On the Bottleneck Concept for Options Discovery:

Theoretical Underpinnings and Extension in Continuous State Spaces

Pierre-Luc Bacon

Computer Science
McGill University, Montreal

October 30, 2013

A thesis submitted to McGill University in partial fulfilment of the requirements of the degree of Master of Science.

Dedication

This thesis is dedicated my niece Anne.

Acknowledgements

I would like to thank everyone with whom I had the chance to share my enthusiasm upon exploring new ideas; those who gave me the freedom to fail, thinker, and be slightly obsessed on certain days.

I am deeply grateful to my supervisor Doina Precup, Amir-massoud Farahmand, Gheorghe Comanici and Cosmin Paduraru for their precious insights and encouragement. Special thanks to lab members Clement Gehring and Neil Girdhar for their persistence in constructively challenging my ideas.

Abstract

The bottleneck concept in reinforcement learning has played a prominent role in automatically finding temporal abstractions from experience. Lacking significant theory, it has however been regarded by some as being merely a *trick*. This thesis attempts to gain better intuition about this approach using spectral graph theory. A connection to the theory of Nearly Completely Decomposable Markov Chains (NCD) is also drawn and shows great promise. An options discovery algorithm is proposed and is the first of its kind to be applicable in continuous state spaces. As opposed to other similar approaches, this one can have running time $\mathcal{O}(n^2 \log n)$ rather than $\mathcal{O}(n^3)$ making it suitable to much larger domains than the typical *grid worlds*.

Résumé

L'identification automatique de goulots d'étranglement dans la structure de solution a joué un rôle important en apprentissage par renforcement hiérarchique au cours des dernières années. Bien que populaire, cette approche manque toujours de fondements théoriques adaptés. Ce mémoire tente de pallier ces lacunes en établissant des liens en théorie spectrale des graphes, espérant ainsi obtenir une meilleure compréhension des conditions garantissant son applicabilité. Une revue des efforts réalisés concernant les chaines de Markov presque complètement décomposable (NCD) permet de croire qu'elles pourraient être utiles au problème ici considéré. Un algorithme de découverte d'options motivé par la théorie spectrale des graphes est proposé et semble être le premier du genre à pouvoir être aussi appliqué dans un espace d'états continu. Contrairement à d'autres approches similaires, la complexité algorithmique en temps peut être de l'ordre de $\mathcal{O}(n^2 \log n)$ plutôt que $\mathcal{O}(n^3)$, rendant possible la résolution de problèmes de plus grande envergure.

Contents

1

Introduction

The breakthroughs in hierarchical reinforcement learning (HRL) at the beginning of the last decade have promised to propel the field to new levels. Its development has however been hindered by an easily explainable, but yet challenging problem: how can temporal abstractions be discovered automatically ? It is the “elephant in the room” for HRL which, despite much effort, remains unsolved. The existence of this problem has been acknowledged by many authors. In T. G. Dietterich 2000, the author of the MAXQ framework goes as far as qualifying it as being the “biggest open problem” in HRL. Lacking automatic methods for decomposing the problem structure into simpler sub-problems, hierarchical approaches are of limited interest as the effort for manually specifying them can quickly become insurmountable. Hengst 2002 even refers to manual problem decomposition as being an “art-form” .

The bottleneck concept arose early in the field of HRL as an intuitive response to this issue. Bottlenecks have been defined as those states which appear frequently on successful trajectories to a goal but not on unsuccessful ones (McGovern and Barto 2001; Stolle and Precup 2002) or as nodes which allow for densely connected regions of the interaction graph to reach other such regions (Menache, Mannor, and Shimkin 2002; Şimşek and Barto 2004; Kazemitabar and Beigy 2009). Şimşek and Barto 2004 qualifies such states as *access states*, allowing difficult regions of the state space to be reached. The canonical example for the bottleneck concept often explains it with the *doorways* of some navigation problem in a grid-world domain.

The ideas presented in this thesis found their roots early in the history of HRL. It seems that two lines of approach have since been studied by a number of authors and could be recognized as either belonging to some *graph clustering*, or *classification* perspective. While aiming towards the same goal, the classification perspective might have been more influenced by the *online* and *model-free* requirements traditionally sought for in RL. It will be argued in chapter 4 that bottlenecks are intrinsically related to graph clustering and can be tackled more easily under this angle.

1.1 Graph Clustering Perspective

Hauskrecht et al. 1998 described a state space partitioning approach leading to an abstract MDP being overlaid to the periphery of the partitions. Macro-actions would then be learnt as transitions between such peripheral states. While a decomposition is assumed to be available *a priori*, a contribution of this paper was to prove that such a model can lead to a reduction in the size of the state or action spaces. A similar state space partitioning intuition had been previously explored by a collaborator in Dean and Lin 1995. Even though the method of Dean and Lin 1995 was not cast yet under the framework of reinforcement learning, it showed how a global planning problem could be decomposed into smaller locally-solvable problems which could be then be recombined into an optimal policy. The machinery or HRL being in its infancy, the Dantzig-Wolfe method of decomposition (Dantzig and P. Wolfe 1960) – a linear programming algorithm – was used to solve the abstract MDP. The problem of obtaining a proper partitioning was also sidestepped in this work.

Along the same line of work, Menache, Mannor, and Shimkin 2002 also proposed a decomposition algorithm called Q-Cut based on graph partitioning view of the state-transition graph. The Max-Flow/Min-Cut algorithm was used for finding the bottleneck states for which flat policies were constructed to reach them. The problem of representation and learning for temporal abstraction was cast under the options framework (R. S. Sutton, Precup, and Singh 1999) in which the SMDP Q-Learning algorithm could be applied. The approach was revisited in Mannor et al. 2004 where an agglomerative graph clustering algorithm was applied. Under this other perspective, options where defined between pairs of clusters rather than from arbitrary states to bottleneck states. Two types of graph clusterings were also defined: *topological* and *value-based*. The former attempted to group states into clusters based on structural regularities of the environment while the latter considered the reward information collected during learning. This agglomerative approach has recently been reformulated under an online setting in the *Online Graph-based Agglomerative Hiearchical Clustering* algorithm (OGAHC) of Metzen 2012.

The L-Cut algorithm of Şimşek, A. P. Wolfe, and Barto 2005 can be seen as an extension of Q-Cut under a local knowledge of the environment only. Rather than setting up the graph bisection problem under the Max-Flow/Min-Cut formulation, the NCut criterion of Shi and Malik 2000 was used to measure the quality of the partitioning and solved as as generalized eigenvalue problem. A local graph representation was obtained by collecting samples of experience through a random walk process. After a certain number of iterations, the spectral bipartionning algorithm was applied and a policy, encoded as an *option*, was learnt to reach the identified subgoals. A statistical test was applied on the set of boundary states of each partition to discriminate *useful* subgoals from noise.

The spectral clustering approach was revisited more recently in Mathew, Peeyush, and Ravindran 2012 under the Robust Perron Cluster Analysis of Weber, Rungsarityotin, and Schliep 2004 and their PCCA+ algorithm. While NCut and PCCA+ are both spectral techniques sharing much in common, Perron Cluster Analysis lends itself more easily to an interpretation in terms of metastability and invariant subsets of Markov chains. Intuitively, metastable regions correspond to subsets of states in which the stochastic process spends more time, switching to other such subsets on rare occasions. The *cascade decomposition* technique of Chiu and Soo 2010 is another recent attempt to find bottlenecks by computing the second eigenvector of the normalized graph Laplacian. It is in essence very similar to L-Cut which uses NCut Shi and Malik 2000, also derived from the normalized graph Laplacian.

Trying to address both the discovery problem and the related one of transfer learning, Bouvrie and Maggioni 2012 builds upon the diffusion maps framework of Coifman and Lafon 2006 and sharing ideas from Mahadevan 2007. A spectral partitioning approach is also adopted to identify *geometric bottlenecks* of the environment, avoiding to take the reward structure into consideration. Policies and reward functions associated with these partitions are obtained by an intricate *blending* procedure across scales. The diffusion framework can be explained in terms of the random walk process on a graph and the spectrum of the corresponding normalized graph Laplacian. To that account, the algorithm presented in this thesis shares much in common.

Instead of the general graph partitioning paradigm considered so far, Kazemitabar and Beigy 2009 attempts to find strongly connected components (SCC). While similar in spirit, this approach raises some questions regarding ergodicity. It seems that this scheme could be highly sensible to the sampling strategy, resulting in many cases with only a single component being found. The author's viewpoint is that the SCC constraint might be too strong. Surprisingly, the HEXQ framework of Hengst 2002 is not mentioned by Kazemitabar and Beigy 2009. HEXQ was an early attempt to automatically discover hierarchical structure based on a SCC decomposition of individual state variable. The author gave an interpretation of the resulting decomposition in terms of maximum *predictability* within those regions. Such a viewpoint could also admit an information theoretic interpretation which is hinted to the reader later in this thesis.

1.2 States Classification Approaches

Rather than explicitly partitioning the state space using graph theoretic tools, another body of work focused on directly classifying out bottleneck states based on a visitation frequency principle. This view of bottleneck detection is in a sense closer to the philosophical principles of reinforcement learning. The proposed solutions tended to be cheaply computable *online* and without a complete model of the environment. It seems however that they rely abundantly on heuristics, making it even hard to compare them . On the other hand, the graph partitioning approach appears to be justifiable more easily in theory as argued in chapter 4.

The bottleneck discovery problem was formulated in McGovern and Barto 2001 as a multiple-instance learning problem over bags of feature vectors collected by interacting with the environment. Two sets of bags were obtained from observations collected along successful and unsuccessful trajectories. The notion of diverse density then served during either exhaustive search or gradient descent to find regions of the feature space with the most positive instances and the least negative ones. The authors acknowledged that this method can be particularly sensitive to noise. The classification constraint imposed by McGovern and Barto 2001 between *good* and *bad* trajectories was lifted in Stolle and Precup 2002. It rather tried to directly extract those states which have been visited frequently and define them as subgoals. The initiation sets of the options was found by interpolation over the states which appear frequently enough (above average) on the paths to the subgoals.

Şimşek and Barto 2004 identified subgoals by looking for *access states*, leading to the regions of the state space that have not been visited recently: a notion which they called *relative novelty*. The time frame within which novel events can happen is parameter of the algorithm and has a direct influence on which subgoals are detected.

The use of a standard network centrality measure called *betweenness* is investigated in Simsek and Barto 2008. Betweenness centrality measures the fraction of shortest paths passing through a given vertex of the graph. Nodes with high betweenness are deemed more important as they would appear more frequently on short, therefore more likely to be optimal, trajectories to the goal. The reward function is somehow taken into account into this measure by weighting the paths depending on weather they successfully reached the goal or not. A new graph centrality measure is proposed in Rad, Hasler, and Moradi 2010 called *connection graph stability* and is argued to be a better fit than betweenness centrality for discovering bottlenecks. Some of the contributors then exploited slight variations of this definitions under different names such as *connection bridge centrality* or *co-betweenness centrality* (Moradi, Shiri, and Entezari 2010).

1.2.1 Outline

Basic theory of stochastic processes and Markov chains is first presented in chapter 2. The inclusion of this material is motivated by the probabilistic interpretation of spectral graph theory studied in chapter 4. The theory of Markov Decision Processes is presented in section 2.2 as a necessary prerequisite for the appreciation of the techniques developed in reinforcement learning. Chapter 3 focuses on the presentation of the options framework (R. S. Sutton, Precup, and Singh 1999) in reinforcement learning as a way to represent temporal abstraction and learn optimal control over it.

The connection from graph *structure* to system *dynamics* is developed throughout chapter 4 and is instrumental is understanding the strengths and pitfalls of the graph partitioning approach for options discovery. It also allows a better understanding of the relevant work on Nearly-Completely Decomposable Markov Chains (NCD) for future theoretical research on the bottleneck concept. The NCD theory seems to call for an information theoretic comprehension of temporal abstraction which is briefly developed at the end of this section.

A new algorithm for options discovery is proposed in chapter 5 based on the

WALKTRAP community detection algorithm of Pons and Latapy 2005. Although WALKTRAP finds its roots into spectral graph theory, its running time is only order $\mathcal{O}(mn^2)$ rather than $\mathcal{O}(n^3)$ by avoiding to compute the eigenvectors explicitly. The problem of options discovery and construction is also set under the assumption of a continuous state space. Techniques for constructing proximity graphs in Euclidean space are developed in section 5.1. Section 5.3.1 shows how approximate nearest neighbors algorithms can be used to properly define the initiation and termination components of options under continuous observations.

An illustration of the proposed algorithm is provided in chapter 6 with the Pinball domain of Konidaris and Barto 2009. Practical difficulties having to do oscillations and off-policy learning are analysed. The proper empirical choices for the number of nearest neighbors, type of proximity graph and time scale for the WALKTRAP algorithm are discussed.

2

Sequential Decision Making under Uncertainty

2.1 Markov Chains

A discrete-time stochastic process is a family of random variables (r.v.'s) $\{X(t), t \in T\}$ indexed by a time parameter $t \in \mathbb{N} = \{0, 1, 2, ...\}$. The value of X_n is referred to as the **state** of the process. When the family of random variables is defined over a discrete sample space Ω, the stochastic process is said to be *discrete-valued*. Stochastic processes are useful in modelling probabilistic phenomena evolving in time such the price of a stock, or the number of connections to a web service over the day for example. Due to the complexity of many systems in real life, it is desirable to assume certain properties to make the modelling exercise more tractable. One could, for instance, decide to treat the r.v.'s as being *independent and identically distributed* (iid), but only at the cost of loosing the ability to capture correlation among them. The so-called *Markov* assumption is often used when the iid property is deemed too restrictive.

Definition 2.1.1. A **Markov Chain** is a discrete-time and discrete-valued stochastic process which possesses the Markov property. The Markov property implies that for all times $n \geq 0$ and all states $i_0, \ldots, i, j \in \Omega$:

$$P(X_{n+1} = j | X_0 = i_0, X_1 = i_1, ..., X_n = i) = P(X_{n+1} = j | X_n = i) \tag{2.1}$$

The Markov property (or **memoryless property**) could be stated simply by saying that the *future is independent of the past, given the present state.* Therefore, knowing the current value of $X_n = i$ is enough to characterize the future evolution of the stochastic process $\{X_{n+1}, X_{n+2}, \dots\}$ and the history $\{X_0, X_1, \dots, X_{n-1}\}$ can be discarded.

A Markov chain is completely specified by the one-step transition probabilities $p_{ij} = P(X_{n+1} = i | X_n = j)$ contained in its Markov or **stochastic matrix**. For a finite state-space $\mathcal{X}$, we have:

$$\mathbf{P} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p_{22} & \cdots & p_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{bmatrix} \tag{2.2}$$

Furthermore, $\mathbf{P}$ must satisfy the following properties:

$$p_{ij} \geq 0 \text{ and } \sum_{j=1}^{M} p_{ij} = 1 \tag{2.3}$$

Let $\mathbf{P}^\star$ denote the adjoint of $\mathbf{P}$. If $\mathbf{P}^\star = \mathbf{P}$ holds, then the stochastic matrix is said to be **time-reversible**.

Those elements of $\mathbf{P}$ for which $p_{ii} = 1$ are said to be **absorbing** and once entered, the Markov chain can never escape.

If the elements of $\mathbf{P}$ are independent of the time index, the Markov chain is said to be **time-homogeneous** and has **stationary transition probabilities**. If this property is satisfied and one knows the transition probabilities of a Markov Chain, the problem of computing probability distributions is greatly simplified and can be succinctly expressed using matrix multiplication.

The **Chapman-Kolmogorov** equation lets us express the probability of going from state i to j in n steps by $[\mathbf{P}^n]_{ij} = [\underbrace{\mathbf{P} \times \mathbf{P} \times \cdots \times \mathbf{P}}_{\text{n times}}]_{ij}$. The probability of transitioning from state i to any other state under n steps then becomes:

$$P(\cdot, n|i) = e_i \mathbf{P}^n \tag{2.4}$$

Here e_i stands for the vector with zero components everywhere except of for its ith component set to 1.

Using the Chapman-Kolmogorov equation, a state j is classified as being **accessible** from i if $[\mathbf{P}^n]_{i,j} > 0$ for some $n \geq 0$. When the relation holds in both direction, i and j are said to **communicate**. If every possible pairs of states can communicate, the Markov chain is classified as being **irreducible**. Given that a state j has been initially encountered once, it is said to be **recurrent** if the probability of visiting it again is nonzero. That is, denote T_j the time at which the chain returns to j, the recurrence property expresses the fact that $P(T_j < \infty | X_0 = j) > 0$. When $E(T_j|X_0) < \infty$, state j is said to be **positive recurrent**.

These last definitions are essential for defining the **ergodicity** property: a condition often assumed in the analysis of Markov Decision Processes.

Definition 2.1.2. A Markov chain is **ergodic** if it is irreducible and positive recurrent.

It often occurs that one is interested in knowing how the probability distribution for X_n would evolve in time. Denote the distribution of X_n by the row vector μ_n and the **initial distribution** by μ_0. The follow relation can be shown to hold

$$\mu_n = \mu_0 \mathbf{P}^n \tag{2.5}$$

by noting that the Markov property implies that

$$\begin{multline}P(X_0 = i_0, X_1 = i_1, X_2 = i_2, \ldots, X_n = i_n) = \\ P(X_0 = i_0)P(X_1 = i_1|X_0 = i_0)P(X_2 = i_2|X_1 = i_1)\ldots P(X_n = i_n|X_{n-1} = i_{n-1}) \tag{2.6}\end{multline}$$

and that our Markov chain is time-homogeneous ($\mathbf{P}$ is fixed).

If it happens that the distribution of μ_n does not change with n, then the Markov chain is said to be **stationary**.

Definition 2.1.3. A Markov chain said to be stationary if it satisfies $P(X_0 = i_0, X_1 = i_1, \dots, X_n = i_n) = P(X_m = i_0, X_{m+1}, \dots, X_{m+n})$ for any $m \in \mathbb{Z}_+$.

Using 2.5, the stationarity problem for a Markov Chain amounts to finding a vector π such that

$$\pi^\intercal \mathbf{P} = \pi^\intercal \tag{2.7}$$

The stationary distribution $\pi^\intercal$ can understood as being the left eigenvector of $\mathbf{P}$ associated with the eigenvalue 1. Furthermore, it must be that $\sum_k \pi_k = 1$ in order for π to be a well-defined probability distribution. When the Markov chain reaches the stationary distribution, it is said to be in its **steady-state** mode.

An important characterization of the properties of stochastic matrices comes from the theory of nonnegative matrices in the form of the Perron-Frobenius theorem. First, define the spectral radius of a square matrix as

$$\rho(A) \overset{\text{def}}{=} \max_i(|\lambda_i|) \tag{2.8}$$

The theorem in its original form by Horn and C. R. Johnson 1986 is about irreducible matrices and amounts to the following.

