基于人工智能的 SPARQL 查询优化

冯运

院 （系）： 计算机科学与技术学院 专 业： 计算机科学与技术

学 号： 1160300524 指导教师： 王宏志

2020 年 6 月 10 日

哈爾濱工業大學

毕业设计（论文）

题　　目　基于人工智能的

SPARQL 查询优化

专　　业　计算机科学与技术

学　　号　1160300524

学　　生　冯运

指 导 教 师　王宏志

答 辩 日 期　2020 年 6 月 10 日

摘　要

随着知识图谱的广泛使用和 RDF 数据规模的不断增大，业界对 SPARQL 查询效率不断提出更高的要求，所以 SPARQL 的查询优化一直是研究的热点。近年来，人工智能技术在关系型数据库中得到了大量的研究和应用，为 SQL 的查询效率带来了显著的提升，然而基于人工智能的 SPARQL 查询优化却一直没有得到研究，所以本文旨在使用强化学习技术来优化 SPARQL 查询，实现传统优化方法无法达到的优化效果。

本文针对 SPARQL 查询中的基本图模式，使用强化学习方法来优化其中三元组模式的连接顺序。本文将强化学习问题转换为马尔可夫决策过程 (MDP)，定义了马尔可夫决策过程的五元组 (状态，动作，策略，奖励，初始状态)，使用了创新的方法对状态和动作进行特征化，并提出了奖励的计算方法。本文基于 Apache Jena 图数据库，使用了自己实现的优化模块对其中的基本图模式优化模块进行替换，对强化学习模型进行了训练和改进，最后使用 LUBM 标准数据集和查询集对本文的优化方法进行了测试和验证。

关键词：SPARQL 查询优化；强化学习；马尔可夫决策过程，基本图模式；jena

Abstract

With the widespread application of knowledge graphs and the increasing scale of RDF data, the industry continues to raise higher requirements on SPARQL query efficiency. Hence, SPARQL query optimization has always been a research hot spot. In recent years, artificial intelligence technology has been widely researched and applied in relational databases, which has brought significant improvement in SQL query efficiency. However, SPARQL query optimization based on artificial intelligence has not been researched. This article aims to use Reinforcement Learning to optimize SPARQL queries and achieve great optimization effect that traditional optimization methods can't do.

In this paper, for the basic graph pattern(BGP) in SPARQL query, the reinforcement learning method is used to optimize the join order of the triple patterns. This paper converts the reinforcement learning problem to a Markov decision process (MDP) and defines the five-tuple(state, action, policy, reward, initial state) of MDP. The paper uses innovative methods to encode state and action, and proposes a method of calculating rewards. Based on the Apache Jena graph database, this article uses the optimization module implemented by myself to replace Jena's original BGP optimization module, trains and improves the reinforcement learning model. Finally, I use the LUBM standard data set and query set to test the effectiveness of the optimization method raised in this paper.

Keywords: SPARQL query optimization,Reinforcement learning,MDP,Basic graph pattern,Jena

目　录

第 1 章 绪论

1.1 课题的背景和研究的目的与意义

知识图谱作为人工智能的重要基石，近些年得到了不断的发展和应用。特别是搜索引擎开始利用知识图谱来改进搜索的质量，加快搜索的速度，并且实现语义搜索和推理。

知识图谱是由语义表示的网络，它能够帮助使用者搜索知识网络中的所有知识。知识网络是由许多个实体，属性和属性值构成的。它用全局唯一的标识符 URI 去代表一个实体，用属性来表现实体之间的关联关系或实体的内在特性。W3C 提出了一种资源描述框架 RDF[1] 来表示知识图谱网络。在 RDF 中，一条知识可以表示为一个 SPO 三元组（主体，属性，客体）。我们从图的角度来看 RDF 数据的话，RDF 的实体或属性值是图的顶点，RDF 的属性是图的边。为了方便 RDF 数据的检索，W3C 组织还提出了一种形式化查询语言 SPARQL[2]，这是一种类似于 SQL 的查询语言。SPARQL 查询以三元组模式 (triple pattern) 为基本单位，三元组模式之间通过多种运算符形成更多复杂的图模式。SPARQL 查询是将查询问题转化为图上的子图匹配问题，因为这一过程可以看作在 RDF 图中找到全部满足查询图模式的匹配。

随着知识图谱应用的广泛使用，到目前为止，已经出现了很多百万规模顶点和上亿条边的知识图谱数据集[3]，例如，DBpedia 和 LinkedGeoData 各自都有大于 30 亿条的数据，而蛋白质知识图谱 (UniProt) 更是有大于 130 亿条的数据。在数据量如此巨大的情况下，实际应用场景需要查询性能更高的图数据库来满足业务需要。在这种情况下，业界对图数据库性能的优化，尤其是对 SPARQL 的查询优化显得尤为重要。

近年来人工智能的迅速发展为数据库查询问题提供了新的研究思路[4]。人工智能技术强大的适应性和数据表征能力能够智能地对大量、复杂、动态的数据和工作负载进行深入的分析。在现有工作中，已有许多基于机器学习的数据管理技术被提出。研究人员分别从数据库系统调优和查询优化两方面出发，提出了如 SageDB[5]、GALO[6]、Neo[7]、SkinnerDB[8] 等人工智能赋能的数据库新技术。

虽然研究人员利用人工智能技术针对查询优化已经进行了很多研究工作，但是它们大多是面向关系数据库的，而尚未有研究针对图数据库的智能查询优化问题。因此，基于人工智能的图数据查询优化问题亟待解决。

