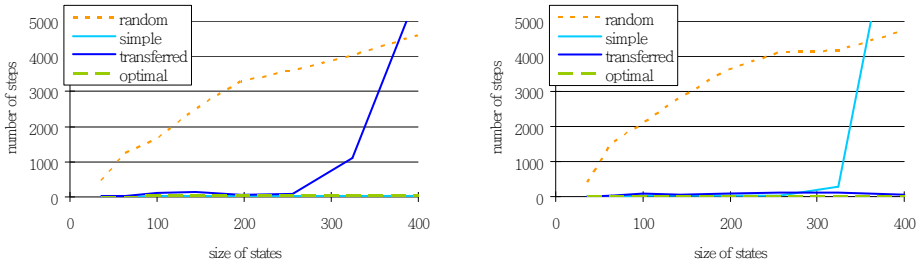


Fig. 14. The performance of transferred policies in the topological domain transfer

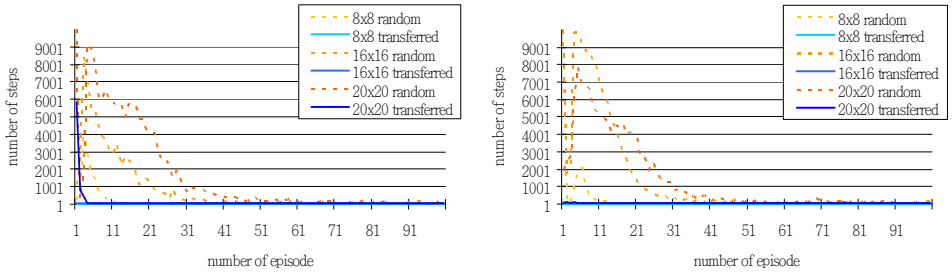


Fig. 15. The accelerated effect of the transfer method in the topological domain transfer

4.4 Task Transfer

So far, we compare only the simple transfer method and the transfer method in the domain transfer cases. In this section, we investigate the transfer method in the task transfer. The reason why we do not discuss the simple transfer method is that it could not use in the task transfer because it does not take the reward into consideration. The source task and the target tasks are shown in Fig. 16. Fig. 16(a) represents the source task and the Fig. 16(b), (c), (d), (e) represent the target tasks with different rewards. Notice that we investigate the task transfer in a fixed size 10x10 because we think the task transfer is independent to the size.

The performance of the transfer method is shown in Fig. 17. We could discover that the transferred policy is much better than the random policy. The accelerated effect of the transferred method is shown in Fig. 18. The convergence is obviously much faster than the random policy. In other words, the transfer method could accelerate learning in the task transfer cases.