Theorem 2.1.1 (Perron-Frobenius Theorem)**.** *Let* $\mathbf{A}$ *be an irreducible and nonnegative matrix, then the following claims hold*

1. *The spectral radius* $\rho(\mathbf{A}) > 0$
2. $\rho(\mathbf{A})$ *is an eigenvalue of* $\mathbf{A}$
3. *There exists a positive vector* $\mathbf{x}$ *such that* $\mathbf{Ax} = \rho(\mathbf{A})\mathbf{x}$ *and* $\rho(\mathbf{A})$ *is a simple eigenvalue of* $\mathbf{A}$*.*
4. *The unique eigenvector whose components sum to 1 is called* ***Perron vector****, that is*

$$\mathbf{Ap} = \rho(\mathbf{A})\mathbf{p}, \quad \mathbf{p} > 0 \;\; \textit{and} \;\; \|\mathbf{p}\|_1 = 1 \tag{2.9}$$

The theorem can also be conveniently adapted to Markov chains under the following lemma proven in Montenegro and Tetali 2006.

Lemma 2.1.2. *Let* **P** *be the stochastic matrix associated with a irreducible and reversible Markov chain over a state space* $\mathcal{X}$ *of size* n*.* **P** *must then have a complete spectrum of real eigenvalues of magnitude at most 1 and of the form*

$$1 = \lambda_0 \geq \lambda_1 \lambda \geq \cdots \geq \lambda_{n-1} \geq -1 \tag{2.10}$$

2.2 Markov Decision Processes

The theoretical treatment given about Markov chains has not considered so far the influence of an external *input* or *control.* The case of controlled Markov chains under the framework of Markov Decision Processes is studied in this section. The theory of Markov Decision Processes was instrumental in the development of the field of Reinforcement Learning studied later in this chapter. The following presentation adopts most of the notation from Szepesvári 2010.

Definition 2.2.1. A finite-action discounted **Markov Decision Process** (MDP) is a tuple $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, r, \mathcal{P}, \gamma \rangle$ where $\mathcal{X}$ is a non-empty countable set of states, $\mathcal{A}$ is a finite state of actions, $\mathcal{P}$ is a transition probability kernel giving rise to a distribution $P(\cdot|x, a)$ over $\mathcal{X}$ given any $x \in \mathcal{X}$ and $a \in \mathcal{A}$, r is the reward function $r : \mathcal{X} \times \mathcal{A} \mapsto \mathbb{N}$ and $\gamma \in [0, 1]$ is a discount factor.

A **deterministic policy** π is a mapping $\pi : \mathcal{X} \mapsto \mathcal{A}$. In the theory of MDPs, stochastic policies of the form $\pi : \mathcal{X} \mapsto \Omega(\mathcal{A})$ are also considered, in which case actions are drawn from a conditional probability distribution over states according to $A_t \sim \pi(\cdot|X_t)$ (here A_t and X_t are r.v.'s at time t).

Fixing a policy for an MDP induces a **Markov Reward Process** (MRP) $\langle \mathcal{X}, \mathcal{R}^\pi, \mathcal{P}^\pi, \gamma \rangle$ with the reward function $\mathcal{R}^\pi(x) = r(x, \pi(x))$ and state transition probability kernel

$P(\cdot|x) = P(\cdot|x, \pi(x))$. An MRP can be thought of as a Markov chain augmented with a reward at every state. The concepts of induced Markov chains and Markov Reward Processes are particularly useful in the analysis of MDPs. At the level of the induced Markov chain of an MDP, the existence of a stationary distribution is not necessarily guaranteed.

2.2.1 Value Function

The notion of a value function goes on par with the principle of optimality that underlies the decision theoretic framework of MDPs. It is assumed in this context that a decision maker, or *agent*, is acting in such a way as to optimize some intrinsic notion of *appropriateness* in its behaviour. In the definition 2.2.1 of an MDP, the reward function r captures this idea. Under this setting, an agent tries to optimize a quantity known as the **return**

Definition 2.2.2. The return of an Markov Decision Process is the total discounted sum of the rewards originating from the induced Markov Reward Process.

$$\mathcal{R} = \sum_{t=0}^{\infty} \gamma^t R_{t+1} \tag{2.11}$$

Fixing the value of $\gamma < 1$ leads to a kind of return where the immediate reward is worth exponentially more than the far future. In this case, the resulting MDP is called a **discounted Markov Decision Process**. On the other hand, setting $\gamma = 1$ makes the immediate reward at each step equally important: one then talks of an **undiscounted Markov Decision Process**

Knowing the desirability, or value, of a state at any moment makes the problem of finding an optimal way of behaving much easier.

Definition 2.2.3. The **value function** $V^\pi : \mathcal{X} \mapsto \mathbb{R}$ of a stationary policy π is the conditional expectation of the return (discounted or undiscounted) given a state.

Under the discounted model,

$$V^\pi(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \,\middle|\, X_0 = x\right], x \in \mathcal{X} \tag{2.12}$$

A behaviour is said to be optimal when the value of its policy is also optimal. An optimal value function $V^* : \mathcal{X} \mapsto \mathbb{R}$ is one which obtains the maximum possible expected return for every state.

The **action-value function** of a policy π is closely related to the notion of value function presented above. Instead of being defined only over states, it specifies the expected return of initially being in state x, choosing a first action a and subsequently committing to π.

Definition 2.2.4. The action-value function $Q^\pi : \mathcal{X} \times \mathcal{A} \mapsto \mathbb{R}$ of a stationary policy π is the conditional expectation of the return given an initial state-action pair.

$$Q^\pi(x, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} | X_0 = x, A_0 = a\right],\ x \in \mathcal{X}, a \in \mathcal{A} \tag{2.13}$$

If the action-value function yields the maximum expected return for every state-action pair, it is said to be optimal and is denoted by Q^*. One can go from the action-value function to the value function by noting that for a finite-action MDP

$$V^*(x) = \max_{a \in \mathcal{A}} Q^*(x, a) \tag{2.14}$$

In the more general case of a countable non-empty sets of actions, the max operator should be replaced with the supremum (sup) of $\mathcal{A}$. The optimal action-value function can also be recovered from the value function as follow:

$$Q^*(x, a) = r(x, a) + \gamma \sum_{y \in \mathcal{X}} P(y, a, x) V^*(y),\ x \in \mathcal{X}, a \in \mathcal{A} \tag{2.15}$$

2.2.2 Bellman Equations

Similar to equation 2.15, given an MDP, the so-called **Bellman equations** are expressed under

$$V^\pi(x) = r(x, \pi(x)) + \gamma \sum_{y \in \mathcal{X}} P(y, \pi(x), x) V^\pi(y) \tag{2.16}$$

The recursive formulation of the Bellman equations relates the value of states to that of its possible successor states, weighted by their probability of occurrence. For a finite d-dimensional state space $\mathcal{X}$, V^π and r^π can be thought to be vectors in $\mathbb{R}^d$ with $\mathcal{P}$ being a transition matrix $\mathbf{P} : \mathbb{R}^{d \times d}$. It can be seen that the Bellman equations define a linear system of equations in d unknowns whose unique solution is V^π

$$\mathbf{V}^\pi = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{V}^\pi \tag{2.17}$$

Solving for the left hand side,

$$\mathbf{V}^\pi = (\mathbf{I} - \gamma \mathbf{P}^\pi)^{-1} \mathbf{r}^\pi \tag{2.18}$$

Under the framework of reinforcement learning adopted in this work, $\mathbf{P}$ and $\mathbf{r}^\pi$ are not available *a priori*, making the direct solution of 2.18 impossible to compute. Furthermore, since matrix inversion is generally of order $\mathcal{O}(n^3)$, the computational cost would quickly become impractical for large state spaces. Reinforcement learning algorithms generally solve this problem in an iterative fashion at a very low cost per iteration.

2.2.3 Bellman Operator

The notion of **Bellman operator** subtends the theoretical justification of the online methods of RL for computing the value function of policies.

Definition 2.2.5. The Bellman operator $T^\pi : \mathbb{R}^\mathcal{X} \mapsto \mathbb{R}^\mathcal{X}$ underlying a policy π for an MDP is defined as

$$(T^\pi V) = r(x, \pi(x)) + \gamma \sum_{y \in \mathcal{X}} P(y, \pi(x), x) V(y),\ x\mathcal{X} \tag{2.19}$$

Similarly, one can define the Bellman operator $T^\pi : \mathbb{R}^{\mathcal{X}\times\mathcal{A}} \mapsto \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$ for the action-value function as

$$T^\pi Q(x, a) = r(x, a) + \gamma \sum_{y \in \mathcal{X}} P(y, a, x) V(y),\ x \in \mathcal{X} \tag{2.20}$$

At first blush, it can be easy to miss the difference between the overall form of the Bellman equations 2.16 and that of equation 2.19. One must in fact observe that, even though T^π is defined for a given policy, the V term on the other hand might not correspond to V^π. The Bellman operator happens to be a special type of function called a **contraction mapping**. This characterization allows the **Banach fixed-point theorem** to be applied for proving convergence. Starting with an estimate of the value function for a policy, the iterative application of the Bellman operator will converge in the limit to a unique fixed point corresponding to V^π. The same principle underlies the value and policy iteration algorithms for computing the optimal value function.

Before delving into the details of these algorithms, the **Bellman optimality operator** must be introduced. Just as T^π, $T^\star : \mathbb{R}^\mathcal{X} \times \mathbb{R}^\mathcal{X}$ is a maximum-norm contraction mapping but is defined this time as

$$(T^\star V) = \max_{a \in \mathcal{A}} \left\{ r(x, a) + \gamma \sum_{y \mathcal{X}} P(y, \pi(x), x) V^\star(y) \right\},\ x \in \mathcal{X} \tag{2.21}$$

The optimal value function $V^\star$ satisfies the fixed-point equation $T^\star V^\star = V^\star$.

2.2.4 Solving MDPs

The Banach fixed-point theorem mentioned in the previous section lays the foundations of two important approaches for solving MDPs: the value and policy iteration

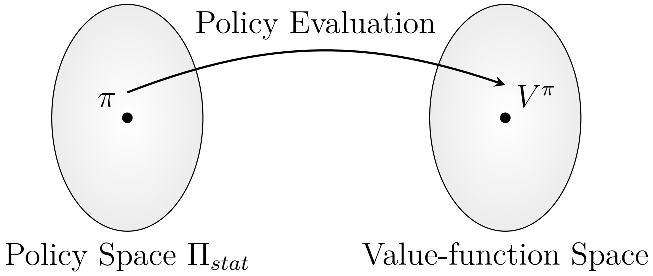


Figure 2.1: The policy evaluation problem consists in finding the value function corresponding to a given policy π. The focus here is on the space of stationary policies Π_{stat}

algorithms. By *solving and MDP*, one generally refers to finding the optimal value function underlying an MDP. If the value function found in that way is indeed optimal, an optimal policy can be derived by greedily picking the best action from it.

The two main problems of RL which consist in finding the value of a policy or finding an optimal policy are usually called the **prediction** (figure 2.1) and **control** (figure 2.2) problems. In the control problem, one tries to derive an optimal policy by taking the greedy policy with respect to the optimal value function (see theorem 2.2 of Ross 1983 for a proof). The optimal stationary greedy policy maximizes the right side of

$$\pi^\star(x) = \arg\max_{a \in \mathcal{A}} \left[r(x,a) + \gamma \sum_{y \in \mathcal{X}} P(y,a,x) V^*(y) \right], \; x \in \mathcal{X}, a \in \mathcal{A} \tag{2.22}$$

Finding V^* thus allows the control problem to be solved using equation 2.22. The **value-iteration** algorithm is one way in which $V^\star$ can be obtained. Let V_0 be some arbitrary initial bounded function, this method consists in applying the Bellman optimality operator $T^\star$ successively in the following manner

$$V_{k+1} = T^\star V_k \tag{2.23}$$

V_k can be shown (proposition 3.1 of Ross 1983) to uniformly converge to V^* as $k \to \infty$

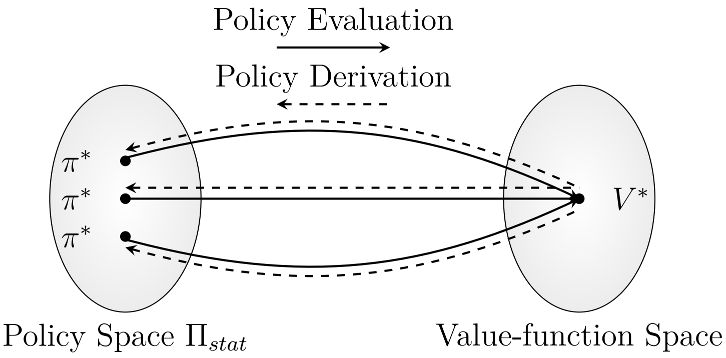


Figure 2.2: The control problem aims at finding an optimal policy i.e. one for which the corresponding value function is optimal. There might be multiple optimal policies, but the optimal value function must be unique (Ross 1983).

With **policy iteration**, two steps are interleaved: policy evaluation and policy improvement. The general idea goes as follow: from an initial policy π_0 compute its corresponding value function (policy evaluation) and derive the greedy policy π_{k+1} from it (policy improvement), then repeat these two steps as necessary. It can be shown that the policy computed by policy iteration after k steps is not worse than the greedy policy computed by value iteration. Because of the policy evaluation step, policy-iteration is computationally more expensive. It plays however an important role in the Actor Critic architectures (R. Sutton 1984).

The policy and value iterations algorithms belong to a class known as **dynamic programming** (DP) methods. Assuming a perfect knowledge of the transitions dynamics and reward function, the computational burden is greatly reduced compared to a naive direct policy search approach which could be order $|\mathcal{X}|^{|\mathcal{A}|}$. Under the framework of reinforcement learning, it will not be possible to maintain these assumptions. DP remains of great importance for the understanding of RL algorithms.

2.3 Reinforcement Learning

Reinforcement Learning (RL) considers the problem setting of a situated **agent** learning to optimize a loss (return) from the direct experience provided by the **environ-**

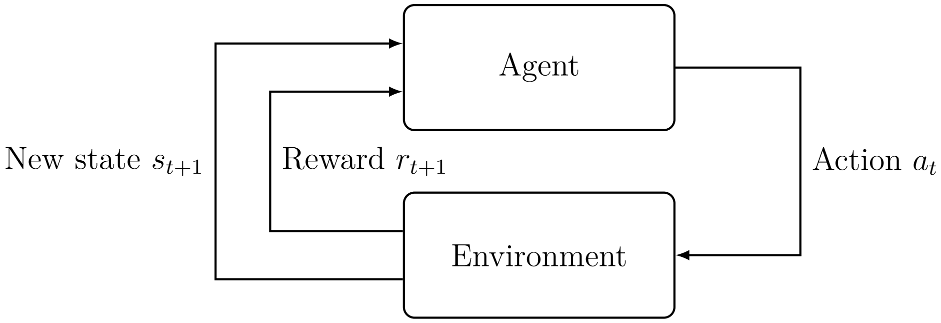


Figure 2.3: Reinforcement Learning. An agent executes an action a_t in the environment at time t producing a state transition and instantaneous reward dictated by the environment dynamics.

ment (figure 2.3). Historically, the field has been influenced by the work on trial-and-error learning from psychology (R. S. Sutton and Barto 1998). RL hinges heavily upon the theory of Markov Decision Processes introduced in the previous section.

The DP assumption of an **environment model** being known is lifted under the RL framework and **model-free** learning becomes the focus. The presence of a model can still be accommodated and subtend a set of RL methods for **planning**, i.e. determining the best course of action for accomplishing a goal by simulating the consequences of actions in the model. Additionally, RL is concerned with the problem of acquiring relevant experience from the environment in an online fashion: a problem of joint **exploration** and control. The way in which the acquired experience relates to the target policy depends on the learning algorithm which is either classified as **on-policy** or **off-policy**.

In the following sections, the Monte-Carlo (MC) method will be presented as a way to learn about the value of states in the absence of a model. Then the Temporal Difference (TD) learning algorithm will be introduced and shown to encompass the MC methods. Finally, the off-policy Q-Learning algorithm will be presented.

2.3.1 Monte-Carlo

The Monte-Carlo approach for solving the control problem estimates the value of a state by taking online samples of the return directly from the environment or with simulated experience using a model. Throughout multiple episodes, independent samples of return are averaged to obtain an estimate of the true expectation. According to the law of large number, as the number of samples goes to infinity, the average becomes an unbiased estimator.

The procedure described above is however not sufficient for solving the control problem as it only answers the prediction one. In the absence of a model, it is necessary to obtain an action-value function for control. Since certain state-action pairs might be difficult to sample frequently enough in large state spaces, it is often assumed that the agent can be reset in some arbitrary state-action configuration: an **exploring start**. In practice this assumption is often impossible to meet. Fortunately, it can be overcome using **soft-policies**.

The Monte-Carlo method for control is based on the general framework of policy iteration where the current policy is improved greedily with respect to some estimate of its value function. It must be noted that policy iteration only requires to update the policy towards the greedy policy. Soft-policies of the form $\pi(x, a) > 0$ consider a slightly perturbed instance of their greedy policy such that the previous condition is not completely violated. It then becomes possible to explore non-greedy actions and the need for exploring starts is eliminated (R. S. Sutton and Barto 1998). A class of soft policies commonly used is the **ϵ-greedy** one, where a random action is chosen with probability ϵ, and the greedy action for $1 - \epsilon$.

The general scheme described so far could be described as an on-policy learning approach, where the value of a policy is simultaneously being evaluated and used for control. If the policy used to obtain samples, the **behavior policy**, is different from the one being evaluated and improved upon, the **estimation policy**, importance sampling techniques must be applied to compensate for the discrepancy in the action selection distributions. Algorithms capable of learning the value of a target policy while following a different behavior policy are said to be off-policy methods.

2.3.2 Temporal Difference Learning

The main drawback of using Monte-Carlo methods in an online setting is the need to wait for the complete execution of an episode before updating the value function estimate. The Temporal Difference (TD) learning algorithm of R. Sutton 1984 showed how to overcome this problem and was instrumental in making the RL approach practical. In TD learning, immediate predictions are used as targets, a technique known as **bootstrapping**, and alleviates the need to wait for a full backup. The TD(λ) is a extension of the original algorithm that unifies DP and Monte-Carlo methods.

Recalling the definition of the value function, for all $x \in \mathcal{X}$

$$\begin{aligned} V^\pi(x) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \,\middle|\, X_0 = x\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, X_t = x\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma V(X_{t+1}) \mid X_t = x\right] \end{aligned} \tag{2.24}$$

The TD(0) algorithm incrementally updates the value function estimate using samples of the form $R_{t+1} + \gamma V(X_{t+1})$ exposed in equation 2.24. The value function estimate is updated by

$$\hat{V}_{t+1}(x) = \hat{V}_t(x) + \alpha_t \left[R_{t+1} + \gamma \hat{V}_t(X_{t+1}) - \hat{V}_t(X_t)\right] \tag{2.25}$$

TD is a Stochastic Approximation (SA) method and can be shown to converge (Szepesvári 2010) to the true V^π by treating the sequence $\hat{V}_t$ as a linear ordinary differential equation (ODE). The sequence of step sizes α must also be subject to the Robbins-Monro (RM) conditions according to which

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \qquad \sum_{t=0}^{\infty} \alpha_t^2 < +\infty \tag{2.26}$$

Algorithm 1 shows how the TD update is performed procedurally.

```
Data: An arbitrary initialized vector $V$, a policy $\pi$ to be evaluated
Result: The value of $\pi$
$x \leftarrow$ initial state
while $x$ is not terminal do
    $a \leftarrow \pi(x)$
    $r, x' \leftarrow \texttt{Step}(a)$
    $V(x) \leftarrow V(x) + \alpha\left[r + \gamma V(x') - V(x)\right]$
    $x \leftarrow x'$
end
```

Algorithm 1: Tabular TD(0) algorithm for policy evaluation. The *Step* function performs the state transition in the environment and returns the immediate reward and new state.