综上所述，由于图数据库中的 RDF 数据规模不断增大，需要优化 SPARQL 查询过程才能继续满足应用对查询的效率要求。研究人员应用人工智能技术在 SQL 的查询优化中已经取得了显著的成效，而且尚未有人应用人工智能针对 SPARQL 进行优化，所以本课题使用人工智能技术对 SPARQL 查询进行优化是有价值有必要的。

1.2 国内外研究现状

1.2.1 人工智能在关系型数据库查询优化中的应用

查询优化问题已经被研究了四十多年，至今仍然是一个活跃的研究领域。该问题的组合复杂性使得启发式算法得以普遍应用[9]。许多商用 DBMS 的查询优化策略都是基于 System R[10] 或 Volcano Optimizer Generator[11] 中引入的思想。这些系统使用了动态规划（DP）并结合一组规则来找到好的查询计划。这些规则缩减了潜在查询计划的搜索空间，减少了优化查询所花费的时间，但也会降低在大型搜索空间中找到最佳查询计划的可能性。传统查询优化方法除了搜索策略的局限性外，还存在第二个问题：它们依靠代价模型来估计执行查询的成本。这些代价模型建立在基数估计的基础上，基数估计大多基于分位数统计，频率分析或没有理论基础的方法。基数估计中的错误通常会导致查询计划的效率不够理想。此外，传统的查询优化器无法从以前执行的查询中学习。

受机器学习最新进展的启发，数据库研究的一个新趋势是将启发式算法用机器学习算法代替。Krishnan 等人探索了使用机器学习算法合成特定数据集的连接搜索策略[12]。假设一个给定的代价模型和计划空间，研究是否能够对一个特定数据集上所有可能的连接计划进行搜索。这一问题的目标是从之前能够大幅度减少搜索时间的计划中学习到适合的搜索策略。Krishnan 等人的主要发现是连接排序与强化学习有深层次的算法联系，即连接排序的顺序结构也支持强化学习问题的结构。所以，Krishnan 等人利用这一算法上的联系将强化学习嵌入到传统的查询优化器中，建立了一个基于强化学习的优化器 DQ。该优化器优化了选择-投影-连接、查询连接以及物理操作符选择，能够根据之前执行查询计划的结果训练一个强化学习模型，从而有效完善后续的查询操作。

Marcus 等人也将深度强化学习应用于查询优化中，提出了一个自动查询优化器，使得优化器能够自动调整特定的数据库，无需专业数据库管理员的干预[13]。此外，该优化器紧密结合过去查询优化和执行的经验反馈，从而显著提高了查询性能。Marcus 等人还提出了一个依赖深度神经网络来生成查询执行计划的查询优化器 Neo[7]。该优化器从已有的优化器生成查询优化模型，并持续根据到来的查询学

习，从成功的查询中建立模型，从失败的查询中进一步学习调整。此外，Neo 优化器能够适应不同的数据模式，且能够容忍一定的估计错误。

1.2.2 SPARQL 查询优化的研究

在 SPARQL 查询优化的方向上，也有很多研究者做出了努力。Markus 在 ARQ(Apache SPARQL Processor) 的基础上进行了改进[14]，做出了 optARQ 原型，它通过建立选择性估计索引的方式，来最小化查询中间结果的生成数量，从而提高查询的效率。Groppe 认为 SPARQL 语言不够精简[15]，他提出了 SPARQL 语言的核心部分，并称之为 coreSPARQL，它具有与 SPARQL 相同的表达能力，但是消除了 SPARQL 的冗余语言构造，而且它的语法对机器更加友好。Groppe 将 SPARQL 语言转换为 coreSPARQL 语言，并开发了一组重写规则来优化 coreSPRQL 查询，以此来提高执行效率。

国内学者在 SPARQL 的查询优化上也做出了很多的努力。武汉大学信息资源研究中心提出了一种优化方法[16]，他们使用 RDF 的模式信息来精简 SPARQL 基本图模式 (BGP)，然后通过运用 B 树结构来迅速估计 SPARQL 连接图的节点大小和边的权值大小，估计连接代价并结合动态规划方法来找到最优逻辑查询计划。除了这种基于传统算法的优化方法，启发式算法也得到了大量的应用，例如，有学者提出了一种基于精英蚁群算法与权重矩阵的 SPARQL 查询优化算法[17]，他针对不同的图形状设计了有效的权重矩阵模型，为不同的查询形状设置了专门的优化参数，然后将权重矩阵作为蚁群算法的输入参数，分别利用人工蚁群与精英蚁群方法对 SPARQL 不同形状的查询语句进行优化。

1.2.3 研究现状分析

根据国内外对 SPARQL 查询优化的研究内容，对研究现状的分析如下：

1. 应用人工智能技术在 SQL 的查询优化中已经取得了显著的成效，例如 Neo 查询优化器和 DQ 查询优化器，其性能较传统方法有极大的提升而且还能在查询流中进行自学习，不断提升适应能力和查询效率。

2. 现有的针对 SPARQL 的查询优化主要是改进代价估计模型，精简 SPARQL 语言和改进 RDF 的存储方式，这些方法与 SQL 查询优化的研究一样，都存在搜索空间不足和代价估计模型不够准确的问题，而目前还没有采用人工智能技术的 SPARQL 查询优化研究。

1.3 本文研究内容和结构安排

本文的主要研究内容为基于人工智能的 SPARQL 查询优化技术，采用了强化

学习算法对 SPARQL 中的连接顺序进行优化，文章的主要章节如下：

第一章为绪论部分，开始介绍了本课题的研究背景和本研究的目的意义，然后对目前国内外对 SPARQL 查询优化的现状进行了分析和总结，最后给出了本文的研究内容和结构安排。

第二章介绍了本文用到的查询优化的预备知识。首先我们介绍了 RDF 图模型和本文的主要研究对象 SPARQL 查询语言，并给出了其相关概念的形式化定义和示例。然后我们又详细描述了强化学习的概念和本文将要用到的算法。最后我们介绍了图数据库中最常用的 Jena 框架。