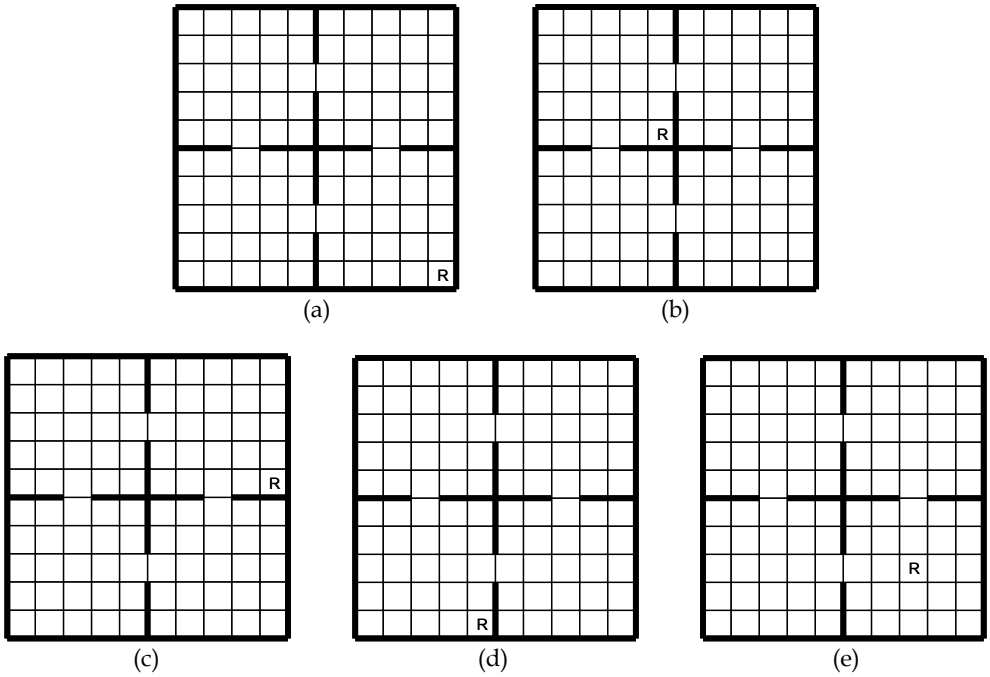


Fig. 16. The source and target tasks of the task transfer

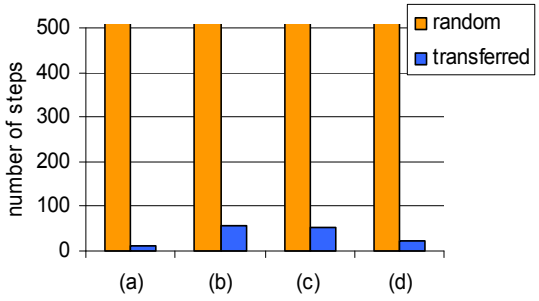


Fig. 17. The performance of the transfer method in the task transfer

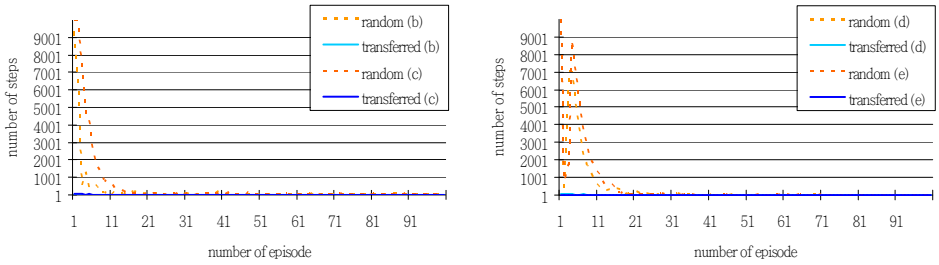


Fig. 18. The accelerated effect of the transfer method in the task transfer

4.5 Synthetic Transfer

In this section, we synthesize the scaling domain transfer, the topological domain transfer and the task transfer to be a synthetic transfer. The synthetic transfer is like transferring a maze to another. The source task and the target tasks are shown in Fig. 19. Notice that these three tasks are randomly generated with the condition that each state could reach the terminal state. The accelerated effects of the transfer method are shown in Fig. 20. The results show that the transfer method is not only used in one of transfer types, but also in the synthetic case. That is the reason why we discuss the transfer method rather than the simple transfer method.

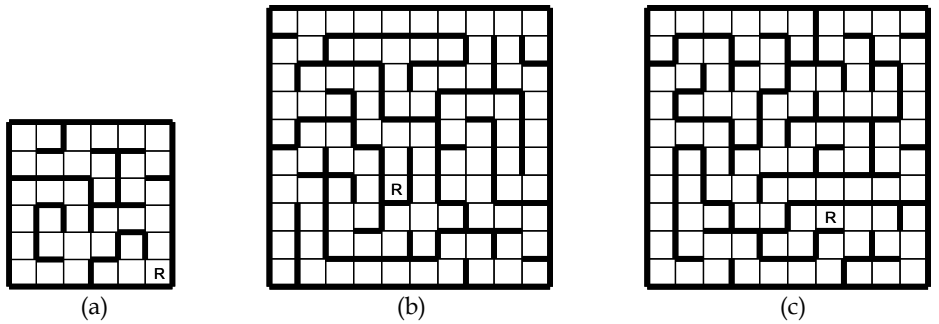


Fig. 19. The source task (a) and the target tasks (b) and (c) of the synthetic transfer