2.3.3 Eligibility Traces

An important extension to the original TD algorithm is the TD(λ) family (R. Sutton 1984), unifying TD(0) at one end and Monte-Carlo prediction at the other. **Eligibility traces** act as kind of memory modulating the propagation of backups at a given state. Instead of updating the value function based on a single n-steps estimate, TD(λ) computes an average, known as the λ-return, over a range of multi-step predictions of the return. The multi-step discounted return is defined as

$$\mathcal{R}_{t:k} = \sum_{s=t}^{t+k} \gamma^{s-t} R_{s+1} + \gamma^{k+1} \hat{V}_t(X_{t+k+1}) \tag{2.27}$$

The λ-return is a mixture of multi-step return with weight $(1-\lambda)\lambda^k$ on each term

$$\mathcal{R}_t^{\lambda} = (1-\lambda) \sum_{k \geq 0} \mathcal{R}_{t:k} \tag{2.28}$$

As λ goes to 0, equation 2.28 simplifies to $R_{t+1} + \gamma \hat{V}_t(X_{t+1})$ and amounts to a one-step TD backup of the TD(0) algorithm. On the other hand, as λ goes to 1 samples of the return from time t until the end of the episode are obtained as in the MC method.

TD(λ) is implemented using an additional vector of size $|\mathcal{X}|$ where each component gets incremented by 1 each time the corresponding state is visited. A decay of $\lambda\gamma$ is also applied upon each component for every time step.

2.3.4 Sarsa

The TD algorithm only solves the policy evaluation problem but not control one. Following the general policy iteration paradigm and the MC algorithm for control, Sarsa is an on-policy control algorithm making use of TD for policy evaluation under a soft-policy exploration strategy. Similar to equation 2.25, the action-value function is updated as follow

$$\hat{Q}_{t+1}(x,a) = \hat{Q}_t(x,a) + \alpha\left[R_{t+1} + \gamma\hat{Q}_t(X_{t+1}, A_{t+1}) - \hat{Q}_t(X_t, A_t)\right]\mathbb{I}_{X_t=x, A_t=a} \tag{2.29}$$

```
Data: An arbitrary initialized matrix $Q$
Result: An optimal action-value function for control
foreach episode do
    $x \leftarrow$ initial state
    $a \leftarrow$ Greedy($x$)
    while $x$ is not terminal do
        $r, x' \leftarrow$ Step($a$)
        $a' \leftarrow$ Greedy($x'$)
        $Q(x,a) \leftarrow Q(x,a) + \alpha\left[r + \gamma Q(x',a') - Q(x,a)\right]$
        $x \leftarrow x'$
        $a \leftarrow a'$
    end
end
```

Algorithm 2: The on-policy Sarsa algorithm based on a TD(0) policy evaluation scheme. The *Greedy* function is the soft greedy policy derived from the current estimate of the action-value function. An ϵ-greedy exploration strategy would be commonly used.

As the number of samples for each state-action pair goes to infinity, $\lim_{t\to\infty}\epsilon = 0$, and under the RM conditions, Sarsa is guaranteed to converge to an optimal policy (R. S. Sutton and Barto 1998).

2.3.5 Q-Learning

Since TD learning is an on-policy learning method, the choice of exploration strategy directly impacts the convergence to the estimation policy. Q-Learning (C. Watkins 1989) decouples the exploration and evaluation problems and allows for any behavior policy to be followed while still converging to $Q^\star$. In this case, the updates to the action-value function estimate consist in

$$Q_{t+1}(x,a) = Q_t(x,a) + \alpha_t \left[R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q_t(X_{t+1}, a') - Q_t(X_t, A_t) \right] \mathbb{I}_{X_t = x, A_t = a} \tag{2.30}$$

The corresponding procedural form is given in algorithm 3.

```
Data: An arbitrary initialized matrix $Q$
Result: An optimal action-value function for control
foreach episode do
    $x \leftarrow$ initial state
    while $x$ is not terminal do
        $a \leftarrow \texttt{Greedy}(x)$
        $r, x' \leftarrow \texttt{Step}(a)$
        $Q(x,a) \leftarrow Q(x,a) + \alpha \left[r + \gamma \max_{a' \mathcal{A}} Q(x', a') - Q(x,a)\right]$
        $x \leftarrow x'$
    end
end
```

Algorithm 3: The Q-Learning algorithm under some arbitrary exploration scheme. ϵ-greedy could once again be used for this task.

2.4 State-Abstraction

The algorithms presented so far assumed *tabular* update rules where value or action-value functions were expressed as vectors and matrices respectively. For many problems, the state space might have infinite cardinality or be simply too large to fit in memory. Furthermore, as the number of dimensions increases, the computational cost also increases exponentially. It matters then to seek for a representation of the value function capable of generalizing across possibly unseen states or state-action pairs.

By expressing the value function as a parametrized function, the incremental updates are performed upon the entries of some parameter vector θ. For large state spaces, the number of components in θ would be much smaller than the number of possible states. Value function approximation can be seen as an instance of a supervised learning problem with a training set consisting of states as inputs and samples of the return (one-step TD, or full Monte-Carlo) as targets.

Any existing method from supervised learning could potentially be used for this task: neural networks, or k-nearest neighbors for regression for example (Szepesvári 2010). TD-Gammon of Tesauro 1995 is considered to be one of the great success story or RL and used neural networks for value function approximation. In this thesis, the attention will be mainly drawn upon the so-called *linear methods*. They are simple, but yet expressive, function approximators of the form

$$V_\theta(x) = \theta^\intercal \varphi(x) \tag{2.31}$$

Linear methods represent the value function as a linear combination of **features**. In equation 2.31, $\theta \in \mathbb{R}^d$ and $\varphi : \mathcal{X} \to \mathbb{R}^d$ with its components being defined by a set of **basis functions** $\varphi_i : \mathcal{X} \to \mathbb{R}$. The way in which φ is obtained specifies a feature extraction method which can be non-linear.

Value function approximation most commonly relies on **gradient-descent** methods to derive update rules for θ. Because of the simple form of equation 2.31, taking the gradient of the value function with respect to θ yields

$$\nabla_\theta V_\theta(x) = \varphi(x) \tag{2.32}$$

Gradient-descent methods update the components of θ by small increments in the direction of the gradient, pointing in the direction of the steepest error. A typical error function minimized in supervised learning techniques is the mean-squared error (MSE). Casting the function approximation problem under a setting where the true (yet unknown) underlying value function V^π is a target, and V_θ is some approximation using the parametrized form of equation 2.31, the MSE is

$$MSE(\theta) = \sum_{x \in \mathcal{X}} P(x) \left[V^\pi(x) - V_\theta(x) \right] \tag{2.33}$$

The $P(x)$ term here accounts for probability of observing a given state $x \in \mathcal{X}$ as an input.

TD(λ) with linear function approximation (algorithm 4) can be shown to converge under the usual RM conditions on the α parameter (Tsitsiklis and Van Roy 1997).

```
Data: An arbitrary initialized vector $\theta$, a policy $\pi$ to be evaluated
Result: The value of $\pi$
$x \leftarrow$ initial state
while $x$ is not terminal do
    $a \leftarrow \pi(x)$
    $r, x' \leftarrow \texttt{Step}(a)$
    $\delta \leftarrow r + \gamma\theta^\intercal\varphi[x'] - \theta^\intercal\varphi[x]$
    $\mathbf{z} \leftarrow \varphi[x] + \gamma\lambda\mathbf{z}$
    $\theta \leftarrow \theta + \alpha\delta z$
    $x \leftarrow x'$
end
```

Algorithm 4: TD(λ) with linear function approximation

2.4.1 Basis functions

The choice of proper function space and feature extraction techniques is a challenging problem which can have a great impact on the quality of the approximation. If one has prior knowledge about certain *regularities* of the optimal value function for the problem at hand, relevant features could be specified explicitly. However, when facing the problem of solving a broad class of problems for which little is known about its structure, general set of basis functions must be used. Two feature extraction schemes are presented below, chosen for their wide applicability and ease of use.

2.4.2 Radial Basis Functions

Given a state $x \in \mathcal{X}$, each component of φ is specified by a radial basis function with the property that $\varphi_i(x_i) = \varphi_i(\|x - c_i\|)$, where c_i specifies the *center* of the basis. Gaussian functions are most commonly used as radial basis functions (RBF) and are defined as

$$\varphi_i(x) = \exp\left(\frac{\|x - c_i\|}{2\sigma_i^2}\right) \tag{2.34}$$

The σ_i parameter specifies the *width* of the Gaussian and must be tuned by hand or using some model selection technique together with the layout of the basis functions and their number. Decreasing the width and increasing the number of basis functions would result in a finer approximation but also incur a higher computational cost.

The choice of norm in equation 2.34 is not restricted to the Euclidean distance and other metric could be used. In Sugiyama et al. 2008 for example, the Geodesic distance taken over the graph induced by some MDP attempts to better capture intrinsic geometrical features.

2.4.3 Fourier Basis Functions

The use of Fourier basis functions for value function approximation was introduced in Konidaris, Osentoski, and P. Thomas 2011 but relies on the well established theory of Fourier analysis.

The nth order Fourier expansion of some univariate periodic function f with period T is given by

$$\hat{f}(x) = \frac{a_0}{2} \sum_{k=1}^{n} \left[a_k \cos\left(k\frac{2\pi}{T}x\right) + b_k \sin\left(k\frac{2\pi}{T}x\right)\right] \tag{2.35}$$

with Fourier coefficients a_k and b_k defined as

$$a_k = \frac{2}{T}\int_0^T f(x)\cos\frac{2\pi kx}{T}dx, \text{ and } b_k = \frac{2}{T}\int_0^T f(x)\sin\frac{2\pi kx}{T}dx \tag{2.36}$$

Since $V^\star$ is unknown, the Fourier coefficients cannot be obtained directly from 2.36. They must rather be treated as *weights* and approximated under the linear approximation framework. The Fourier expansion of f results in $2n+1$ terms but Konidaris, Osentoski, and P. Thomas 2011 showed how it can be simplified to only $n+1$ terms if some assumptions are made on the periodicity of the value function. A function f is even if $f(x) = f(-x)$, in which case the sin terms of equation 2.35 can be droped. A similar observation can be made if f is odd $-f(x) = f(-x)$ and the cos terms can be omitted.

Setting the period to $T = 2$, the nth order univariate Fourier basis is defined as:

$$\phi_i(s) = \cos(i\pi x) \;\; \forall i = 0, ..., n \tag{2.37}$$

The multivariate Fourier expansion of some function F with period T up to order n bears a similar form

$$\hat{F}(\mathbf{x}) = \sum_{\mathbf{c}} \left[a_c \cos\left(\frac{2\pi}{T}\mathbf{c}\cdot\mathbf{x}\right) + b_c \sin\left(\frac{2\pi}{T}\mathbf{c}\cdot\mathbf{x}\right)\right] \tag{2.38}$$

The $\mathbf{c}$ term in equation 2.38 is the Cartesian power $\mathcal{X}^d$ of the d-dimensional state space such that $\mathbf{c} = [c_1, \ldots, c_i, \ldots, c_d]$ where $c_i \in [0, \ldots, n]$. The number of basis functions required for the nth order expansion under this scheme is $2(n+1)^d$. Fortunately using the same argument as for the univariate case, only half of the terms must be kept. The ith basis function is then defined as

$$\varphi(x)_i = \cos\left(\pi \mathbf{c}^i \cdot \mathbf{x}\right) \tag{2.39}$$

While Fourier basis severely suffer from the curse of dimensionality, the approach has the merit of being simple and effective empirically. Experiments conducted during this thesis support this claim. Furthermore, the only choice practitioners have to make is the order of the expansion. Reasonable results can usually obtained by using only the few first lower frequencies of the expansion.

3

Temporal Abstraction

It was shown in the previous chapter how the value function over large (possibly infinite) state spaces can be represented compactly using function approximation. It would then appear natural to harness a similar idea for exploiting *temporal regularities*. In the context of planning, reasoning at different abstract time scales can drastically reduce the problem complexity. The importance of temporal abstraction can be overlooked when considering *simple* everyday human tasks such as preparing coffee, going on a camping trip, changing a bicycle tire, etc. A decomposition of these activities at the level of muscle twitches would however quickly expose the impossibility of carrying any form of learning at this level. A form of hierarchical organization must then exist from the micro to the macro-scales. Under the view adopted in this work, the task of preparing coffee would rather consist in a sequence of *coarse* closed-loop policies executed sequentially towards the overall goal of *having coffee in a cup*.

Many authors have tried to formalize this idea under the framework of reinforcement learning: the MAXQ method of T. Dietterich 1998, the Hierarchical Abstract Machines (HAM) of R. Parr and Russell 1997 or Macro-Actions of Hauskrecht et al. 1998 for example. These other approaches will not be covered in this thesis and the options framework of R. S. Sutton, Precup, and Singh 1999 will rather be adopted. For a survey of the early work on HRL, the interested reader is referred to Stolle 2004; R. S. Sutton, Precup, and Singh 1999.

3.1 Options Framework

Definition 3.1.1. An **Option** is a triple $\langle \mathcal{I} \subseteq \mathcal{X},\ \pi : \mathcal{X} \times \mathcal{A} \to [0,1],\ \beta : \mathcal{X} \to [0,1] \rangle$ consisting of an initiation set, a policy π and a termination condition β.

An option $o \in \mathcal{O}$ can be executed in a *call-and-return* fashion only if under the current state $x \in \mathcal{X}$, $x \in \mathcal{I}$. When o is chosen, the agent acts according to the option policy π until β dictates termination, at which point the agent must choose a new option again.

The initiation component is meant to facilitate the decision procedure by reducing the number of possible options to consider at a given state. Its presence could also had been motivated by historical considerations to accommodate STRIPS-style planning (Fikes, Hart, and Nilsson 1972) more easily. The definition of the initiation set is sometimes neglected by some authors, assuming that all options are available everywhere. The algorithm proposed in this thesis automatically curtails admissible states to form a well-defined initiation set for every option. It should be noted finally that primitive actions can be seen as a special kind of options which are available everywhere, have a policy which always chooses the same action, and last exactly one step.

The options framework is at the crossroad of Markov Decision Processes and semi-Markov Decision Processes of Bradtke and Duff 1994. It considers a base MDP overlaid with variable length courses of action represented as options. It is shown in R. S. Sutton, Precup, and Singh 1999 (theorem 1) how an MDP and a pre-defined set of options form a semi-Markov Decision Process. Most of the theory on SMDP can thus be reused for options. As opposed to the usual theory of SMDP that treats extended actions as indivisible and *opaque* decision units, the options framework allows to look at the structure *within*. Furthermore, options can be either Markov or semi-Markov. In the fictitious coffee domain, waiting a predefined amount of time for the coffee to percolate would most likely be a semi-markov option unless the state representation is changed to compensate for the non-markovianity.

3.2 Bellman Equations for Options

Let $\mu : \mathcal{X} \times \mathcal{O} \to [0, 1]$ be a general (Markov or semi-Markov) policy where $\mathcal{O}$ is any set of options. The value of μ is defined as

$$V^{\mu}(x) = \mathbb{E}\left[R_{t+1} + \cdots + \gamma^{k-1} R_{t+k} + \gamma^k V^{\mu}(X_{t+k}) \;\middle|\; \varepsilon(x, \mu, t)\right] \tag{3.1}$$

where k is a random variable denoting the duration of the first option selected by μ and $\varepsilon(x, \mu, t)$ is the event that the policy over options μ is executing at time t in state x. Similarly, an **option-value function** for control can be defined as

$$\begin{aligned} Q^{\mu}(x, o) &= \mathbb{E}\left[R_{t+1} + \cdots + \gamma^{k-1} R_{t+k} + \gamma^k V^{\mu}(X_{t+k}) \;\middle|\; \varepsilon(x, o, t)\right] \\ &= \mathbb{E}\left[R_{t+1} + \cdots + \gamma^{k-1} R_{t+k} + \sum_{o'\mathcal{O}_x} \mu(X_{t+1}, o') Q^{\mu}(x, o') \;\middle|\; \varepsilon(x, o, t)\right] \end{aligned} \tag{3.2}$$

$\mathcal{O}_x$ in the equation 3.2 is the subset of options which can be initiated under state x. The optimality principle is carried over to the following set of optimal Bellman equations

$$V^{\star}_{\mathcal{O}}(x) = \max_{o \in \mathcal{O}_x} \mathbb{E}\left[R_{t+1} + \cdots + \gamma^{k-1} R_{t+k} + \gamma^k V^{\star}_{\mathcal{O}_x}(X_{t+k}) \;\middle|\; \varepsilon(x, o, t)\right] \tag{3.3}$$

The **optimal option value function** is in turn

$$\begin{aligned} Q^{\star}_{\mathcal{O}}(x, o) &= \mathbb{E}\left[R_{t+1} + \cdots + \gamma^{k-1} R_{t+k} + \gamma^k V^{\star}_{\mathcal{O}_x}(X_{t+k}) \;\middle|\; \varepsilon(x, o, t)\right] \\ &= \mathbb{E}\left[R_{t+1} + \cdots + \gamma^{k-1} R_{t+k} + \gamma^k \max_{o' \in \mathcal{O}_{x_{t+k}}} Q'_{\mathcal{O}}(o', X_{t+k}) \;\middle|\; \varepsilon(x, o, t)\right] \end{aligned} \tag{3.4}$$

If the optimal option value function is available, the greedy policy is guaranteed to be optimal. A policy over options is optimal if given a set of options, its underlying option value function is the optimal value function, i.e. $V^{\mu^{\star}_{\mathcal{O}}}(x) = V^{\star}_{\mathcal{O}}(x) \;\forall\; x \in \mathcal{X}$.

3.3 Learning Behavior Policies

Assuming that an options set $\mathcal{O}$ is specified, existing theory of semi-Markov Decision Processes in continuous time can be applied to solve the control problem (M. L. Puterman 1994; Bradtke and Duff 1994). Under the SMDP model, decision are taken at certain *epochs* spaced by random time intervals during which the system dynamics can be arbitrary. The optimal control problem can be solved by an extension of Q-Learning to the SMDP case.