第三章介绍了基于强化学习的 SPARQL 查询优化方法。我们首先详细的描述了 SPARQL 的基本图模式优化问题并确定了该优化问题的输出输出。之后，我们从马尔可夫决策过程的五元组（状态，动作，策略，奖励，初始状态）开始，详细的介绍了状态、初始状态、动作的定义和特征化方法，最后介绍了奖励值的计算方法。

第四章介绍了强化学习优化方法的实现，包括 Q 学习算法和深度 Q 学习算法的实现与改进以及优化模块的整体结构。之后介绍了实验的环境，包括 CPU、GPU、操作系统和基准测试。最后，给出了实验中两种算法的实测结果，通过与 Jena 自带优化器查询效率进行比较，来验证本文优化方法的有效性。

第 2 章 预备知识

2.1 RDF 图模型

万维网联盟 (World Wide Web consortium，简称 W3C) 为了在语义网络 (semantic web) 上表示和交换机器可理解的信息，创建了一种标准资源描述框架 RDF(resource description framework)。在互联网络中，所有能够用 RDF 数据表示的对象都称之为资源，例如所有的事物、概念和信息等。资源通常情况下使用唯一资源标识符 (URI) 来表示。

定义 2.1 RDF 项 (RDF Term) 用 I 来表示 IRI 的集合，用 RDF-L 来表示文本信息 (Literals) 的集合，用 RDF-B 来表示空值 (Blank nodes) 的集合，用 RDF-T 来表示 RDF 项 (RDF Term)。

定义 2.2 （RDF 三元组） RDF 三元组是由主体 (subject)，属性 (property) 和客体 (object) 组成的数据组合 (s,p,o)。三元组的取值空间可以表示为 $(I \bigcup RDF - B) \times (I \bigcup RDF - B) \times (I \bigcup RDF - B \bigcup RDF - L)$。

例如，对于三元组 <Tony, graduateFrom, HIT>，Tony 代表主体，graduateFrom(毕业于) 代表属性，HIT 代表客体。三元组的意思是 Tony 毕业于 HIT。

表 2-1 RDF 数据

RDF 三元组
<student1,rdf:type,Student>
<student1,telephone,"xxx-xxxx-xxxx">
<student1,takesCourse,course1>
<student1,memberOf,department>
<student1,advisor,teacher2>
<course1,rdf:type,Course>
<teacher2,teacherOf,course1>
<teacher2,teacherOf,course3>
<teacher2,worksFor,department1>
<department1,subOrganizationOf,university0>
<student4,takeCourse,course1>
<student4,takeCourse,course4>
<student4,email."xx@edu.com">
<teacher1,teacherOf,course4>

表格2-1是一个包含 14 个 RDF 三元组的数据集，图2-1是表格2-1RDF 数据对应的图模型表示，其中方框表示实体，双引号中的数值表示属性值，图上的边表示属性值。

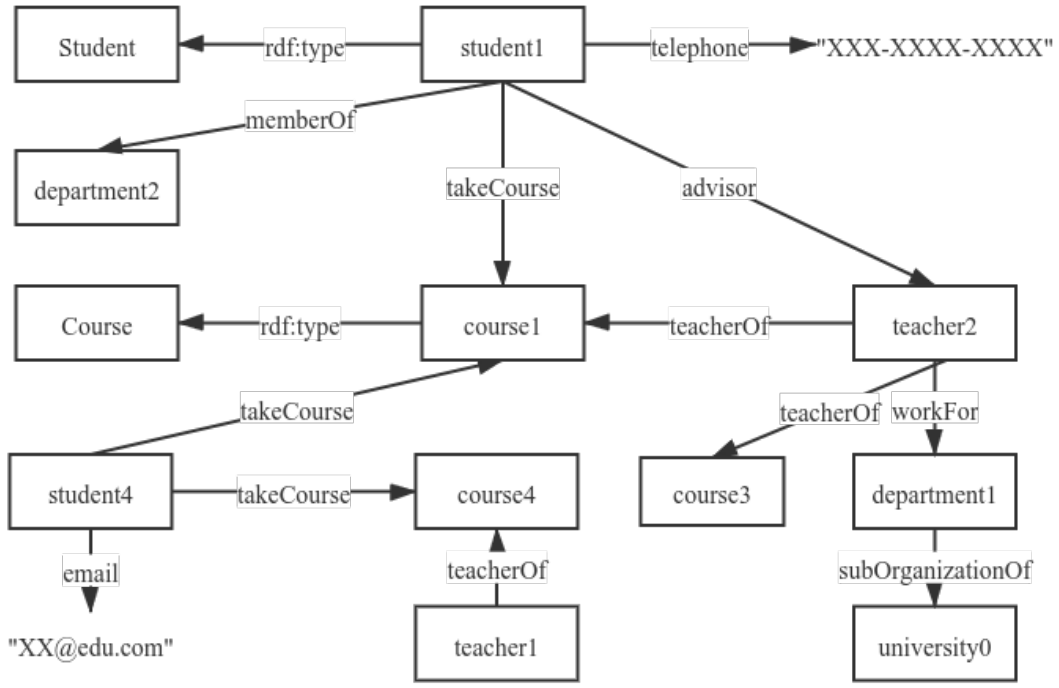

图 2-1　RDF 图模型

2.2 SPARQL 语言与查询过程

为了便于 RDF 数据的检索，万维网联盟 (简称为 W3C) 提出了一种形式化的查询语言 SPARQL，它是目前最广泛使用的查询语言之一。类似于 SQL，SPARQL 也支持例如 SELECT、DESCRIBE、CONSTRUCT 和 ASK 等多种查询格式。(1) SELECT 查询格式是一种最常用的标准查询，查询的结果是匹配到的 RDF 图数据。(2) ASK 查询返回结果不包含具体的数据，而是 yes 或 no 的判断结果。(3) DESCRIBE 查询是用于获取本体和实例数据的一部分，返回一个包含匹配图模式的结点相关信息的图形。