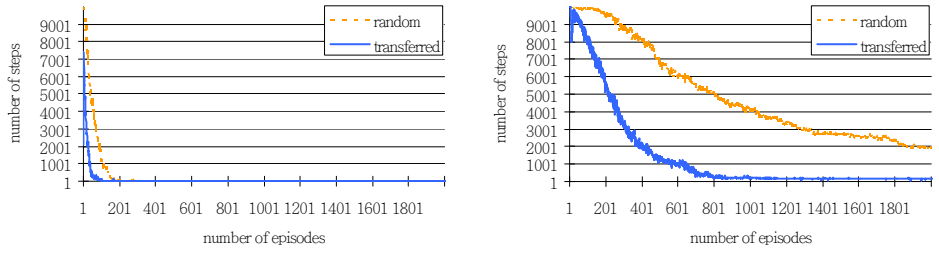


Fig. 20. The accelerated effect of the transfer method in the synthetic transfer

5. Discussions

The theoretical analysis of the simple transfer method is based on the spectral analysis on graph Laplacian. Low-order basis functions of graph Laplacian tend to represent more features of the value functions and high-order basis functions tend to represent fewer features. If low-order basis functions of two tasks are similar, the simple transfer method performs well. In other words, similar tasks tend to keep similar structures in low-order basis functions so transferring weights from one task to another could acquire a good approximate policy. The experimental results show that if two tasks are similar, the transferred policy of the simple transfer method could be very close to the optimal one. However, even though the simple transfer method seems to be good in the domain transfer cases, it could not be used in the task transfer. Furthermore, it still needs more theoretical analysis as to determine if topological similarity is close enough to apply the simple transfer method that ensures the simple transferred policy to be close to the optimal one.

The transfer method could be used in three transfer types: the scaling domain transfer, the topological domain transfer and the task transfer. However, the transfer method is not always better than the simple transfer method. The experimental results show that the transferred policy of the transfer method converges earlier than the random policy. In other words, the evidence demonstrates the accelerated effect of the transfer method. The reason why the transfer method could work in the task transfer is taking rewards into consideration on the modified graph Laplacian. However, how to evaluate the accelerated effect of the transfer method in more objective manner is a challenge because different tasks tend to have different effects.

In this chapter, we have proposed the transfer method based on the topology of state transitions for reinforcement learning. It could be used in three transfer types: the scaling domain transfer, the topological domain transfer and the task transfer. Because the transfer method is transferring the state-value function, we need a perfect transition model to obtain the policy. However, to obtain the perfect transition model sometimes is not easy so extending this idea to the action-value function might be an approach to avoid this problem. Because the transfer method only deals with the discrete tasks, mapping continuous tasks to discrete tasks might be an approach to deal with the transfer in continuous tasks.

6. References

Chung, F. R. K. (1997). *Spectral graph theory*, American Mathematical Society.

Hessling, A. v., & Goel, A. K. (2005). Abstracting reusable cases from reinforcement learning. *Proceedings of the Sixth International Conference on Case-Based Reasoning Workshop*.

Kimberly, F., & Mahadevan, S. (2006). Proto-transfer learning in Markov decision processes using spectral methods. *Proceedings of the Twenty-Third International Conference on Machine Learning Workshop on Structural Knowledge Transfer for Machine Learning*.

Liu, Y., & Stone, P. (2006). Value-function-based transfer for reinforcement learning using structure mapping. *Proceedings of the Twenty-First National Conference on Artificial Intelligence*.

Mahadevan, S. (2005). Proto-value functions: Developmental reinforcement learning. *Proceedings of the Twenty-Second International Conference on Machine Learning*.

Mahadevan, S., & Maggioni, M. (2006). Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Technical Report*.

Mahadevan, S., & Maggioni, M. (2007). Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research, 8*, 2169-2231.

Puterman, M. L. (2005). *Markov decision processes discrete stochastic dynamic programming*, Wiley.

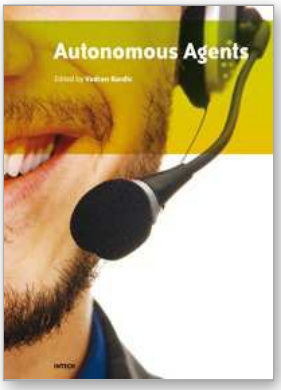
Russell, S., & Norvig, P. (2003). *Artificial intelligence a modern approach*, Prentice Hall.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning an introduction*, MIT press.

Taylor, M. E., & Stone, P. (2007). Cross-domain transfer for reinforcement learning. *Proceedings of the Twenty-Fourth International Conference on Machine Learning*.

Taylor, M. E.; Stone, P., & Liu, Y. (2005). Value functions for RL-based behavior transfer: A comparative study. *Proceedings of the Twentieth National Conference on Artificial Intelligence*.

Taylor, M. E.; Whiteson, S., & Stone, P. (2007). Transfer via inter-task mappings in policy search reinforcement learning. *Proceedings of the Sixth International Conference on Autonomous Agents and Multiagent Systems*.



Autonomous Agents
Edited by Vedran Kordic

ISBN 978-953-307-089-6
Hard cover, 130 pages
Publisher InTech
Published online 01, June, 2010
Published in print edition June, 2010

Multi agent systems involve a team of agents working together socially to accomplish a task. An agent can be social in many ways. One is when an agent helps others in solving complex problems. The field of multi agent systems investigates the process underlying distributed problem solving and designs some protocols and mechanisms involved in this process. This book presents a combination of different research issues which are pursued by researchers in the domain of multi agent systems.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821