Definition 3.3.1. SMDP Q-Learning

$$Q_{t+1}(x,o) \leftarrow Q_t(x,o) + \alpha \left[\mathcal{R} + \gamma^k \max_{o' \in \mathcal{O}'_x} Q_t(x',o') - Q_t(x,o)\right] \tag{3.5}$$

The semantics of equation 3.5 holds the under the event of $\varepsilon(x,o,t)$ where option o is executed for k time steps with cumulative discounted reward $\mathcal{R}$. Convergence results are provided in R. E. Parr 1998.

An identical algorithmic construct called Macro Q-Learning is also often encountered in the literature. It stems from the work on macro-actions by Mcgovern, R. S. Sutton, and Fagg 1997 in an attempt to extend the deterministic macro-operators of Korf 1985 to closed-loop policies. As opposed to the more general SMDP Q-Learning algorithm of equation 3.5, Macro Q-Learning treats macro-actions and primitive actions separately. The SMDP update rule is applied for macro-actions while primitive actions are updated using 2.30. Clearly, SMDP Q-learning encompasses Macro Q-Learning and can be expressed identically as

$$Q_{t+1}(x,o) \leftarrow Q_t(x,o) + \alpha \left[\mathcal{R} + \gamma^k \max_{o' \in \mathcal{O}'_x} Q_t(x',o') - Q_t(x,o)\right] \tag{3.6}$$

but where $\mathcal{O}$ necessarily contains all of the primitive actions wrapped as options. Macro Q-Learning is thus equivalent to SMDP Q-Learning where the options set $\mathcal{O}$ is augmented with single-step options for each primitive action.

3.3.1 Intra-Option Learning

In order for the SMDP Q-Learning algorithm in equation 3.5 to provide a good estimate of the optimal option value function, sufficient experience must be obtained about every option. While executing an option to completion, fragments of experience of experience relevant for the current option could potentially be used to learn about other options. Since it is possible to look *within* an option, such an idea can be realized and leads to a sample efficient algorithm.

Intra-Option Learning is an off-policy algorithm which leverages the valuable experience taking place *within* options. While an option is executing, it can simultaneously updates the value function estimate of all the consistent options which would have had taken the same action under a given state. It is thus required that the options be defined using deterministic policies so that this idea of consistency can be established.

Definition 3.3.2. Intra-Option Value Learning

$$U(x, o) = (1 - \beta(x))Q_t(x, o) + \beta(x) \max_{o' \in \mathcal{O}} Q_t(x, o') \tag{3.7}$$

$$Q_{t+1}(x, o) = Q_t(x, o) + \alpha \left[R_{t+1} + \gamma U(x, o) - Q_t(x, o)\right] \tag{3.8}$$

The update rule takes place under the event $\varepsilon(x, o, t)$ after each primitive transition and is applied over every other consistent option for which $\pi(X_t) = A_t$.

Intra-Option Learning being an off-policy method, it is susceptible to suffer from instabilities and divergence issues just as Q-Learning with function approximation. GQ(λ) was introduced in Maei and R. S. Sutton 2010 as an extension of Q-Learning but with special treatment against the aforementioned problems. Although the theory for GQ(λ) is provably correct, sufficient empirical understanding is still lacking. Importance sampling corrections similar as those used in Precup, R. S. Sutton, and Dasgupta 2001 for off-policy TD(λ) could be used as a substitute to GQ(λ).

Finally, in order to learn the policy π for an option, it is customary to adopt the notion of **subgoals**. Subgoal options can be seen as optimizing some intrinsic notion of reward consistent with the overall task objective. A **terminal subgoal value** assigns a value for reaching the terminal states of an option determined by β. A subgoal value function thus arises from the combination of the original underlying MDP plus the overlaid discounted subgoal value function. This concept is often referred to as a **pseudo-reward** function: a term coined in T. G. Dietterich 1999 about the MAXQ framework.

4

From Structure to Dynamics

Graph theory has deep ramifications in many fields. The field of Markov Decision Processes is one such example where taking a graph perspective provides a natural language for describing and understanding many fundamental properties. As shown in this section, spectral graph theory provides powerful tools for studying the dynamical properties of Markov chains. Some of the early applications of this connection can be traced back to the work of Simon and Ando 1961; Donath and Hoffman 1973; Fiedler 1973.

Spectral graph theory will provide the theoretical justification for the heuristic approach to options discovery algorithm proposed in this work. This presentation should be seen as an initial foray towards a better theoretical understanding of the bottleneck concept. Random walk processes on arbitrary graphs G will first be introduced, followed by a characterization of the properties of the graph Laplacian and its definition for MDPs. It will then be shown how graph partitioning can be achieved on the basis of the dynamics induced by the random walk process. Finally, it will be argued that the mixing time property of the MDP-induced graph is relevant to the problem of options discovery and can be related to spectral graph partitioning.

4.1 Random Walk

The notion of random walk is necessary in the presentation of some important spectral properties related to the graph Laplacian in section 4.2. Given a graph G, we can generate a random sequence by picking a vertex uniformly at random in the neighborhood of an initial vertex and carrying on the procedure over to the newly obtained vertex. This procedure corresponds to the realization of a type of random process called *random walk*. More precisely, we define a **simple random walk** on a graph $G = (V, E)$ as a Markov Chain with state space V and for which the transition matrix is given by:

$$P_{ij} = \begin{cases} \frac{w_{ij}}{d(i)} & \text{if } i \sim j \\ \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

w_{ij} above is an edge weight and $d(i)$ is the degree of i.

Using matrix notation, the stochastic matrix of the random walk can be written as

$$\mathbf{P} = \mathbf{D}^{-1}\mathbf{W} \tag{4.2}$$

If each vertex of G has the same degree, in which case G is said to be *d-regular*, the stationary distribution of the random walk is the uniform distribution.

4.2 Graph Laplacian

The combinatorial (unnormalized) graph Laplacian of a weighted graph G is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{4.3}$$

and corresponds to the symmetric matrix

$$L(u,v) = \begin{cases} d_v & \text{if } u = v, \\ -1 & \text{if } u \text{ and } v \text{ are adjacent,} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Where $\mathbf{D} = \operatorname{diag}(d_1, d_2, \ldots, d_n)$ and $\mathbf{W}$ are the diagonal degree and weight (adjacency) matrices for G respectively.

Because of its easier stochastic interpretation, the *normalized* version of the graph Laplacian is often preferred. The interested reader is referred to the highly influential monograph by Chung 1997 which offers an in-depth treatment on the topic.

The normalized graph Laplacian is defined as

$$\mathcal{L} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \tag{4.5}$$

where $\mathbf{L}$ is the unnormalized Laplacian defined in equation 4.3. The elements of the matrix $\mathcal{L}$ then become

$$\mathcal{L}(u,v) = \begin{cases} 1 & \text{if } u = v \text{ and } d_v \neq 0, \\ -\frac{1}{\sqrt{d_u d_v}} & \text{if } u \text{ and } v \text{ are adjacent,} \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

Clearly, $\mathcal{L}$ is also symmetric since

$$\mathcal{L}^\intercal = \left(\mathbf{D}^{-1/2}\right)^\intercal \mathbf{L}^\intercal \left(\mathbf{D}^{-1/2}\right)^\intercal = \mathcal{L} \tag{4.7}$$

The spectral theorem for symmetric matrices (D. S. Watkins 2010) also ensures that $\mathcal{L}$ only has real eigenvalues. Furthermore, eigenvectors corresponding to distinct eigenvalues must be orthogonal.

The normalized Laplacian is closely related to the transition matrix $\mathbf{P}$ of the random walk process presented in section 4.1. It can be seen through the identity

$$\mathbf{I} - \mathcal{L} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \tag{4.8}$$

that

$$\begin{aligned}\mathbf{P} &= \mathbf{D}^{-1/2}\left(\mathbf{I}-\mathcal{L}\right)\mathbf{D}^{1/2} \qquad (4.9)\\ &= \mathbf{D}^{-1/2}\left(\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\right)\mathbf{D}^{1/2}\\ &= \mathbf{D}^{-1}\mathbf{W}\end{aligned}$$

and $\mathbf{P}$ must be similar (in the linear algebra sense) to $\mathbf{I}-\mathcal{L}$. It follows that they also share the same eigenvalues and the right eigenvectors of $\mathbf{I}-\mathcal{L}$ can be obtained from the eigenvectors of $\mathbf{P}$ by multiplying them by $\mathbf{D}^{1/2}$

$$\begin{aligned}P\mathbf{v} &= \lambda\mathbf{v}\\ \mathbf{D}^{-1/2}\left(\mathbf{I}-\mathcal{L}\right)\mathbf{D}^{1/2}\mathbf{v} &= \lambda\mathbf{v}\\ \left(\mathbf{I}-\mathcal{L}\right)\mathbf{D}^{1/2}\mathbf{v} &= \lambda\mathbf{D}^{1/2}\mathbf{v}\\ \left(\mathbf{I}-\mathcal{L}\right)\mathbf{u} &= \lambda\mathbf{u} \qquad (4.10)\\ \mathbf{u} &= \mathbf{D}^{1/2}\mathbf{v} \qquad (4.11)\end{aligned}$$

Because of this connection to random walks, some authors have also defined the **normalized random walk Laplacian** as

$$\mathcal{L}_{rw} = \mathbf{D}^{-1}L = \mathbf{I}-\mathbf{D}^{-1}\mathbf{W} \qquad (4.12)$$

The Laplacian can then be thought of as an **operator** acting upon the space of functions $\mathcal{F}: V \to \mathbb{R}$, with V denoting the vertex set. Expressing such functions as vector $f \in \mathbb{R}^{|V|}$, the normalized Laplacian can be shown to satisfy

$$\begin{aligned}\mathcal{L}f(u) &= f(u) - \sum_{\substack{v\\ u\sim v}} \frac{f(v)}{\sqrt{d_u d_v}}\\ &= \frac{1}{\sqrt{d_u}}\left(\sum_{\substack{v\\ u\sim v}} \frac{f(u)}{\sqrt{d_u}} - \frac{f(v)}{\sqrt{d_v}}\right) \qquad (4.13)\end{aligned}$$

The same could also be said about the normalized Laplacian generalized over weighted graphs, in which case

$$\mathcal{L}f(u) = \frac{1}{\sqrt{d_u}} \left(\sum_{\substack{v \\ u \sim v}} \frac{f(u)}{\sqrt{d_u}} - \frac{f(v)}{\sqrt{d_v}} \right) w_{uv} \tag{4.14}$$

where w_{uv} denotes the edge weight between adjacent vertices u and v.

An $n \times n$ symmetric matrix A is said to be positive semidefinite (PSD) if for any vector $x \in \mathbb{R}^n, x^\intercal A x \geq 0$. The normalized Laplacian admits a quadratic form from which the PSD property becomes clear

$$\begin{aligned}
f^\intercal \mathcal{L} f &= f^\intercal \left(\mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \right) f \\
&= f^\intercal f - f^\intercal \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} f \\
&= \sum_i f_i^2 - \sum_i \frac{f_i}{\sqrt{d_i}} \sum_j w_{ij} \frac{f_j}{\sqrt{d_j}} \\
&= \frac{1}{2} \left(\sum_i f_i^2 - 2 \sum_{i,j} \frac{f_i}{\sqrt{d_i}} \frac{f_j}{\sqrt{d_j}} w_{ij} + \sum_j f_j^2 \right) \\
&= \frac{1}{2} \sum_{ij} \left(f_i^2 - 2 \frac{f_i}{\sqrt{d_i}} \frac{f_j}{\sqrt{d_j}} w_{ij} + f_j^2 \right) \\
&= \frac{1}{2} \sum_{ij} w_{ij} \left(\frac{f_i^2}{d_i} - 2 \frac{f_i}{\sqrt{d_i}} \frac{f_j}{\sqrt{d_j}} + \frac{f_j^2}{d_j} \right) \qquad \text{by defn. } d_i = \sum_j w_{ij} \\
&= \frac{1}{2} \sum_{ij} w_{ij} \left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2
\end{aligned} \tag{4.15}$$

Clearly, $f^\intercal \mathcal{L} f \geq 0$, and $\mathcal{L}$ must be positive semidefinite. Under the definition of the combinatorial Laplacian, a similar quadratic form is induced (Mohar 1991)

$$f^\intercal L f = \sum_{ij} w_{ij} (f_i - f_j)^2 \tag{4.16}$$

The quadratic form can be interpreted as a measure of smoothness of a function over the graph. Intuitively, if f is smooth over each pair of vertices, $f^\intercal \mathcal{L} f$ must be small. On the other hand, if f locally varies is a more drastic manner, this quantity should grow larger. The weighting factor in equation 4.15 also helps to understand how local density might influence this measure of smoothness. In regions of higher density, the proximity or similarity encoded through w_{ij} is higher; hence the greater amplification of the $(f_i - f_j)^2$ term. This property is exploited in semi-supervised learning to encode the so-called **cluster assumption** and force the decision boundary to lie in low density regions Chapelle, Weston, and Schölkopf 2002.

4.3 Laplacian of MDPs

Consider the Markov Reward Process induced by fixing a policy π for an MDP and let $\mathbf{P}^\pi$ denote the corresponding transition probability matrix. As shown earlier, a kind of Laplacian closely related to the normalized Laplacian can be obtained from the transition matrix of a random walk process. Clearly, $\mathbf{P}^\pi$ also defines a stochastic matrix encoding the dynamics induced by π in the environment. The notion of graph Laplacian can naturally be extended for MDPs.

Definition 4.3.1. Given a transition probability matrix $\mathbf{P}^\pi$ underlying an MDP for a fixed policy, the **Laplacian of an MDP** is

$$\mathcal{L}^\pi = \mathbf{I} - \mathbf{P}^\pi \tag{4.17}$$

The Laplacian view of Markov Decision Processes was adopted by a number of authors (Lamond and M. Puterman 1989; M. L. Puterman 1994; Filar 2007) for the study of their chain structure. While the term Laplacian was not explicitly used, their analysis relied on the properties of the so-called *Drazin* inverse (Drazin 1958) of $\mathbf{I} - \mathbf{P}^\pi$. More recently, the work on Proto-Value Functions (PVF) (Mahadevan 2009) also exploited the concept and terminology of the graph Laplacian for the purpose of representation and control.

An obstacle to studying the spectrum of $\mathcal{L}^\pi$ occurs when the induced Markov chain is not reversible. Furthermore, the assumption that $\mathbf{P}^\pi$ is available from an existing MDP is problematic. Under the reinforcement learning framework, the transition matrix is generally unknown and it is necessary to resort to sampling-based methods. In certain scenarios, the control problem is itself very hard to solve. Therefore, requiring the knowledge of $\mathcal{L}^{\pi}$ to facilitate the problem would be nothing other than a chicken-and-egg scenario. In the approach proposed in this thesis, the random walk Laplacian is considered as a practical replacement. Such a substitution is also adopted in Mahadevan 2009 for the same reasons.

4.4 Graph Partitioning

Through the study of the set of eigenvalues associated with the graph Laplacian, some remarkable results on the subgraph structures can be obtained. The set of problems concerning the optimality of graph cuts with regard to the size of their components is commonly referred to as **isoperimetric problems**. Edge cuts are of particular interest because they relate to spectrum of $\mathcal{L}$ through the so-called Cheeger constant.

Let $A, B \subseteq V$ be subsets of the vertices of a graph G and $E(A, B)$ denote the set of edges having one end in A and the other in B. It is also convenient to describe the set of edges having only one end point some $S \subseteq V$ as the **boundary** (Chung and Yau 1994), **edge boundary** (Chung 1997) or **coboundary** (Mohar 1991) of S, that is

$$\partial S = \{\{u, v\} \in E : u \in S \text{ and } v \notin S\} \tag{4.18}$$

The **volume** for S also corresponds to the sum of the degrees of its vertices

$$\mathrm{vol}(S) = \sum_{x \in S} d_x \tag{4.19}$$

The **conductance** of G, also known as the **Cheeger constant** (Chung 1997), is defined as

$$h_G = \min_S h_G(S) \quad \text{where} \quad h_G(S) = \frac{|E(S, \bar{S})|}{\min\left(\mathrm{vol}(S), \mathrm{vol}(\bar{S})\right)} \tag{4.20}$$

The **Cheeger inequality** relates it to the spectrum of $\mathcal{L}$ as follow (Chung 1997)

Lemma 4.4.1.

$$2h_g \geq \lambda_1 > \frac{h_g^2}{2} \tag{4.21}$$

The spectrum of $\mathcal{L}$ can be elegantly exploited for graph partitioning. The problem of graph bipartitioning is first considered before the more general multi-partitioning case. A graph bipartition can be obtained under the Min-Cut formulation and solved in time $\mathcal{O}(n^3)$ using the Ford-Fulkerson algorithm (Ford and Fulkerson 1956). The resulting partitions however tend to be highly imbalanced and in certain cases can degenerate to singleton partitions (Hagen and Kahng 1992). A related problem to consider is then the minimum-width bisection which attempts to find a minimum cut with partitions of equal size. Unfortunately, the problem was shown to be $\mathcal{NP}$-complete by Garey, D. Johnson, and Stockmeyer 1976. A natural response is then to relax the optimization criterion in such a way as to maintain balanced partitions without requiring them to be of same size. The **ratio-cut** criterion is defined by Wei and Cheng 1989; Wei and Cheng 1991; Hagen and Kahng 1992 as

$$RCut(S) = \min_{S} \frac{|E(S, \bar{S})|}{|S| \cdot |S'|} \tag{4.22}$$

and is very much related to the modified Cheeger constant also called the **isoperimetric number** by some author [1]. Once again, finding the minimum ratio cut is also $\mathcal{NP}$- complete but a spectral relaxation based on the combinatorial Laplacian is proposed in Hagen and Kahng 1992.

A similar measure called the **normalized cut** was made popular by Shi and Malik 2000 and rather tries to optimize

$$NCut(S) = \frac{|E(S, \bar{S})|}{\text{vol}(S)} + \frac{|E(S, \bar{S})|}{\text{vol}(\bar{S})} \tag{4.23}$$

[1] Although some authors seems to make no distinction between the terms (Shi and Malik 2000; Levin, Peres, and Wilmer 2008), Chung 1997 shows in chapter 2 how they relate differently to the eigenvalues of $\mathcal{L}$.