定义 2.3 （三元组模式） 一个三元组模式 (Triple Pattern) 是集合 $\{(RDF-T\bigcup V)\times(I\bigcup V)\times(RDF-T\bigcup V)\}$ 的一项，其中 V 代表变量。三元组模式常用 (s,p,o) 来表示。

定义 2.4 基本图模式 (BGP) 一个基本图模式是

$$\wedge_{1\leqslant i\leqslant m}(s_i,p_i,o_i)$$

其中

1. (s_i,p_i,o_i) 是一个三元组模式 $(1\leqslant i\leqslant m)$。
2. $\wedge$ 表示逻辑合取。

如图2-2是一个示例 SPARQL 查询，其中“?”加字母表示变量，WHERE 子

```
PREFIX rdf: <http :// www.w3.org/1999/02/22−rdf−syntax−ns#>
PREFIX ub: <http :// www.lehigh.edu/~zhp2/2004/0401/univ−bench.owl#>
SELECT ?X ?Y ?Z
WHERE
{
    ?X rdf:type  ub:GraduateStudent  .
    ?X ub:memberOf ?Z .
    ?Z  rdf:type  ub:Department .
    ?Z ub:subOrganizationOf  ?Y .
    ?Y rdf:type  ub: University  .
    ?X ub:undergraduateDegreeFrom ?Y .
}
```

图 2-2　SPARQL 查询示例

句是查询的主要组成部分，包含 6 个三元组模式 (triple pattern)，其中的三元组模式是用变量替换 RDF 三元组中的主体 (subject)，属性 (property) 或客体 (object) 而成的。Prefix 表示前缀定义，用于替换三元组模式中的相同字符串内容，用于提升 SPARQL 查询的可读性，类似于其他编程语言中的常量定义。

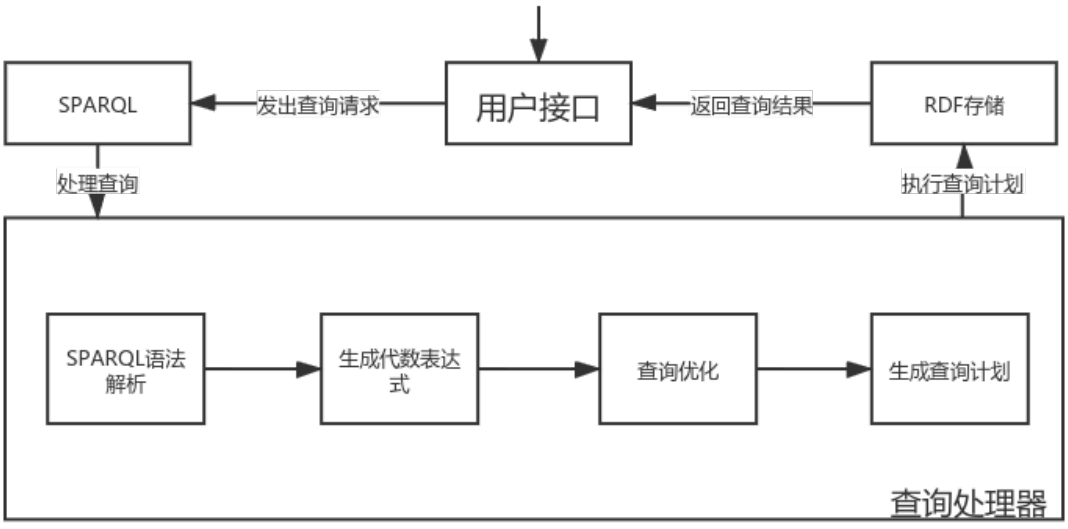

图 2-3　SPARQL 查询过程

如图2-3是一个普遍的 SPARQL 查询处理器模块图。首先，用户通过接口与

查询处理器进行交互，用户通过接口向处理器发出一个 SPARQL 查询请求，在 SPARQL 处理器接收到用户的 SPARQL 查询请求时，它会利用解析器对查询语句进行解析，包括词法和语法分析，如果分析过程中发现了错误，则会通过交互接口给用户反馈错误信息。之后，处理器会进入查询重写阶段，它根据一些优化规则对查询进行分析，重写成优化的查询，查询计划生成器会将优化后的查询转换成具体的物理查询计划，最后查询计划会交给 RDF 底层存储来执行，生成最终的查询结果并返回给用户。

2.3 强化学习

2.3.1 概述

强化学习 (Reinforcement learning)[18] 是机器学习 (machine learning) 的其中一种方法，这种方法的主要思路是让学习者 (agent) 通过不断获得环境的反馈来学习。婴儿学习走路的过程与强化学习的思路十分相近，当婴儿开始学习走路时，他刚开始并不能掌握平衡，并不会直接学会走路，但他通过与周围环境的交互获得反馈，例如摔倒会给他带来疼痛感（惩罚），顺利走出第一步保持了平衡（奖励），婴儿就会知道每个动作的后果（获得惩罚或奖励），从而逐渐选择奖励最多的动作，直到完全学会走路（奖励最大化）。当我们从计算机科学的角度看待这种方法时，我们可以将其解释为一个函数。该函数尝试使奖励累积结果最大化，而无需告知必须采取何种行动。

很多形式的有监督机器学习方法通常都有很多的输入输出对，每一个输入输出都是提前备好的训练数据集，训练过程会被告知要采取哪些动作，但是强化学习不同于前者的是，它必须通过尝试不同的动作，根据与环境的交互获得动作的反馈，从而确定采取哪些动作能产生最大的回报。强化学习更加专注于在线规划，需要在探索未知的领域和遵从现有知识之间找到平衡。

2.3.2 马尔可夫决策过程

从技术角度来看，强化学习是一种随机优化方法，它可以表述为马尔可夫决策过程 (MDP)[19]。其基本思想是，学习者 (agent) 采取一系列行动，以优化 MDP 模型中的给定目标。马尔可夫决策过程 (MDP) 是马尔可夫链 (Markov Chains) 的拓展，所以它需要满足马尔可夫链的属性。例如未来状态的进展仅取决于当前状态，而不取决于过去的状态和采取的动作。一个马尔可夫决策过程可以形式化的表示为以下五元组：

$$\langle S, A, P(s,a), R(s,a), S_0 \rangle \tag{2-1}$$

- S 代表学习者所有可能的状态。
- A 代表学习者可以采取的所有动作来到达一下个状态。
- $S_{t+1} \sim P(s,a)$ 是在状态 s 通过采取动作 a 到达的新状态。
- $R(s,a)$ 代表在状态 s 通过采取动作 a 获得的奖励。
- S_0 是学习者初始时的状态。

总体的目标是找到一个决策策略 $\pi : S \longmapsto A$（一个状态到动作的映射函数），使得期望的奖励最大化。马尔可夫决策过程是一个从交互中学习的过程，图2-4表示出了这个交互学习过程：Agent 是学习者，它在某个状态下采取动作并且与环境交互，Agent 从环境中获取到下一个状态和动作对应的奖励。状态信息通常需要编码，也被称为 observations。

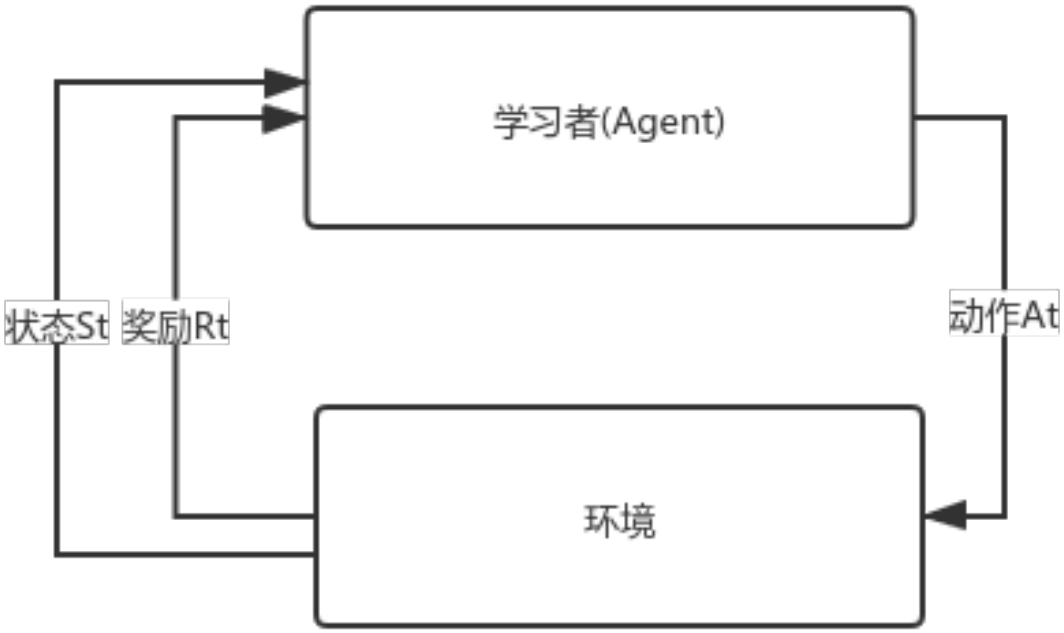

图 2-4　马尔可夫决策学习者与环境交互过程

2.3.3 Q 学习算法

Q 学习算法[20] 是由 Watkins 在其博士论文"Learning from delayed rewards"[21] 中第一次提出的，该算法是强化学习中具有里程碑意义的算法。Q 学习算法是时间差分算法 (TD)[22] 的一种，也是一种模型无关的算法，其主要针对具有折扣的奖励指标的控制。

Q 学习算法在迭代时针对（状态，动作）二元组进行奖惩，Q(s,a) 作为在状态 s 下执行动作 a 后获得的累计回报，即 Q 值函数。Q 值函数取决于当前的立即回报

和期望的延时回报，因此，学习系统每一次学习迭代都需要考察当前的这个动作。Q 学习算法的最佳策略为当前状态 s 下累计回报值最大时对应的策略，即：

$$\pi^*(s) = maxQ(s,a) \tag{2-2}$$

Q 值表是所有（状态，动作）二元组与其对应的 Q 值组成的一张二维表，每迭代一次，这张表的值就更新一次。在学习的过程中，通过不断更新 Q 值表中（状态，动作）二元组所对应的 Q 值，从而对 Q 值表进行更新。随着学习次数的增加，Q 值表逐渐趋于稳定，即收敛。学习者在之后做决策的时候，根据 Q 学习算法学习到的 Q 值表（经验）做出决策，同时，针对新变化的状态留有一定的学习余地，使得 Q 学习算法更适用于动态变化的环境。

在 t 时刻，环境的状态为 s_t 的情况下，假如学习者选择执行动作 a_i，通过学习者 (agent) 与环境的交互，学习者的状态由 s_t 变成了 s_{t+1}，并且环境反馈给学习者一个反馈值 r，算法根据公式2-3[18] 更新 Q 值表：

$$Q(s_t,a_i) = (1-\alpha) * Q(s_t,a_i) + \alpha * [r(s_t,a_i) + \gamma * maxQ(s_{t+1},a)] \tag{2-3}$$