Just as the ratio cut, the normalized cut also falls short of an exact solution which is not $\mathcal{NP}$-complete (Shi and Malik 2000). The *normalized* qualification in the name *normalized cut* might seem misleading at first glance when considering the following optimization setting (Luxburg 2007) based on the combinatorial Laplacian.

$$\min_{A} \mathbf{f}^{\intercal} \left(\mathbf{D} - \mathbf{W}\right) \mathbf{f} \tag{4.24}$$

$$\text{subject to } \; \mathbf{Df} \perp \mathbb{1}, \;\; \mathbf{f}^{\intercal}\mathbf{Df} = \text{vol}(V)$$

$$f_i = \begin{cases} \sqrt{\frac{\text{vol}(A)}{\text{vol}(\bar{A})}} & \text{if } v_i \in A \\ -\sqrt{\frac{\text{vol}(\bar{A})}{\text{vol}(A)}} & \text{if } v_i \in \bar{A} \end{cases}$$

where $\mathbf{D} - \mathbf{W}$ can be recognized to be the combinatorial Laplacian as defined in equation 4.3. By relaxing $\mathbf{f}$ to values in $\mathbb{R}$ and setting $g = \mathbf{D}^{1/2}\mathbf{f}$, the problem becomes

$$\min_{g \in \mathbb{R}^{|V|}} \mathbf{g}^{\intercal}\mathbf{D}^{-1/2} \left(\mathbf{D} - \mathbf{W}\right) \mathbf{D}^{-1/2} g \tag{4.25}$$

$$\text{subject to } \; \mathbf{g} \perp D^{1/2}\mathbb{1}, \;\; \|\mathbf{g}\|^2 = \text{vol}(V) \tag{4.26}$$

Under this form, the normalized Laplacian $\mathcal{L}$ finally appears. By the Rayleigh-Ritz theorem, this quantity is minimized when $\mathbf{g}$ equals the second eigenvector of $\mathcal{L}$ (Chung 1997). From the real vector $\mathbf{g}$, the partition membership can be recovered by thresholding its components. The problem of multi-partitioning or *k-way* partitioning can be solved by applying the spectral bipartitioning approach in a top-down recursive manner in each partition.

The normalized cut criterion is not only reported to produce better empirical results, but also provides an intuitive interpretation in terms of random walks. Recall the connection between $\mathbf{I} - \mathcal{L}$ and the normalized random walk Laplacian in equation 4.9 where they were shown to be similar. If λ is an eigenvalue of $\mathcal{L}$, then $1 - \lambda$ must be an eigenvector of $\mathbf{P}$. Maila and Shi 2001 showed how this can be further exploited to express the NCUT criterion in terms of probability distributions of the random walk process. The stationary distribution of a random walk on a connected, non-bipartite graph is equal to (Levin, Peres, and Wilmer 2008)

$$\pi_i = \frac{d_i}{\mathrm{vol}(V)} \tag{4.27}$$

The probability of transitioning from a partition A to B if at the stationary distribution the initial state is in A becomes

$$P(A \to B|A) = \frac{\sum_{x \in A, y \in B} \pi(x) P(x,y)}{\pi(S)} \tag{4.28}$$

which, for a random walks, boils down to

$$P(A \to B|A) = \frac{\sum_{x \in A} d_x}{\mathrm{vol}(A)} \tag{4.29}$$

From equation 4.28 and 4.23, it is then easy to see that

$$NCut(S) = P(S \to \bar{S}|S) + P(\bar{S} \to S|\bar{S}) \tag{4.30}$$

Interestingly, Levin, Peres, and Wilmer 2008 calls this measure in 4.28 the **bottleneck ratio** and defines the **bottleneck ratio for a Markov chain** as

$$\Phi_\star = \min_{S \,:\, \pi(S) \leq \frac{1}{2}} P(S \to \bar{S}|S) \tag{4.31}$$

which corresponds to the Cheeger constant presented in equation 4.20. Such a quantity happened to have been studied intensively as a way to bound the **mixing time** of Markov chains through the notion of **conductance** (a term referring to the same Cheeger constant or the bottleneck ratio presented here). The notion of mixing time can described intuitively as the number of steps necessary for a Markov chain to reach its stationary distribution. It is customary to define the notion of mixing time as the number of steps needed for being *close* to the steady-state distribution (Jerrum and Sinclair 1988).

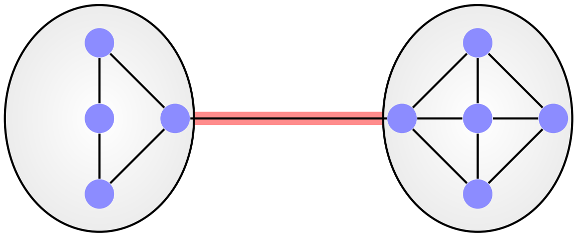


Figure 4.1: Some arbitrary graph presenting a bottleneck. The mixing time can be bounded using the bottleneck ratio of a Markov chain.

The smaller the conductance is, the longer the chain is expected to *mix*. Furthermore, the Markov chain would spend more time within a given partition of the graph and with lower probability, transition to a different adjacent partition (see figure 4.1). It can then be understood that low conductance would also result in lower probabilities of inter-partition transitions.

It has shown above how the second smallest eigenvector of $\mathcal{L}$ minimizes the relaxed NCut criterion and can also bound the Cheeger inequality. It turns out that it can also bound the mixing time through the **spectral gap** $1 - \lambda_1$ The interested reader is referred to Jerrum and Sinclair 1988; Lovász 1996; Levin, Peres, and Wilmer 2008 for some of these results.

4.5 Relevance for Options Discovery

The definition of what makes for *interesting* options has remained vaguely defined throughout the vast body of work on bottleneck-like approaches in reinforcement learning. The intuition developed in this chapter might provide further developments on this question. After having distilled the essence of the bottleneck concept through the graph Laplacian, it is also easier to take an historical perspective and situate it along the work on *Nearly-Completely Decomposable Markov Chains* (NCD).

It was shown in this chapter how the optimal graph bipartitioning problem relates to the Cheeger constant or bottleneck ratio which in turn, provide bounds on the mixing time of Markov chains. It might seem then that bottlenecks allow for the discovery of fast mixing regions of the state space. In the planning problem, fast mixing might also translate into computational speedup, making policies *simpler* to compute in those regions. Efficient random sampling techniques over large state spaces are also often concerned with fast mixing time (Boyd, Diaconis, and Xiao 2004). In reinforcement learning, the same problem would be translated under the question of designing smart exploration strategies. Optimal partitioning of the MDP-induced graph Laplacian could thus help to identifying regions of fast mixing time and create options tailored for exploration.

It also seems that good options are those which are intrinsically *simple*, acting as elementary building building blocks of an increasingly complex hierarchy. Traces of this ideal of simplicity can be found under different forms. For example, it is known from (Maila and Shi 2001), that the stochastic matrix $\mathbf{P}$ has piecewise constant eigenvectors with respect to the NCut criterion. This suggests that optimal partitions can be described more succinctly (simply). The authors also note the potential connection with the notion of **lumpability** introduced by Kemeny and Snell 1976. Lumpability is concerned with the aggregation of states with similar dynamics to simplify large Markov chains.

The idea of matrix decomposition was initiated by Ando and Fisher 1963 from economics for the study of linear dynamical systems with *nearly decomposable* structures. The class of Markov chains with strong intra-cluster interactions and weak inter-cluster ones has been studied under the name of *Nearly Completely Decomposable Markov chains* (NCD) (Gaitsgori and Pervozvanskii 1975). Courtois 1977 also dedicated a book on the topic in the field of queuing theory. The control problem of Markov chains with such structure was considered in Teneketzis, Javid, and Sridhar 1980; Delebecque and Quadrat 1981; Phillips and Kokotovic 1981; Coderch et al. 1983 building upon Courtois' work. This approach treats the *slower* inter-cluster transitions as matrix perturbations. Similar ideas were also brought closer to the framework of Markov Decision Processes with Haviv 1985; Aldhaheri and Khalil 1991; Zhang, Yin, and Boukas 1997.

From an information theoretic point of view, lumpability can be seen as a lossy compression scheme (Watanabe and Abraham 1960; Lindqvist 1978). On the more general topic of graph clustering, the INFOMAP algorithm of Rosvall and Bergstrom 2008 also exploits some compression intuition. By taking random walks in the graph, their algorithms seeks a partitioning of *minimum length*. More recently, Deng, Mehta, and Meyn 2011 gave a characterization of the optimal aggregation of uncontrolled NCD systems in terms of the Kullback-Leibler divergence. They showed that optimal partitioning yields aggregates of maximum *predictability* as measured by the mutual information between $X(t)$ and $X(t+1)$. Similar derivations could be attempted for MDPs by considering the rate distortion for a fixed value function as proposed in Still and Precup 2012 for the problem of exploration.

5

Building Options

The dual perspective offered by spectral graph theory for studying the dynamics of Markov chains is exploited in this chapter for options discovery. As opposed to other similar algorithms (Menache, Mannor, and Shimkin 2002; Mannor et al. 2004; Mathew, Peeyush, and Ravindran 2012; Bouvrie and Maggioni 2012), the problem of discovering and constructing options in continuous state space is tackled. This setting brings along additional challenges in terms of computational efficiency, function approximation and graph representation.

The main distinction between the control techniques of RL and those of the more traditional stochastic dynamic programming approach is that the latter assumes some knowledge of the transition and reward probabilities. Therefore, the graph Laplacian underlying an MDP cannot be assumed to be available under the RL setting. By sampling a large number of state-action transitions, the dynamics could certainly be recovered in the limit. The curse of dimensionality would however quickly render this approach hopeless.

One way to address this problem is by building a *proximity graph* over sampled states. While such a representation is not faithful to the Markov chain induced by some optimal policy, useful geometrical properties can still be inferred by resorting to the random walk process. This reduction is not without consequence as the precious optimality-preserving transitions of $\mathbf{P}_\pi$ are completely ignored. The random walk graph Laplacian can capture relevant geometrical features of the state space, which in certain environments, also reflect optimal structures of the solution space. Mahadevan 2007 also had recourse to the same strategy but argued for its merits in transfer learning.

5.1 Graph Construction

When dealing with a discrete state space, the task of estimating the state-transition graph is much easier than with its continuous counterpart. It suffices indeed to collect sample trajectories and setting graph edges between any two temporally successive states. In the continuous case, temporal ordering can be enough to reconstruct a single geodesic from a trajectory but merging them to form a graph is problematic. Furthermore, with a countably infinity state space, the probability of encountering exactly the same state twice is infinitely small. In machine learning, a common approach for estimating the low-dimensional manifold assumed to underly a set of points consists in building a **proximity graph**. The same technique is adopted in this work for estimating the random walk Laplacian.

Proximity graphs arise from the general problem of extracting geometrical structure from a set of points in the plane. In machine learning, they are commonly found in non-linear dimensionality reduction techniques (Tenenbaum, Silva, and Langford 2000; Roweis and Saul 2000), clustering (Luxburg 2007) or non-parametric classification (G. T. Toussaint and Berzan 2012). Under the manifold assumption, in a *small enough* region around a point, the topological space is assimilable to the Euclidean space. Edges in a proximity graph then capture the notion of distance in this neighborhood.

5.1.1 Empty Region Graphs

Definition 5.1.1. The circle passing through all three vertices of a triangle is called the **circumcircle of a triangle**.

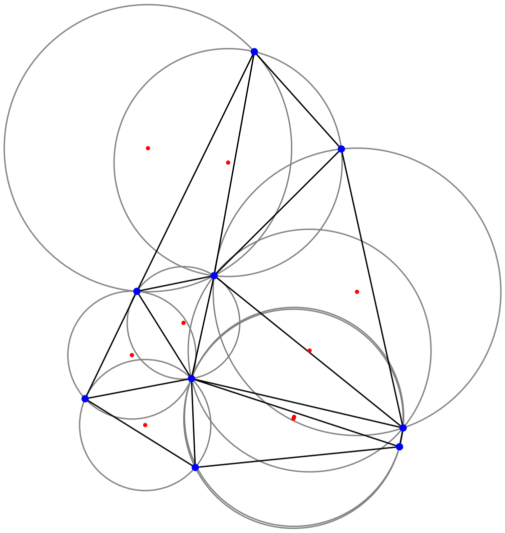
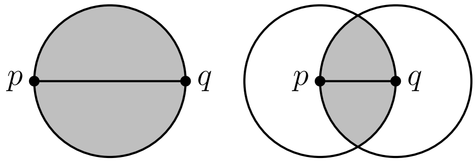


Figure 5.1: The Delaunay triangulation of a set of points in the plane. The red points indicate the circumcenters. It can be seen that no vertex is within the circumcircle of some triangle.

Definition 5.1.2. The **Delaunay Triangulation** (DT) of a set V is the dual of the **Voronoi diagram** decomposing $\mathbb{R}^d$ into $|V|$ *cells*. The Delaunay Triangulation is obtained by setting edges between any two adjacent cells in the Voronoi diagram. The DT also has the property that no point can be found in the circumcircle of any triangle.

From a practical point of view, the Delaunay triangulation offers the advantage of producing connected graphs. However, it only comes at the cost of having a much larger edge set. In $\mathbb{R}^d$ where $d \geq 3$, the number of triangles in the Delaunay graph is known to be $\Omega(n^{\lceil \frac{d}{2} \rceil})$. Furthermore, the worst case runtime complexity for computing it in $d \geq 3$ is $\mathcal{O}(n^{\lceil \frac{d}{2}+1 \rceil})$ using the gift-wrapping algorithm (Fortune 1997).

Definition 5.1.3. Let $B(x, r)$ denote the sphere or radius r centered at x and $\delta(p, q)$ be the distance between two points p and q (figure 5.2). Let the neighborhood of two points be defined by $\Pi_{p,q} = B(\frac{p+q}{2}, \frac{\delta(p,q)}{2})$. The **Grabriel graph** (GG) of a set of vertices V is such that for all edges $(p, q) \in E$ the space within $\Pi_{p,q}$ is empty. That is, $\Pi_{p,q} \cap V = \varnothing$



(a) GG (b) RNG

Figure 5.2: Empty regions of the GG and RNG. The shaded area represents the empty region.

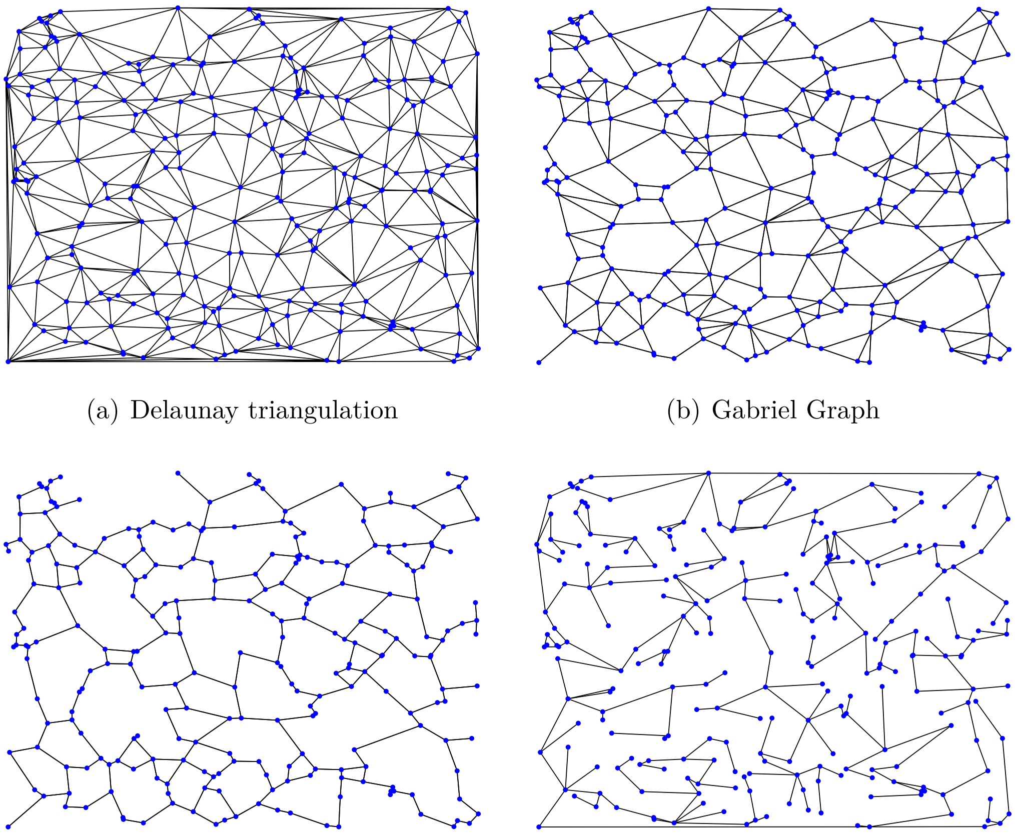
The Gabriel graph was introduced in Gabriel and Sokal 1969 for the analysis of geographic data. In the worst case, the GG of a set of points yields $\Omega(n^2)$ edges (Jaromczyk and G. Toussaint 1992). The expected number of edges of the GG was also studied by Devroye 1988 and shown to be order of $2^{d-1}n$ for most probability densities. In d-dimensional Euclidean space, the trivial brute-force algorithm in $\mathcal{O}(dn^3)$ time complexity is the only one known to date (G. T. Toussaint and Berzan 2012).

Definition 5.1.4. The intersection of the two balls centered at p and q and of radius $\delta(p,q)$ respectively is called a *lune* (figure 5.2); $\Lambda_{p,q} = B(p,\delta(p,q)) \cap B(q,\delta(p,q))$. The **Relative Neighborhood Graph** (RNG) is the graph for which the set of edges E is such that $(p,q) \in E$ if and only if $\Lambda_{p,q} \cap V = \varnothing$

The Relative Neighborhood graph was introduced by G. T. Toussaint 1980 and studied for its ability to capture perceptually meaningful regularities. It was also shown that in the Euclidean plane, the minimum spanning tree (MST) is a subgraph of the RNG which in turns is contained in the DT. In $\mathbb{R}^d$ where $d > 3$, the maximum number of edges in the RNG is $\Omega(n^2)$ (Jaromczyk and G. Toussaint 1992) and the brute-force algorithm for computing it is $\mathcal{O}(n^3)$ (G. T. Toussaint 1980).

Definition 5.1.5. The Euclidean Minimum Spanning Tree (EMST) is the minimum tree that connects every vertices with minimum weights sum. The EMST is a subgraph of the Delaunay triangulation.

The fastest algorithm for computing the EMST was recently obtained by March, Ram, and Gray 2010 and appears to be approximately $\mathcal{O}(n \log n)$ even in $\mathbb{R}^d$.



(a) Delaunay triangulation

(b) Gabriel Graph

(c) RNG

(d) EMST

Figure 5.3: Empty region graphs, ordered according to $EMST(V) \subseteq RNG(V) \subseteq GG(V) \subseteq DT(V)$. A set of 250 points were drawn uniformly at random in $\mathbb{R}^2$

5.1.2 Nearest Neighbor Graphs

While the empty region graphs presented in the last section are geometrically appealing, their density (exponential for the GG) is problematic. Furthermore, the computational cost for obtaining them is prohibitively expensive for the general large scale problems envisioned in this thesis. The class of nearest neighbor graphs tends to be a good replacement against these issues and have a long history in spectral clustering (Luxburg 2007).

Definition 5.1.6. The **Nearest Neighbor Graph** (NNG) is a directed graph connecting vertex p to q if q is the nearest neighbor of p.

The nearest neighbor relation is not symmetric and thus forgo a definition of

NNG as an undirected graph. When admitting k nearest neighbors, the concept can be generalized to the K-NNG (Miller et al. 1997) and the NNG appears to be only a special case where $k = 1$.