其中

α 是学习因子，$0 < \alpha < 1$；

γ 是折扣因子，$0 < \gamma < 1$；

$r(s_t,\alpha_i)$：在 t 时刻，学习者的状态为 s_t，他执行动作 α_i 后从环境获得奖励；

$Q(s_t,\alpha_i)$：策略 $\pi = (s_t,\alpha_i)$ 所对应的累计奖赏值；

在动作选择的过程中，需要用到动作选择策略，可以使用随机动作法，也常采用贪心策略 $(\varepsilon - greedy)$，即学习者在选择动作时，以 ε 的概率自由探索，以 $1-\varepsilon$ 的概率根据学习的经验选择动作。动作选择策略只需要满足一定条件就能保证 Q 函数的最终收敛。所以，Q 学习算法是一种非常有效的模型无关的强化学习算法。

Q 学习算法的执行步骤如下：

1. 初始化。初始化学习因子、折扣因子和 Q 值表，t=0；
2. 初始化当前的状态 s_t；
3. 根据当前状态 s_t，Q 学习算法选择一个动作 a_i；
4. 执行动作 a_i，并确定下一个时刻的新状态 s_{t+1} 和环境的奖励 r；
5. 根据公式2-3更新 Q 值表，设置 $t = t + 1$；
6. 判断是否满足终止状态的条件，若满足，则结束算法，否则，返回到第二步。

2.3.4 深度 Q 学习算法 (DQN)

Q 学习算法能够较好地解决很多问题，但是如果在情况较为复杂，状态与动作的组合太多甚至无限多的时候，Q 矩阵的存储容量和计算复杂度将会急剧增大。这个问题并非无法解决，Graves 等人提出了结合深度学习与 Q 学习算法的深度 Q 学习算法[23]，也被广泛地称为 DQN。目前，DQN 受到了许多关注和探索，不仅被应用于自动控制领域，也被尝试应用于包括计算机视觉领域在内的其他领域。

一般而言，深度学习与强化学习结合的思路有以下几种：利用深度神经网络来模拟环境，即输出动作对状态的改变方式和相应的回报值；利用深度神经网络来估计 Q 值函数；直接利用深度神经网络来学习策略。DQN 属于第二种，可以通过神经网络回归拟合出 Q 值的近似值，而不是像 Q 学习算法一样记住所有的 Q 值。

强化学习结合深度学习的困难在于：

1. 深度学习的成功依赖于大量的标注样本，从而能够进行有监督学习，而强化学习只有一个奖励值 r，而且这个值总是伴随有噪声、延迟和稀疏性，无法给每一个状态一个即时的回报作为监督信息。

2. 深度学习的样本都是独立的，而强化学习中的状态与状态之间是具有相关性的，前后的状态之间会相互影响。用于深度学习模型的样本如果具有相关性，将会严重影响模型的训练效果。

3. 深度学习的目标分布是固定的，例如图像分类任务中，猫就是猫，不会随时间改变。然而强化学习中的输入和学习目标都在随着决策的过程不断改变，使得训练十分不稳定。

为了解决这些困难，DQN 提出了两个关键的技术：经验回放 (Experience replay) 和目标网络 (Target network)，使用 Q 学习来为神经网络提供有监督数据，保证了 DQN 的训练与普通的监督学习保持一致。

经验回放就是建立一个经验池，把每次训练收集到的经验都保存起来，这里的经验是指学习者状态间的转移和相应的回报，在训练的时候，随机从这个经验池中采样一个批次的数据作为深度神经网络的训练数据。由于从经验池中采样数据的操作是随机的，所以采样所得的训练数据可以近似认为是独立同分布的，这样就可以满足深度神经网络对训练数据的要求。其中，动作的选择一般采用常规的贪心策略 ($\varepsilon - greedy$)。

目标网络是一种训练技巧，其通过构造两个一模一样的网络 θ 和 θ'，利用 θ 来进行 Q 值的回归但是不对其参数进行更新，θ' 网络的参数则随着训练的过程不断更新，在训练达到一定的迭代次数后，再将 θ' 中的参数赋值给 θ 中。这样做的目

的是在一定时间内保证目标（Q 值）网络的稳定，避免了网络训练过程中去拟合一个随时变化的目标分布。此外，由于网络训练的过程中，目标网络参数 θ 的参数不是即时变化的，所以尽管训练的数据不是完全独立同分布的，对网络训练的影响也会减小。

DQN 算法见伪代码2-1：

算法 2-1 DQN 算法

初始化容量为 N 的经验池 D;
以随机参数初始化动作值函数 Q;
for *episode = 1 to M* **do**
    初始化状态序列 $s_1 = x_1$，预处理后的状态序列 $\phi_1 = \phi(s_1)$;
    for *t=1 to T* **do**
        以概率 ε 选择一个随机动作 a_t;
        否则，$a_t = maxQ^*(\phi(s_t), a; \theta)$;
        执行 a_t 并求得相应的回报 r_t 和得到新的输入 x_{t+1};
        更新状态 $s_{t+1} = s_t$ 与 $\phi_{t+1} = \phi(s_{t+1})$;
        将 $(\phi_i, a_t, r_t, \phi_{t+1})$ 存进经验池 D 中;
        从 D 中随机采样一个批次的样本 $(\phi_i, a_t, r_t, \phi_{t+1})$;
        求得

$$y_j = \begin{cases} r_j, & \text{for terminal } \phi_{j+1} \\ r_j + \gamma max_{\alpha'} Q(\phi_{j+1}, a'; \theta), & \text{for else } \phi_{j+1}. \end{cases}$$

        利用该监督信息对深度神经网络进行监督学习;
    end
end

2.4 Jena 框架简介

Apache Jena（简写为 Jena）是一个免费的开源 Java 框架，是一个用于构建语义网络和链接数据的应用程序。Jena 最初是英国布里斯托尔的 HP Labs 的研究人员在 2000 年开发的，并且在 2010 年成为 Apache 软件基金会的开源项目。Jena 一直是一个开源项目，并且已广泛用于各种语义网络应用程序中。

Jena 框架由多个不同的 API 组成，它们相互交互来处理 RDF 数据。其结构如图2-5所示。

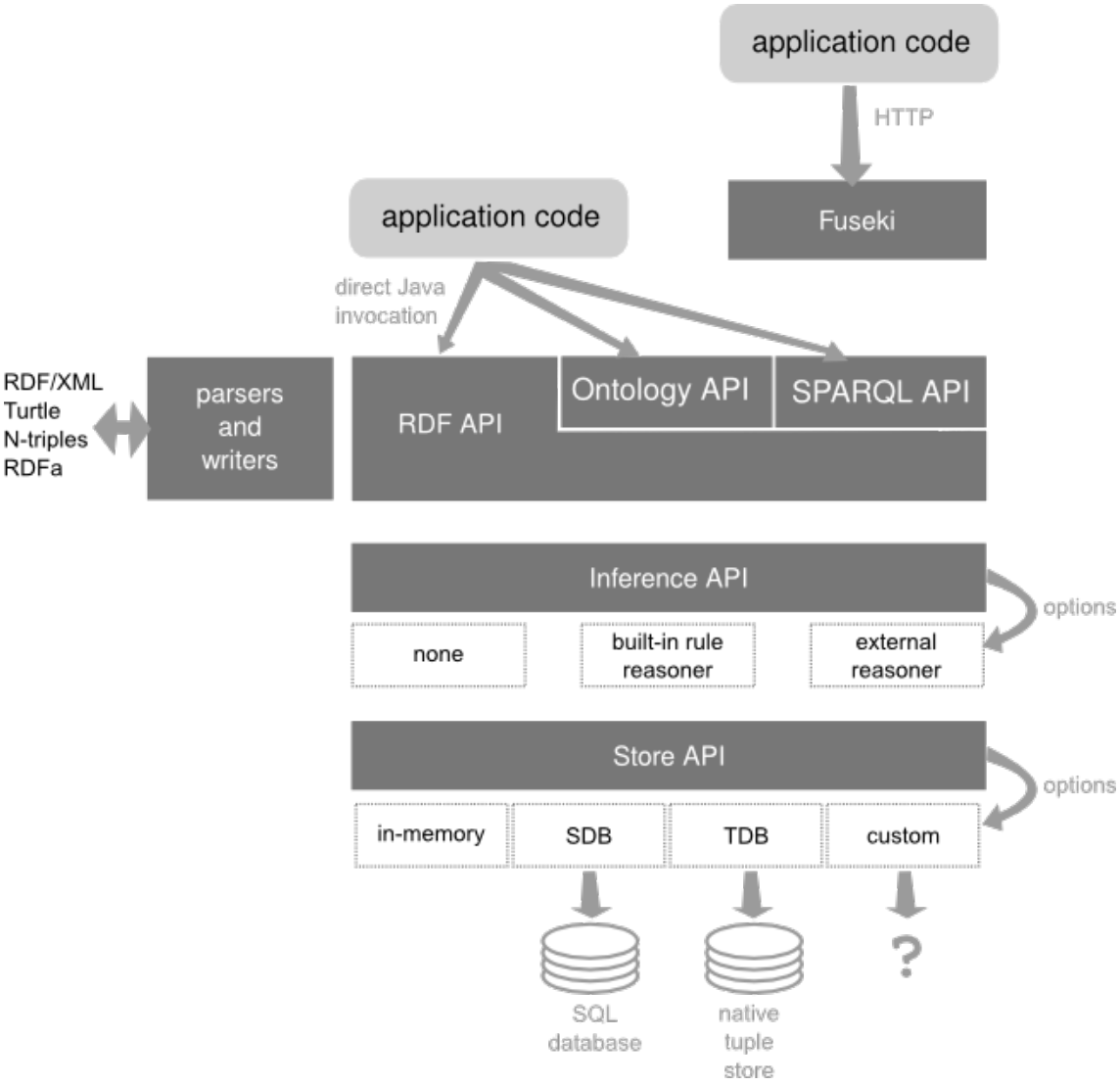

图 2-5　jena 框架

Jena 具体的接口和功能主要有：

1. RDF 模块。RDF 单元的主要功能是它可以将 RDF 信息映射到 Java 模型中，能够读写和处理 RDF 数据，同时给其他的模块提供接口。

2. 查询模块。Jena 提供了 ARQ 查询引擎来负责 SPARQL 语言的查询功能，以实现复杂的知识查询功能。

3. 本体 (Ontology) 模块。Ontology 模块是基于 RDF 单元构建的，其最主要的功能是进行 OWL 相关的操作。Ontology 模块中大量的 API 接口都运用了 RDF 模块中的技术，可以实现的功能与 RDF 模块相近。

4. 基于规则的推理引擎。这个模块提供对本体的类、属性和实例的推理功能，推理器可以由 Jena 提供，也可以使用外部的推理器。用户可以自己定制推理规则，检测 RDF 和 OWL 本体模型的一致性。

5. 存储模块。这个模块负责持久化存储 Model。Jena 提供了三种存储方式：

(a) 内存模式。该方式是把 Model 中的信息读取并存储到内存中，因此它的存储速度比其他的存储模式更快，但是这样会受制于本机的内存容量，不适合存储较大规模的模型。