Definition 5.1.7. The **Symmetric K-Nearest Neighbor Graph** (K-NNG) is graph in which an edge connects vertex p to q only if q is among the k nearest neighbors of p. Let $N_k(p)$ denote the set of k points closest to p, the edge set is defined as

$$E = \{(p, q) : p \in N_k(q) \textbf{ or } q \in N_k(p)\} \tag{5.1}$$

Definition 5.1.8. The **Mutual K-Nearest Neighbor Graph** (K-NNG) is graph for which an edge connects vertex p and q only if q is among the k nearest neighbors of p and similarly for q in the other direction. That is,

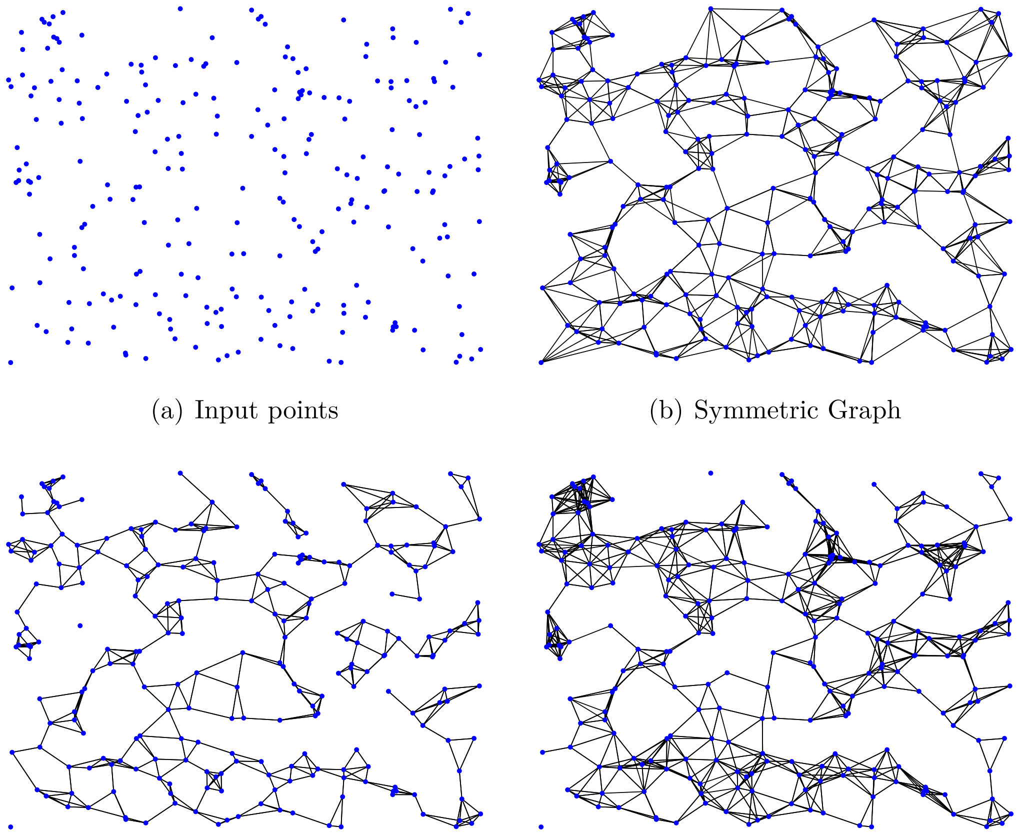
$$E = \{(p, q) : p \in N_k(q) \textbf{ and } q \in N_k(p)\} \tag{5.2}$$

Definition 5.1.9. The **ϵ-Graph** is the graph connecting vertex p and q only if q is within a ball of radius ϵ centered at p. That is,

$$E = \{(p, q) : \delta(p, q) < \epsilon\} \tag{5.3}$$

The choice of an appropriate radius for the ϵ-graph tends to be difficult in practice. Furthermore, if the sampled data exhibit varying densities, a fixed value for ϵ would do poorly and potentially result in a disconnected graph. The mutual graph is usually sparser and can better function over constant densities. It however also leads more easily to disconnected graphs as shown in figure 5.4. Finally, the symmetric graph better handles data at different scales but produces more edges.

K-NN graphs can be implemented effectively using a KD-Tree structure (Friedman, Bentley, and Finkel 1977) in low dimensions but otherwise suffers from the curse of dimensionality. When allowing for approximate neighbors, very efficient algorithms have recently been proposed to solve this problem based on the concept of *locality-*



(a) Input points

(b) Symmetric Graph

(c) Mutual Graph

(d) ϵ-graph

Figure 5.4: K-NN Graphs. The number of nearest neighbors was set to $k = 5$ for the first two examples while ϵ was set to 0.01 for the last.

sensitive hashing (Andoni and Indyk 2008). As opposed to the KD-Tree approach, these algorithms perform efficiently in high dimensional spaces.

5.2 Graph Clustering

Chapter 4 showed how the spectral properties of the graph Laplacian can reveal the clusters structure. In general, computing the eigenvectors of dense matrices is $\mathcal{O}(n^3)$, making the NCut criterion difficult to apply over large state spaces. The community detection algorithm of Newman 2006, the archetypical spectral clustering approach of Ng, Jordan, and Weiss 2001, or the PCCA+ algorithm of Deuflhard and Weber 2005 all exhibit the same running time.

The random walk perspective of graph partitioning is taken in the Walktrap algorithm by Pons and Latapy 2005 to reduce the time complexity. The intuition underlying this algorithm is that of a random walker visiting the graph and spending more time getting *trapped* into densely connected regions and on rare occasions, jump to a different set of vertices.

This probabilistic interpretation of graph partitioning had already been exploited thoroughly in the literature on NCD systems (section 4.5) and highlighted by Maila and Shi 2001, a corner stone for spectral clustering techniques in machine learning. The contribution of Walktrap was to show how a probabilistic distance measure can be defined between vertices and used to obtain clusters using a fast agglomerative method.

Given the random walk transition matrix $\mathbf{P}$, the distance measure between two vertices is defined as

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{P_{ik}^t - P_{jk}^t}{d_k}} = \| D^{-1/2}[\mathbf{P}^t]_i - D^{-1/2}[\mathbf{P}^t]_j \|_2 \tag{5.4}$$

where $\| \cdot \|_2$ is the Euclidean norm and $[\mathbf{P}]_i^t$ is the ith row of matrix $\mathbf{P}$. It appears that the same measure was presented under the name of *diffusion distance* in the same year by Nadler et al. 2005. Both papers also relates it to the spectrum of $\mathbf{P}$ by showing that it amounts to explicitly computing the embedding and taking the Euclidean distance in that space (what Nadler et al. 2005 calls the *diffusion space*).

Intuitively, the diffusion distance measures the L^2 distance between two probability distributions. The vector $[P^t]_i$ contains the probabilities of starting from vertex i and reaching any other vertex j in t steps, taking into account all possible paths between them. For two vertices within a community, their probability vector are expected to be very close as they are both likely to reach the other vertices using the same paths. However, two vertices in different communities *see* very different sets of vertices under t steps and their distance must be thus larger. Nadler et al. 2005 refers to this notion of distance as the *dynamical proximity* since it depends on the dynamics of the random walk over the graph. Varying the number of steps t also has for effect to expose the dynamics at different scales. In dense graphs, fewer steps are required to cover a larger portion of the graph. On the other hand, in sparser graphs larger values for t would not impact as much the locality of the vertices being visited.

The distance measure defined in equation 5.4 yet remains to be incorporated into a clustering scheme. Ward's agglomerative method (Ward 1963) is used in Walktrap to obtain sets of vertices which are as similar as possible with respect to this measure. Initially, the algorithm starts with $|V|$ singleton partitions. At each iteration, communities are merged in a pairwise manner and the distances for the new partitions are updated. The notion of distance between communities is a straightforward extension of equation 5.4 and expresses the probability of going from a given community C to any vertex j in t steps.

$$P^t_{Cj} = \frac{1}{|C|} \sum_{i \in C} P^t_{ij} \tag{5.5}$$

$$r_{C_1,C_2} = \|D^{-1/2} P^t_{C_1 \bullet} - D^{-1/2} P^t_{C_2 \bullet}\| \tag{5.6}$$

The vectors $P^t_{C\bullet}$ are thus probability distributions over the graph expressing the probabilities of choosing uniformly at random a vertex from C and reaching some other vertex j under t steps. What makes Ward's algorithm efficient in this setting is that the distances after merging can be updated efficiently. Each distance computation requires $\mathcal{O}(n)$ and an upper bound is given as a function of the height of the dendogram. The number of distance computations is $\mathcal{O}(mn(H+t))$ (Pons and Latapy 2005 theorem 7). For sparse graphs where the number of edges $m = |V|$ and the height of the dendogram is $\mathcal{O}(\log n)$, the worst case complexity then becomes $\mathcal{O}(n^2 \log n)$. In the general case, the complexity is otherwise $\mathcal{O}(mn^2)$.

5.3 Algorithm

A fast method for discovering and learning subgoal options is presented in algorithm 5. For each option, a terminal subgoal value (pseudo-reward) g is overlaid to the underlying reward function of the base MDP. The type of features or optimal control algorithm is left unspecified. In the experiments presented in the next chapter, Fourier features and Sarsa were used. The Least-Square Policy Iteration (LSPI) (Lagoudakis and R. Parr 2003) algorithm could equally do well and make efficient use of the batch data collected in the first step. In order to make the algorithm capable of handling large state spaces, the Walktrap algorithm is used to discover dynamically stable regions of the state space from the random walk process.

The symmetric K-NN proximity graph construction was retained for its computational efficiency, better adaptation across different scales and sparsity. In the spectral clustering literature the problem of obtaining a similarity graph is sometimes neglected by simply assuming a complete graph with n^2 edges. Such a dense representation however quickly leads to poor computational performance. K-Nearest Neighbor graphs are also used for spectral clustering and their empirical properties are discussed in Luxburg 2007. The family of empty regions graphs consisting of the beta-skeleton graph, GG and RNG was also investigated in Correa and Lindstrom 2012. The use of EMST graphs was ruled out in this algorithm because of its extreme sparsity. It is worth nothing however that Carreira-Perpiñán and Zemel 2004 experimented with ensembles of *perturbed* EMST. The resulting denser combined graph was reported to perform well for clustering.

5.3.1 Initiation and Termination in Continuous Space

When applying algorithm 5 in continuous state spaces, the definition of the initiation and termination components needs to be generalized over regions rather than only discrete states. Having already constructed a nearest neighbor index for the proximity

```
Data: An environment from which to collect experience, terminal subgoal
      reward $R_{subgoal}$, number of random walk steps $t$, number of nearest
      neighbors $k$
Result: A set of options

1. Acquire experience
dataset ← collect samples of experience $\{(x_t, a_t, x_{t+1}, r_t)\}$ through some fixed
policy. A random walk process can be used if the optimal policy is not known.
dataset ← optionally subsample the dataset uniformly at random if too large
index ← build an approximate nearest neighbor index over the sampled states

2. Build the symmetric K-NN graph
vertices ← set of sampled $x_t$ in dataset
edges ← $\varnothing$
foreach state $x$ in dataset do
    knn ← query the $k$ nearest neighbors of $x$
    foreach nearest neighbor nn in knn do
        edges ← edges $\cup$ $(x, nn)$
    end
end

3. Discover and learn options
communities ← Walktrap(Graph(vertices, edges), $t$)
options ← $\varnothing$
foreach community $c$ in communities do
    $\mathcal{I} \leftarrow c$
    subgoals ← $\partial(c)$
    $\beta \leftarrow \mathbb{1}_{\text{subgoals}}$
    g ← $\mathbb{1}_{\text{subgoals}} \cdot R_{subgoal}$
    $\pi$ ← LearnSubgoalMDP(g)
    options ← options $\cup$ Option($\mathcal{I}$, $\beta$, $\pi$)
end
return options
```

Algorithm 5: Bottleneck-Options construction algorithm

graph, an efficient solution is also to use it in the definition of the termination function as

$$\beta(x) = \begin{cases} 1 & \text{if } N_1(x) \in C_x \\ \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

where $N_1(x)$ is the first NN of x and C_x stands for the community to which x belongs to. For better robustness against noise, the majority vote among k nearest neighbors can be taken.

Membership to the initiation set can be determined by a logical negation of β. Let C be the community associated with the initiation set of an option, a membership query consists in

$$N_1(x) \in C \Rightarrow x \in C \tag{5.8}$$

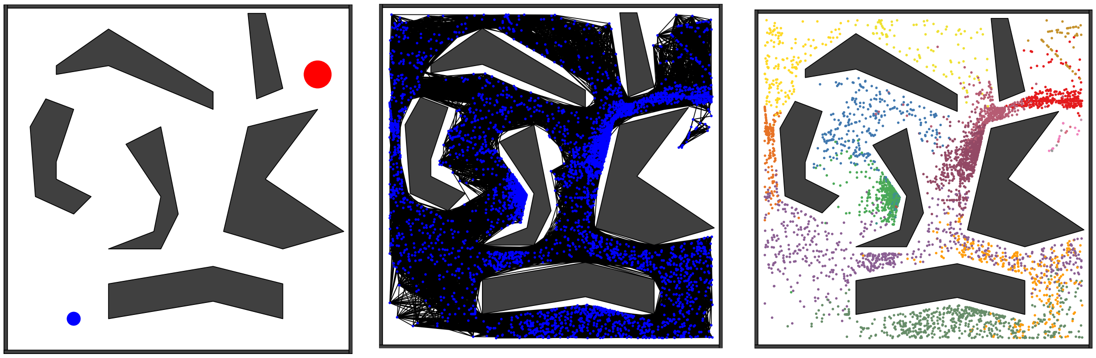
Once again, majority voting can be used to reduce the effect of noise.

6

Empirical Evaluation

The continuous Pinball domain of Konidaris and Barto 2009 was chosen to better understand the practical implications of the options construction algorithm presented in the last chapter. All of the options discovery algorithms surveyed in this work have only shown results for small grid-world domains. The choice of using Pinball thus stems from the desire to show the scalability of the proposed algorithm over significantly larger state spaces. This environment consists in arbitrarily shaped obstacles laid out on the plane and among which an agent must learn to navigate a ball to the target (figure 6.1(a)). The agent has access to four primitive discrete actions which increase or decrease the velocity of the ball in the x and y directions as well as a fifth *null* action. Collisions with the obstacles are elastic and a drag coefficient of 0.995 effectively stops ball movements after a finite number of steps when the null action is chosen repeatedly. Each thrust action incurs a penalty of -5 while taking no action costs -1. The episode terminates with +10000 reward when the agent reaches the target. A four-dimensional continuous observation vector $[x, y, \dot{x}, \dot{y}]$ is available to the agent at every time step and is prone to sharp discontinuities. While the pinball environment as presented originally in Konidaris and Barto 2009 is purely deterministic, normal noise with standard deviation $\sigma = 0.03$ was added exactly as in Tamar, Castro, and Mannor 2013.

For easier comparison with the results presented in Konidaris and Barto 2009, the obstacles layout named `pinball_simple_single.cfg` from the open source imple-



(a) Pinball (b) Graph construction and pruning (c) Graph clustering

Figure 6.1: Pinball domain for reinforcement learning

mentation [1] of the author was used. It has been chosen to implement [2] Pinball in Python rather than using the Java package already made available. One reason for this was to simplify the integration with the Python-RL [3] project from which some components had been used. Great care has however been taken in ensuring that the two implementations behave exactly in the same way.

The main reason for presenting results in only one environment has to do that with the fact that most standard RL domains are designed with small discrete state spaces suitable only for tabular representation of the value function. The machinery used in the proposed algorithm was intended for the continuous case. Even though it would be possible to deploy it over grid world domains, this would make for a poor demonstration of the algorithmic qualities sought for in this design. A second problem pertains to the availability of software implementations for the standard RL domains. The RL-Glue project of Tanner and White 2009 offers some of the RL domains through a socket interface but still lacks sufficient maturity to make it easily usable – to truly use it as a *black box*. Finally, it has to be understood that the idea of machine learning datasets does not translate directly to that of RL environments. When an environment is released to the community, it is not under the form of a

[1] http://www-all.cs.umass.edu/~gdk/pinball/

[2] https://github.com/pierrelux/pypinballl

[3] https://github.com/amarack/python-rl

dataset of sampled transitions but rather as a full-blown simulator which can be acted upon by the learning agent. This has to do with the fact that RL is fundamentally concerned about the learning problem for situated learning agents having no prior knowledge of their environment. It brings about the problem of exploration which requires the ability to actively sample transitions from the environment. A static dataset of transitions could not satisfy this requirement.

6.1 Graph Construction

Instead of collecting samples from a random walk process, a Sarsa(λ) agent was trained with $\alpha = 0.001, \gamma = 0.9, \lambda = 0.9, \epsilon = 0.01$ and 50 trajectories were collected from it. This choice was motivated by the practical need of keeping the number of samples as low as possible and only collect the most relevant ones. The set of trajectories was then merged into one dataset resulting in 48060 sampled states. The dataset was later uniformly subsampled to 5000 data points for easier experimentation.The `FLANN` library of Muja and Lowe 2009 was then used to build an approximate nearest neighbors index for the construction of the symmetric KNN graph. The `igraph` library of Csardi and Nepusz 2006 was used to manipulate the graph more easily.

In practice, it was found that k had a major impact on clustering. Since it is a data-dependent parameter, it must be chosen on a case-by-case basis and cannot be decided in advance. Little is known in the spectral clustering literature about the effect of this parameter and even less about the general problem of choosing a suitable proximity graph construction. This lack of guidance often leads practitioners to simply settle on a k that ensures connectedness. Maier, Luxburg, and Hein 2008 provide some theoretical results about the convergence of graph clustering as a function of k but fail to provide practical guidelines. In the experiment considered here, k was obtained by manually searching for the smallest value producing a connected graph, while minimizing the number of communities found by Walktrap and maximizing the so-called *modularity*. As more communities get merged, the modularity is also expected to increase. Therefore, the optimization is not over conflicting objectives. However, smaller values of k result in sparser graphs which in turn induce a larger number of communities. This brings about a tradeoff between computational resource and succinctness of the options set. It is up to the implementer to decide on this question as a function the available computation resource.

A given graph partitioning among c clusters or *modules* has a high value of modularity when the density of connections within a module is high compared to the inter-module ones. A definition of this measure given by Newman 2006 is

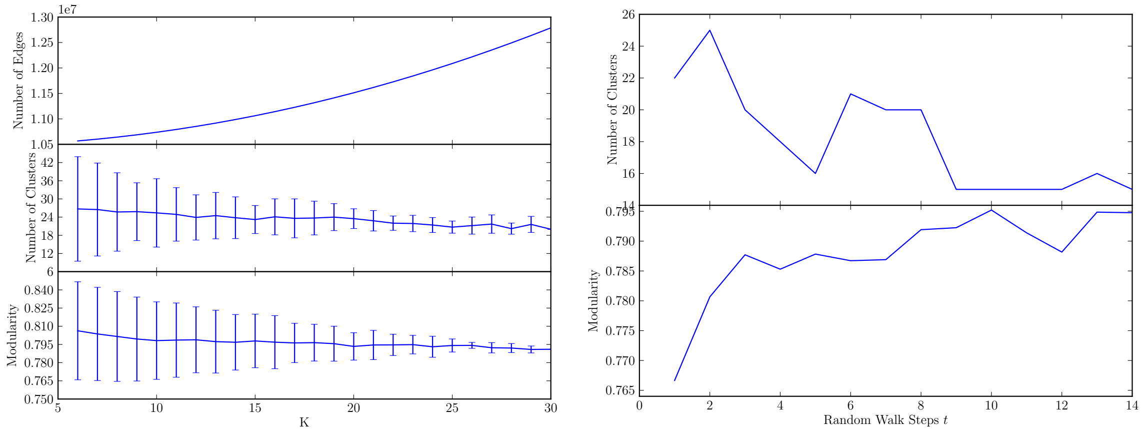
$$Q = \sum_{i}^{c} \left(e_{ii} - a_i^2 \right) \tag{6.1}$$

$$e_{ii} = \sum_j \frac{A_{ij}}{2m} \delta(c_i, c_j) \tag{6.2}$$

$$a_i = \sum_j e_{ij} \tag{6.3}$$

The e_{ii} term here above is the fraction of edges within module i while a_i is of those edges with at least one vertex in i. Walktrap does not directly optimizes the modularity but rather uses the diffusion-like distance as presented in section 5.2. The vertex dendogram produced by Ward's algorithm is however cut at its maximum of modularity.

As K increases, the graph becomes denser and drives the communities to be more compact and less numerous. Walktrap being $\mathcal{O}(mn^2)$, higher values of K also have a direct impact on the running time as the number of edges m gets larger. The presence of error bars in figure 6.2(a) is due to the approximate nearest neighbour graph construction which results in probabilistic edge assignments. As K gets larger, the graph gets denser and the clustering becomes more stable. Intuitively, adding more edges in an already dense graph should have less of an impact on the inter-community transition probabilities than for a very sparse one. A perturbation analysis could shed



(a) From top to bottom: influence of the number of nearest neighbors on the number of edges, number of communities and modularity. Averaged over 10 graphs for each K

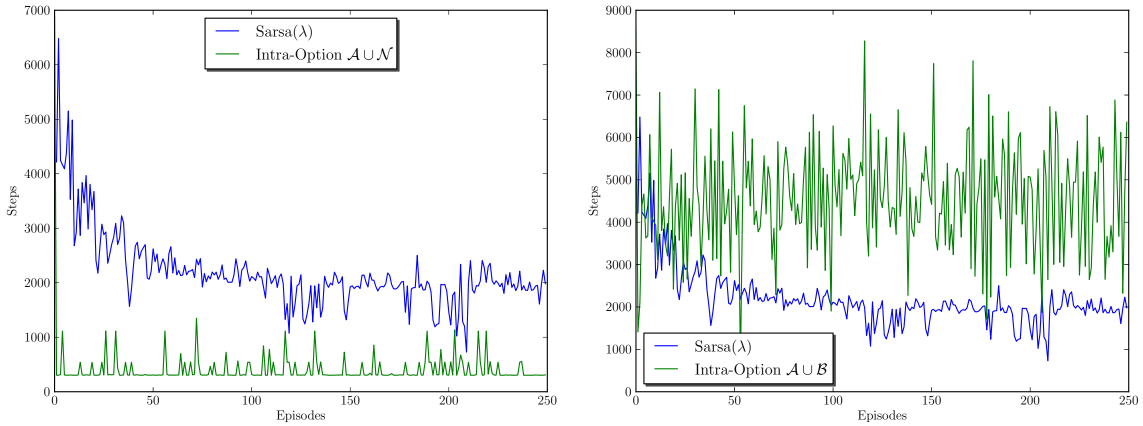
(b) From top to bottom: influence of the number of random walk steps on the number of communities found and associated modularity.

Figure 6.2: Effect of K and t during graph construction and clustering

some light on this phenomenon. Based on the curves shown in figure 5.4, $K = 25$ was retained for this experiment as it yielded good clustering stability and an edge set $|E| = 90348$ of tractable size.

The number of random walk steps in Walktrap affects the scale at which the dynamical proximity is measured. As discussed in section 5.2, the graph density should be considered in this choice. For large values of K, the resulting denser graph will blur away faster the difference between pairs of vertices under longer random walks. Figure 6.2(b) shows this phenomenon starting from $t = 8$ where the number of communities sharply decreases from 26 to 16 as the vertices get dynamically more similar. A smaller number of time steps $t = 4$ was deemed appropriate for this experiment, providing big enough communities, a modularity of 0.7852 and fast computation.

A post-processing step was added in this experiment in order to prune infeasible edges and improve the graph representation. In the Pinball domain, the observation space leads to a graph representation where edges are set between four-dimensional Euclidean points of the form $[x, y, \dot{x}, \dot{y}]$. While it can be difficult to establish the feasibility of a transition between any two such points, it is clear that no edge should



(a) The flat policy take more episodes to achieve the same number of steps as the one over options

(b) Bottleneck options produce uncontrollable oscillatory behavior at the boundary

Figure 6.3: Learning with options

cross the obstacles in the x-y plane. A line intersection test is thus performed over the symmetric graph to remove any such edge (figure 6.1(b)). The options construction algorithm can be dispensed with this step in the absence of such domain specific knowledge. The experience has however shown that it can greatly improve the clustering quality.

6.2 Learning

The bottleneck options algorithm 5 was applied over the clustering for the definition of the initiation, policy and termination components of the options. The bottleneck construction seeks option policies which can bring the agent at the boundary of their respective cluster from any state within the initiation set. It was found however that the *vanilla* bottleneck concept appears to be flawed when it comes to the continuous case. The canonical HRL domain for motivating the bottleneck concept has been mainly concerned with the *doors and rooms* domains in discrete state space. In an environment such as the four-rooms domain (R. S. Sutton, Precup, and Singh 1999), bottlenecks are precisely those single rectangular cells connecting two rooms in the x-y plane.

The situation becomes much more complex for the type of community structures found in Pinball. The problem is twofold: the number of adjacent communities is larger and the bottleneck regions are wider. The difficulty stems mostly from the fact that bottlenecks are now identified in a four-dimensional space rather than only across x-y coordinates. In the four-rooms domain, learning a policy which bring the agent to any of its two doors arbitrarily at random would not impact the rate of convergence as much. In general, the agent should however be constrained to an optimal stochastic choice about which of the bottleneck states should be reached in order to obtain maximum payoff. Furthermore, as the number of adjacent communities gets larger, the number of possible outcomes arising from a transition to any of them might increase and become difficult to evaluate. This intuition was verified empirically by training the bottleneck options using the Sarsa($\lambda = 0.9$) algorithm with learning rate $\alpha = 0.001$, discount factor $\gamma = 0.9$ and an epsilon-greedy exploration strategy with $\epsilon = 0.01$. A policy over such options was then obtained using Intra-Option Learning with $\alpha = 0.001, \gamma = 0.9, \epsilon = 0.01$. A linear fourth-order Fourier approximation of the action-value function was used for every flat or options-based agent. Figure 6.3(b) compares the number of steps taken per episode with that of a flat Sarsa(λ) learning agent. The latter shows steady convergence to some successful policy which can take the ball from its initial position to the goal. No learning however seems possible under the naive bottleneck construction and intra-option learning is very unstable.

6.3 Navigation Options

The need to impose a stochastic choice on which bottlenecks to reach on the boundary leads to slightly different kind of options referred to as *navigation options* in this thesis. This designation highlights the fact that such options specify policies to *navigate* between fixed pairs of adjacent communities.

```
Data: An environment from which to collect experience, terminal subgoal
      reward $R_{subgoal}$, number of random walk steps $t$, number of nearest
      neighbors $k$
Result: A set of options

1. Acquire experience
2. Build the symmetric K-NN graph
3. Discover and learn options
3.1 Establish relevant adjacent communities
communities, membership ← Walktrap(Graph(vertices, edges), $t$)
adjacencies ← $\underbrace{[\varnothing, \ldots, \varnothing]}_{|\text{communities}|}$
foreach community $c$ in communities do
    foreach node $n$ in $c$ do
        knn ← query the $k$ nearest neighbors of $n$
        label, frequency ← Mode(knn)
        if label $\neq$ membership[n] and frequency $\geq \lceil \frac{k}{2} \rceil$ then
            adjacencies[membership[$n$]] ← adjacencies[membership[$n$]] $\cup$ label
        end
    end
end
3.2 Learn options
options ← $\varnothing$
foreach ⟨label, adjLabel⟩ in adjacencies do
    $\mathcal{I}$ ← communities[label]
    subgoals ← communities[adjLabel]
    $\beta$ ← $\mathbb{1}_{\text{subgoals}}$
    g ← $\mathbb{1}_{\text{subgoals}} \cdot R_{subgoal}$
    $\pi$ ← LearnSubgoalMDP(g)
    options ← options $\cup$ Option($\mathcal{I}$, $\beta$, $\pi$)
end
return options
```

Algorithm 6: Navigation-Options construction algorithm

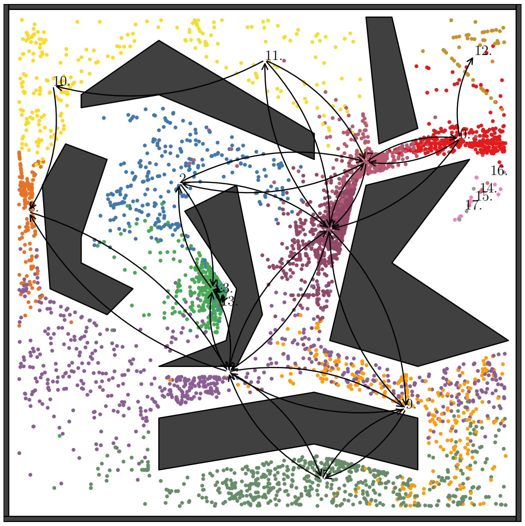


Figure 6.4: Navigation options found using algorithm 6. Each arrow corresponds to an option for transitioning across the given pair of communities. Macro transitions 1 to 7, 5 to 6, 6 to 1, and 7 to 0 were manually selected for learning.

Algorithm 6 is identical to 5 on the sampling and graph construction steps. It however differs in the option discovery and construction approach by defining options *between* communities. The adjacent communities could be discovered simply by considering the bottleneck edges and the label of their adjacent vertices. While this approach would be valid in a perfect graph representation, the identification of relevant bottleneck states is hindered by the high edge density needed for clustering. Under such densities, most vertices of a community are also bottleneck states. It is thus desirable to identify adjacent communities on the basis of a sparser proximity graph. Algorithm 6 for navigation options thus queries the k nearest neighbors and the most frequent label appearing among them is subject to majority test before establishing the adjacency relation. This procedure both reduces the number bottleneck outliers and irrelevant adjacent communities.

The navigation options construction was applied over the 18 communities found by Walktrap at maximum modularity under $t = 4$. It lead to 34 adjacency relations being established, pruning at the same time small communities consisting of only a few vertices. The learning curve in figure 6.4 was obtained by averaging 10 Intra-

Option learning agents with $\alpha = 0.1, \gamma = 0.9, \epsilon = 0.1$ and a fourth order Fourier approximation.

The off-policy Intra-Option Learning is known to diverge under function approximation. Such a problem was encountered during experimentation and motivated the decision to manually select a subset of four options from the 34 found automatically. Such a simplification reduced the learning complexity and made the task of finding suitable learning parameters more easy. The options set used in 6.4 thus consisted of the five primitive actions of pinball augmented with these four navigation options. From the very few first episodes, it can be seen that the agent having access to options successfully learnt to reach the target under approximately 300 steps while the Sarsa agent having access to only primitive actions flattens off above 1000.

One of the main benefit from using options is to speed up learning R. S. Sutton, Precup, and Singh 1999. This experiment clearly demonstrates that the proposed algorithm can produce useful options in that regard. Compared to an agent using no options, it learns much more quickly. Options discovered and learned in the early phases are then effectively exploited by the hierarchical agent to quickly discover a solution. Options in that sense did capture some structure in the solution space. This can be observed in figure 6.4 where options were extracted over the geometrical bottlenecks defined by the obstacles layout. Learning policies to traverse such regions are useful since a stricter control over the actions choice is needed. A wrong choice of action in such a neighborhood could quickly result in an uncontrollable bouncing behavior from which it would be difficult to recover. An agent having access to options for stepping over bottleneck regions can thus explore more safely, making the process of discovering a solution to reach the goal easier.

A comparison with other methods would certainly be useful. However, none of the surveyed method could be readily applied to this problem because of their limited scope to discrete state spaces. Despite the higher time complexity, the betweenness algorithm of Simsek and Barto 2008 would be applicable only if the bottleneck states, defined as states of high betweenness, could somehow be generalized to regions of the continuous state space. Furthermore, it would be necessary to put in place a mechanism for identifying initiation and termination as suggested in this work. The graph clustering approach of Mathew, Peeyush, and Ravindran 2012 or Bouvrie and Maggioni 2012 would also require a similar extension. Future work could include a comparison to the SKILLS CHAINING algorithm of Konidaris and Barto 2009 which was shown to produce useful options in the Pinball domain. This would however go beyond the objectives of this study as SKILLS CHAINING is arguably more of backward chaining approach for planning rather than a bottleneck one.

7

Conclusion

The bottleneck approach to options discovery can be rebuttal because of the lack of proper theoretical foundations. This thesis attempted to motivate this family of heuristics by establishing a connection to spectral graph theory and relating it to the vast body of work on Nearly Decomposable Markov Chains (NCD) spanning across more than three decades. Furthermore, an algorithm based on these principles was proposed for discovering options in continuous state spaces. While many bottleneck-like approaches have succeeded in discrete domains, none of them had yet shown how to extend these ideas in a continuous domain such as Pinball. The empirical evaluation of the proposed algorithm showed its feasibility but also highlighted practical difficulties to use it as a *black-box* approach.

A frequent critique brought against this class of algorithms has to do with the reward structured being apparently ignored. After having studied the properties of the graph Laplacian in section 4, this claim can be dismissed. It is true that algorithm proposed here constructs a graph representation which does not capture the underlying transitions probabilities of some optimal policy. The resulting clustering can then be understood in terms of the dynamics of a random walk process over it; a different stochastic process than the Markov chain induced by some optimal policy. Bottlenecks captured in this way are thus related to those of the random walk Laplacian rather than the true Laplacian of the MDP. If such a substitution does not preserve the reward structure, how can it then succeed in practice ?

A plausible reason might have to do with the kind of domains commonly chosen in HRL. In the four-rooms domains for example, doors appear as natural bottleneck states but also happen to be in structural elements in solution space. The four-rooms domain is so ubiquitous that bottlenecks are often explained in terms of doors. Such references have the unfortunate effect of misguiding practitioners into conceiving bottlenecks as merely *state features*, or *saliencies*, rather than as an effective *coarsening* of the dynamics induced by an MDP.

The empirical results presented in this work depart from those obtained by previous authors in the fact that a bottleneck-like option construction algorithm for continuous state space was proposed and evaluated under a new HRL domain. While subject to the same oversimplification concerning the reward structure, a useful options decomposition was still obtained in the Pinball domain, but also uncovered practical obstacles.

It seems at this point that a proper research methodology should be concerned with not only a more diverse set of domains but also consider the generation of random MDPs with varying degrees of decomposability. The approach of Archibald, McKinnon, and L. C. Thomas 1995 for the generating MDPs seems to be an appropriate fit with the mixing rate being controllable as well as other structural properties of the transition matrix.

The recurrence of the bottleneck concept in the literature should motivate the establishment of a research agenda to equip this notion with proper theoretical foundations. The theory of NCD Markov chains, and more generally that of MDPs from operations research, seems to have been overlooked in in HRL. The problem of options discovery should be first tackled using the theory of MDPs and extended to more of a model-free and online setting compatible with the reinforcement learning framework. A promising research avenue might lie in some information theoretic approach similar to Deng, Mehta, and Meyn 2011 under a rate distortion perspective of the value function as initiated by Still and Precup 2012. Such results would not only benefit our understanding of temporal abstraction in RL but also provide insights to the line of work initiated by Botvinick 2012 on the cognitive process involved in human subgoal discovery.

Bibliography

Aldhaheri, R.W. and H.K. Khalil (1991). “Aggregation of the policy iteration method for nearly completely decomposable Markov chains”. In: *Automatic Control, IEEE Transactions on* 36.2, pp. 178–187.

Ando, Albert and Franklin M. Fisher (1963). “Near-Decomposability, Partition and Aggregation, and the Relevance of Stability Discussions”. In: *International Economic Review* 4.1, pages.

Andoni, Alexandr and Piotr Indyk (Jan. 2008). “Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions”. In: *Commun. ACM* 51.1, pp. 117–122.

Archibald, T. W., K. I. M. McKinnon, and L. C. Thomas (1995). “On the Generation of Markov Decision Processes”. In: *The Journal of the Operational Research Society* 46.3, pages.

Botvinick, Matthew Michael (2012). “Hierarchical reinforcement learning and decision making”. In: *Current Opinion in Neurobiology* 22.6, pp. 956–962.

Bouvrie, Jake V. and Mauro Maggioni (2012). “Multiscale Markov Decision Problems: Compression, Solution, and Transfer Learning”. In: *CoRR* abs/1212.1143.

Boyd, Stephen, Persi Diaconis, and Lin Xiao (Apr. 2004). “Fastest Mixing Markov Chain on a Graph”. In: *SIAM Rev.* 46.4, pp. 667–689.

Bradtke, Steven J. and Michael O. Duff (1994). “Reinforcement Learning Methods for Continuous-Time Markov Decision Problems”. In: *NIPS*. Ed. by Gerald Tesauro, David S. Touretzky, and Todd K. Leen. MIT Press, pp. 393–400.

Carreira-Perpiñán, Miguel Á. and Richard S. Zemel (2004). “Proximity Graphs for Clustering and Manifold Learning”. In: *NIPS*.

Chapelle, Olivier, Jason Weston, and Bernhard Schölkopf (2002). “Cluster Kernels for Semi-Supervised Learning”. In: *NIPS*. Ed. by Suzanna Becker, Sebastian Thrun, and Klaus Obermayer. MIT Press, pp. 585–592.

Chiu, Chung-Cheng and Von-Wun Soo (2010). “AUTOMATIC COMPLEXITY REDUCTION IN REINFORCEMENT LEARNING”. In: *Computational Intelligence* 26.1, pp. 1–25.

Chung, Fan R. K. (1997). *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society.

Chung, Fan R. K. and S.-T. Yau (1994). “A near optimal algorithm for edge separators (preliminary version)”. In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM.

Coderch, M. et al. (Jan. 1983). “Hierarchical Aggregation of Singularly Perturbed Finite State Markov Processes”. In: *Stochastics* 8.4, pp. 1017–1030.

Coifman, Ronald R. and Stephane Lafon (2006). “Diffusion maps”. In: *Applied and Computational Harmonic Analysis* 21.1, pp. 5–30.

Correa, Carlos D. and Peter Lindstrom (2012). “Locally-scaled spectral clustering using empty region graphs”. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’12. Beijing, China: ACM, pp. 1330–1338.

Courtois, Pierre Jacques (1977). *Decomposability : queueing and computer system applications / P. J. Courtois*. Academic Press New York.

Csardi, Gabor and Tamas Nepusz (2006). “The igraph software package for complex network research”. In: *International Journal of Complex Systems*.

Dantzig, George B. and Philip Wolfe (1960). “Decomposition Principle for Linear Programs”. In: *Operations Research* 8.1, pages.

Dean, Thomas and Shieu-Hong Lin (1995). “Decomposition techniques for planning in stochastic domains”. In: *Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2*. IJCAI’95. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., pp. 1121–1127.

Delebecque, François and Jean-Pierre Quadrat (1981). “Optimal control of markov chains admitting strong and weak interactions”. In: *Automatica* 17.2, pp. 281–296.

Deng, Kun, P.G. Mehta, and S.P. Meyn (2011). “Optimal Kullback-Leibler Aggregation via Spectral Theory of Markov Chains”. In: *Automatic Control, IEEE Transactions on* 56.12, pp. 2793–2808.

Deuflhard, Peter and Marcus Weber (2005). “Robust Perron cluster analysis in conformation dynamics”. In: *Linear Algebra and its Applications* 398. Special Issue on Matrices and Mathematical Biology, pp. 161–184.

Devroye, L. (1988). “The expected size of some graphs in computational geometry”. In: *Computers & Mathematics with Applications* 15.1, pp. 53–64.

Dietterich, TG (1998). “The MAXQ method for hierarchical reinforcement learning”. In: *Fifteenth International Conference on Machine Learning*. Morgan Kaufman.

Dietterich, Thomas G. (2000). “An Overview of MAXQ Hierarchical Reinforcement Learning”. In: *Proceedings of the 4th International Symposium on Abstraction, Reformulation, and Approximation*. SARA ’02. London, UK, UK: Springer-Verlag, pp. 26–44.

— (1999). “Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition”. In: *CoRR* cs.LG/9905014.

Donath, W.E. and A.J. Hoffman (1973). “Lower Bounds for the Partitioning of Graphs”. In: *IBM Journal of Research and Development* 17.5, pp. 420–425.

Drazin, M. P. (1958). “Pseudo-Inverses in Associative Rings and Semigroups”. In: *The American Mathematical Monthly* 65.7, pages.

Fiedler, M. (1973). “Algebraic connectivity of graphs”. In: *Czechoslovak Mathematical Journal* 23.98, pp. 298–305.

Fikes, Richard, Peter E. Hart, and Nils J. Nilsson (1972). “Learning and Executing Generalized Robot Plans”. In: *Artif. Intell.* 3.1-3, pp. 251–288.

Filar, Jerzy A. (Jan. 2007). “Controlled Markov chains, graphs, and Hamiltonicity”. In: *Found. Trends. Stoch. Sys.* 1.2, pp. 77–162.

Ford, L. R. and D. R. Fulkerson (1956). “Maximal Flow through a Network.” In: *Canadian Journal of Mathematics* 8, pp. 399–404.

Fortune, Steven (1997). “Handbook of discrete and computational geometry”. In: ed. by Jacob E. Goodman and Joseph O’Rourke. Boca Raton, FL, USA: CRC Press, Inc. Chap. Voronoi diagrams and Delaunay triangulations, pp. 377–388.

Friedman, Jerome H., Jon Louis Bentley, and Raphael Ari Finkel (Sept. 1977). “An Algorithm for Finding Best Matches in Logarithmic Expected Time”. In: *ACM Trans. Math. Softw.* 3.3, pp. 209–226.

Gabriel, K. Ruben and Robert R. Sokal (1969). “A New Statistical Approach to Geographic Variation Analysis”. In: *Systematic Zoology* 18.3, pages.

Gaitsgori, V.G. and A.A. Pervozvanskii (1975). “Aggregation of states in a Markov chain with weak interaction”. English. In: *Cybernetics* 11.3, pp. 441–450.

Garey, M.R., D.S. Johnson, and L. Stockmeyer (1976). “Some simplified NP-complete graph problems”. In: *Theoretical Computer Science* 1.3, pp. 237–267.

Hagen, Lars W. and Andrew B. Kahng (1992). “New spectral methods for ratio cut partitioning and clustering”. In: *IEEE Trans. on CAD of Integrated Circuits and Systems* 11.9, pp. 1074–1085.

Hauskrecht, Milos et al. (1998). “Hierarchical Solution of Markov Decision Processes using Macro-actions”. In: *UAI*. Ed. by Gregory F. Cooper and Serafín Moral. Morgan Kaufmann, pp. 220–229.

Haviv, Moshe (1985). “Block-successive approximation for a discounted Markov decision model”. In: *Stochastic Processes and their Applications* 19.1, pp. 151–160.

Hengst, Bernhard (2002). "Discovering Hierarchy in Reinforcement Learning with HEXQ". In: *Proceedings of the Nineteenth International Conference on Machine Learning*. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 243–250.

Horn, Roger A. and Charles R. Johnson, eds. (1986). *Matrix analysis*. New York, NY, USA: Cambridge University Press.

Jaromczyk, J.W. and G.T. Toussaint (1992). "Relative neighborhood graphs and their relatives". In: *Proceedings of the IEEE* 80.9, pp. 1502–1517.

Jerrum, Mark and Alistair Sinclair (1988). "Conductance and the rapid mixing property for Markov chains: the approximation of permanent resolved". In: *Proceedings of the twentieth annual ACM symposium on Theory of computing*. STOC '88. Chicago, Illinois, USA: ACM, pp. 235–244.

Kazemitabar, Seyed Jalal and Hamid Beigy (2009). "Using Strongly Connected Components as a Basis for Autonomous Skill Acquisition in Reinforcement Learning". In: *Proceedings of the 6th International Symposium on Neural Networks on Advances in Neural Networks*. ISNN '09. Wuhan, China: Springer-Verlag, pp. 794–803.

Kemeny, John G. and Laurie J. Snell (July 1976). *Finite Markov chains*. Second. Springer-Verlag.

Koller, Daphne et al., eds. (2009). *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*. Curran Associates, Inc.

Konidaris, George and Andrew G. Barto (2009). "Skill Discovery in Continuous Reinforcement Learning Domains using Skill Chaining". In: *NIPS*. Ed. by Yoshua Bengio et al. Curran Associates, Inc., pp. 1015–1023.

Konidaris, George, Sarah Osentoski, and Philip Thomas (2011). "Value Function Approximation in Reinforcement Learning using the Fourier Basis". In: *AAAI*, pp. 380–385.

Korf, Richard E. (Apr. 1985). "Macro-operators: a weak method for learning". In: *Artif. Intell.* 26.1, pp. 35–77.

Lagoudakis, Michail G. and Ronald Parr (2003). "Least-Squares Policy Iteration". In: *Journal of Machine Learning Research* 4, pp. 1107–1149.

Lamond, B. and M. Puterman (1989). "Generalized Inverses in Discrete Time Markov Decision Processes". In: *SIAM Journal on Matrix Analysis and Applications* 10.1, pp. 118–134.

Levin, David A., Yuval Peres, and Elizabeth L. Wilmer (2008). *Markov Chains and Mixing Times*. American Mathematical Society.

Lindqvist, Bo (1978). "On the Loss of Information Incurred by Lumping States of a Markov Chain". In: *Scandinavian Journal of Statistics* 5.2, pages.

Lovász, L. (1996). "Random Walks on Graphs: A Survey". In: *Combinatorics, Paul Erdős is Eighty*. Ed. by D. Miklós, V. T. Sós, and T. Szőnyi. Vol. 2. Budapest: János Bolyai Mathematical Society, pp. 353–398.

Luxburg, Ulrike (Dec. 2007). "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4, pp. 395–416.

Maei, Hamid Reza and Richard S. Sutton (2010). "$GQ(\lambda)$: A general gradient algorithm for temporal-difference prediction learning with eligibility traces". In: *Proceedings of the Third Conference on Artificial General Intelligence*.

Mahadevan, Sridhar (2009). "Learning Representation and Control in Markov Decision Processes: New Frontiers". In: *Foundations and Trends in Machine Learning* 1.4, pp. 403–565.

— (2007). "Proto-value Functions : A Laplacian Framework for Learning Representation and Control in Markov Decision Processes". In: *Journal of Machine Learning Research* 8, pp. 2169–2231.

Maier, Markus, Ulrike von Luxburg, and Matthias Hein (2008). “Influence of graph construction on graph-based clustering measures”. In: *NIPS*. Ed. by Daphne Koller et al. Curran Associates, Inc., pp. 1025–1032.

Maila, Marina and Jianbo Shi (2001). “A Random Walks View of Spectral Segmentation”. In: *AI and STATISTICS (AISTATS) 2001*.

Mannor, Shie et al. (2004). “Dynamic abstraction in reinforcement learning via clustering”. In: *Proceedings of the twenty-first international conference on Machine learning*. ICML ’04. New York, NY, USA: ACM, pp. 71–.

March, William B., Parikshit Ram, and Alexander G. Gray (2010). “Fast euclidean minimum spanning tree: algorithm, analysis, and applications”. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’10. New York, NY, USA: ACM, pp. 603–612.

Mathew, Vimal, Kumar Peeyush, and Balaraman Ravindran (2012). “Abstraction in Reinforcement Learning in Terms of Metastability”. In: *EWRL*, pp. 1–14.

McGovern, Amy and Andrew G. Barto (2001). “Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density”. In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., pp. 361–368.

Mcgovern, Amy, Richard S. Sutton, and Andrew H. Fagg (1997). “Roles of Macro-Actions in Accelerating Reinforcement Learning”. In: *In Grace Hopper Celebration of Women in Computing*, pp. 13–18.

Menache, Ishai, Shie Mannor, and Nahum Shimkin (2002). “Q-Cut - Dynamic Discovery of Sub-Goals in Reinforcement Learning”. In: *Proceedings of the 13th European Conference on Machine Learning*. Springer-Verlag, pp. 295–306.

Metzen, Jan Hendrik (2012). “Online Skill Discovery using Graph-based Clustering”. In: *Journal of Machine Learning Research* W&CP 24. Ed. by Marc Peter Deisenroth, Csaba Szepesvari, and Jan Peters, pp. 77–88.

Miller, Gary L. et al. (Jan. 1997). “Separators for sphere-packings and nearest neighbor graphs”. In: *J. ACM* 44.1, pp. 1–29.

Mohar, Bojan (1991). “The Laplacian spectrum of graphs”. In: *Graph Theory, Combinatorics, and Applications*. Wiley, pp. 871–898.

Montenegro, R. and P. Tetali (May 2006). “Mathematical aspects of mixing times in Markov chains”. In: *Found. Trends Theor. Comput. Sci.* 1.3, pp. 237–354.

Moradi, Parham, MohammadEbrahim Shiri, and Negin Entezari (2010). “Automatic Skill Acquisition in Reinforcement Learning Agents Using Connection Bridge Centrality”. In: *Communication and Networking*. Ed. by Tai-hoon Kim et al. Vol. 120. Communications in Computer and Information Science. Springer Berlin Heidelberg, pp. 51–62.

Muja, Marius and David G. Lowe (2009). “Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration”. In: *International Conference on Computer Vision Theory and Application VISSAPP’09)*. INSTICC Press, pp. 331–340.

Nadler, Boaz et al. (2005). “Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators”. In: *NIPS*.

Newman, M. E. J. (July 23, 2006). “Finding community structure in networks using the eigenvectors of matrices”. In: *Physical Review E* 74.3, pp. 036104+.

Ng, Andrew Y., Michael I. Jordan, and Yair Weiss (2001). “On Spectral Clustering: Analysis and an algorithm”. In: *NIPS*. Ed. by Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani. MIT Press, pp. 849–856.

Parr, Ronald Edward (1998). “Hierarchical control and learning for markov decision processes”. PhD thesis.

Parr, Ronald and Stuart J. Russell (1997). “Reinforcement Learning with Hierarchies of Machines”. In: *NIPS*. Ed. by Michael I. Jordan, Michael J. Kearns, and Sara A. Solla. The MIT Press.

Phillips, R. and P.V. Kokotovic (1981). “A singular perturbation approach to modeling and control of Markov chains”. In: *Automatic Control, IEEE Transactions on* 26.5, pp. 1087–1094.

Pons, Pascal and Matthieu Latapy (2005). “Computing Communities in Large Networks Using Random Walks”. In: *Computer and Information Sciences - ISCIS 2005*. Ed. by pInar Yolum et al. Vol. 3733. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 284–293.

Precup, Doina, Richard S. Sutton, and Sanjoy Dasgupta (2001). “Off-Policy Temporal Difference Learning with Function Approximation”. In: *ICML*. Ed. by Carla E. Brodley and Andrea Pohoreckyj Danyluk. Morgan Kaufmann, pp. 417–424.

Puterman, Martin L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. New York, NY, USA: John Wiley & Sons, Inc.

Rad, A.A., Martin Hasler, and P. Moradi (2010). “Automatic skill acquisition in Reinforcement Learning using connection graph stability centrality”. In: *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pp. 697–700.

Ross, Sheldon (1983). *Introduction to stochastic dynamic programming*. Probability and Mathematical Statistics. New York - London etc.: Academic Press.

Rosvall, Martin and Carl T. Bergstrom (2008). “Maps of random walks on complex networks reveal community structure”. In: *Proceedings of the National Academy of Sciences* 105.4, pp. 1118–1123.

Roweis, Sam T. and Lawrence K. Saul (2000). “Nonlinear dimensionality reduction by locally linear embedding”. In: *Science* 290, pp. 2323–2326.

Shi, Jianbo and Jitendra Malik (2000). “Normalized Cuts and Image Segmentation”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.8, pp. 888–905.

Simon, H. A. and A. Ando (1961). “Aggregation of variables in dynamic systems”. In: *Econometrica* 29, pp. 111–138.

Simsek, Özgür and Andrew G. Barto (2008). “Skill Characterization Based on Betweenness”. In: *NIPS*. Ed. by Daphne Koller et al. Curran Associates, Inc., pp. 1497–1504.

Şimşek, Özgür and Andrew G. Barto (2004). “Using relative novelty to identify useful temporal abstractions in reinforcement learning”. In: *Proceedings of the twenty-first international conference on Machine learning*. ICML ’04. Banff, Alberta, Canada: ACM, pp. 95–.

Şimşek, Özgür, Alicia P. Wolfe, and Andrew G. Barto (2005). “Identifying useful subgoals in reinforcement learning by local graph partitioning”. In: *Proceedings of the 22nd international conference on Machine learning*. ICML ’05. Bonn, Germany: ACM, pp. 816–823.

Still, Susanne and Doina Precup (2012). “An information-theoretic approach to curiosity-driven reinforcement learning”. In: *Theory in Biosciences* 131.3, pp. 139–148.

Stolle, Martin (Feb. 2004). “Automated Discovery of Options in Reinforcement Learning”. Master’s thesis. McGill University.

Stolle, Martin and Doina Precup (2002). “Learning Options in Reinforcement Learning”. In: *Proceedings of the 5th International Symposium on Abstraction, Reformulation and Approximation*. London, UK, UK: Springer-Verlag, pp. 212–223.

Sugiyama, Masashi et al. (2008). “Geodesic Gaussian kernels for value function approximation”. In: *Auton. Robots* 25.3, pp. 287–304.

Sutton, Richard (1984). “Temporal credit assignment in reinforcement learning”. PhD thesis. University of Massachusetts Amherst.

Sutton, Richard S. and Andrew G. Barto (1998). *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. Cambridge, Massachusetts: MIT Press.

Sutton, Richard S, Doina Precup, and Satinder Singh (1999). “Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning”. In: *Artificial Intelligence* 112.1-2, pp. 181–211.

Szepesvári, Csaba (2010). *Algorithms for Reinforcement Learning.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Tamar, Aviv, Dotan Di Castro, and Shie Mannor (June 2013). "Temporal Difference Methods for the Variance of the Reward To Go". In: *Proceedings of the 30th International Conference on Machine Learning.*

Tanner, Brian and Adam White (2009). "RL-Glue : Language-Independent Software for Reinforcement-Learning Experiments". In: *Journal of Machine Learning Research* 10, pp. 2133–2136.

Teneketzis, D., S.H. Javid, and B. Sridhar (1980). "Control of weakly-coupled Markov chains". In: *Decision and Control including the Symposium on Adaptive Processes, 1980 19th IEEE Conference on.* Vol. 19, pp. 137–142.

Tenenbaum, Joshua B., Vin de Silva, and John C. Langford (Dec. 22, 2000). "A Global Geometric Framework for Nonlinear Dimensionality Reduction". In: *Science* 290.5500, pp. 2319–2323.

Tesauro, Gerald (Mar. 1995). "Temporal difference learning and TD-Gammon". In: *Commun. ACM* 38.3, pp. 58–68.

Toussaint, Godfried T. (1980). "The relative neighbourhood graph of a finite planar set". In: *Pattern Recognition* 12.4, pp. 261–268.

Toussaint, Godfried T. and Constantin Berzan (2012). "Proximity-graph instance-based learning, support vector machines, and high dimensionality: an empirical comparison". In: *Proceedings of the 8th international conference on Machine Learning and Data Mining in Pattern Recognition.* MLDM'12. Berlin, Germany: Springer-Verlag, pp. 222–236.

Tsitsiklis, J. N. and B. Van Roy (1997). "An analysis of temporal-difference learning with function approximation". In: *Automatic Control, IEEE Transactions on* 42.5, pp. 674–690.

Ward Joe H., Jr. (1963). "Hierarchical Grouping to Optimize an Objective Function". English. In: *Journal of the American Statistical Association* 58.301, pages.

Watanabe, Satosi and Chacko T. Abraham (1960). "Loss and recovery of information by coarse observation of stochastic chain". In: *Information and Control* 3.3, pp. 248–278.

Watkins, C (1989). "Learning from Delayed Rewards". PhD thesis. Cambridge University, England.

Watkins, David S. (2010). *Fundamentals of Matrix Computations, Third Edition.* John Wiley and Sons.

Weber, Marcus, Wasinee Rungsarityotin, and Alexander Schliep (2004). *Perron Cluster Analysis and Its Connection to Graph Partitioning for Noisy Data.* Tech. rep. November. Berlin: Zuse Institute Berlin ZIB.

Wei, Yen-Chuen and Chung-Kuan Cheng (1991). "Ratio cut partitioning for hierarchical designs". In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 10.7, pp. 911–921.

— (1989). "Towards efficient hierarchical designs by ratio cut partitioning". In: *Computer-Aided Design, 1989. ICCAD-89. Digest of Technical Papers., 1989 IEEE International Conference on*, pp. 298–301.

Zhang, Q., G. Yin, and E.K. Boukas (1997). "Controlled Markov Chains with Weak and Strong Interactions: Asymptotic Optimality and Applications to Manufacturing". In: *Journal of Optimization Theory and Applications* 94.1, pp. 169–194.