(b) TDB 方式。TDB 方式是以本地三元组的方式进行存储，是 Triple Database 的意思。TDB 是一个只负责查询与存储 OWL/RDF 数据的图数据库，而且能够通过 SPARQL 语言来完成基于图的精确有效的查询。TDB 使用的是硬盘的存储方式，拥有较大的存储容量，能够满足大部分用户的存储需求。目前，Jena 最推荐且常用的一种存储方式就是 TDB。

(c) SDB 方式。SDB 是一种基于 SQL 关系型数据库的存储方式。这种方式是先分割本体模型中的数据关系，之后把关系保存到关系型数据库中，这样就可以将 Jena 框架与已经成熟的 SQL 数据库连接，使得开发者构建应用时可以利用存有的关系型数据。但是 SQL 的性能和稳定性都不如 TDB 的方式。

6. 服务机制。Jena 提供了 Fuseki，它是一个 SPARQL 服务器。它可以作为操作系统，Java Web 应用程序和独立服务器运行。它提供了安全性方面的保障机制 (Apache Shiro)，并具有用于服务器监视和管理的用户界面。

2.5 本章小结

在这一章中，我们主要对本文中使用到的基础技术进行了详细的介绍。我们首先介绍了 RDF 图模型的定义，包括几个重要名词解释和形式化的定义，还介绍了 SPARQL 查询语言的定义、语法、查询的构成和查询的执行过程，这两个都是本文后续研究的重要对象。之后，又介绍了强化学习的基本概念，马尔可夫决策过程及本文后续要用到的 Q 学习算法和深度 Q 学习网络算法。最后，我们还介绍了图数据库中最常用的 Jena 框架，包括其模块架构和功能。在下一章中，我们将介绍如何使用强化学习来对 SPARQL 查询过程进行优化。

第 3 章 基于强化学习的 SPARQL 查询优化算法

在本章中，我们先从优化问题的描述开始，定义优化问题的输入输出，然后再介绍马尔可夫决策模型的定义，描述如何定义状态，动作，奖励等要素以及其编码方式。

3.1 优化问题描述

SPARQL 的查询多种多样，其中包括图模式匹配查询，导航式查询和分析性查询这三个大类型，但是最常见最常用的查询是图模式匹配查询。在图模式匹配查询中，基本图模式匹配查询 (BGP) 又是核心。在如图3-1的 SPARQL 查询语句中，根据定义2.3，T_1 到 T_6 是 6 个三元组模式，根据定义2.4，$T_1 \wedge T_2 \wedge T_3 \wedge T_4 \wedge T_5 \wedge T_6$

```
SELECT ?X ?Y ?Z
WHERE
{
    ?X rdf:type  ub:GraduateStudent  .   ------T1
    ?X ub:memberOf ?Z .                  ------T2
    ?Z  rdf:type  ub:Department .        ------T3
    ?Z ub:subOrganizationOf  ?Y .        ------T4
    ?Y rdf:type  ub: University   .      ------T5
    ?X ub:undergraduateDegreeFrom ?Y     ------T6
}
```

图 3-1　SPARQL 查询示例

是一个基本图模式 (BGP)。

假设在某个 RDF 数据集中，对三元组模式 T_1 的查询结果是表3-1的中间结果，三元组模式 T_1 的语义是查询类型是研究生 (graduate student) 的全部实体。在中间结果表格中，变量 ?X 表示的结果是符合条件的全部学生实体的 URI。

假设在相同的 RDF 数据集中，对三元组模式 T_2 的查询结果是如下表3-2的中间结果，三元组模式 T_2 的语义是查询 ?X 是 ?Z 的成员 (member) 的模式。在中间结

表 3-1　三元组模式 T_1 的查询中间结果

?X	rdf:type	ub:GraduateStudent
GraduateStudent101	rdf:type	ub:GraduateStudent
GraduateStudent61	rdf:type	ub:GraduateStudent

表 3-2　三元组模式 T_2 的查询中间结果

?X	ub:memberOf	?Z
UndergraduateStudent242	ub:memberOf	Department15.University2
GraduateStudent101	ub:memberOf	Department14.University2
GraduateStudent61	ub:memberOf	Department8.University2

果表格中，变量 ?X 表示的结果是符合条件的全部学生实体的 URI，变量 ?Z 表示的结果是所有组织实体的 URI，可以看到与表格3-1中表格中 ?X 全是研究生 (graduate student) 结果不同的是，表格3-2的第一行出现了本科生 (undergraduate student 242)。

在图数据库处理该 SPARQL 查询时，会先查询得到三元组模式 T_1 的中间结果3-1，然后再查询得到三元组模式 T_2 的中间结果3-2，之后会将两个中间结果进行连接操作，从而得到新的中间结果如下表3-3所示：

表 3-3　三元组模式 T_1 与 T_2 连接操作后的中间结果

?X	ub:memberOf	?Z
GraduateStudent101	ub:memberOf	Department14.University2
GraduateStudent61	ub:memberOf	Department8.University2

从 $T_1 \bowtie T_2$ 以此类推，当最后完成 $T_1 \bowtie T_2 \bowtie T_3 \bowtie T_4 \bowtie T_5 \bowtie T_6$ 后，将得到 SPARQL 查询3-1的最终结果。

这 6 个三元组模式的连接顺序将会对中间结果的大小以及查询所需时间产生巨大的影响。例如，在相同实验环境和相同的数据集规模下，使用连接顺序 1：$T_1 \bowtie T_3 \bowtie T_2 \bowtie T_5 \bowtie T_4 \bowtie T_6$ 所需的查询时间实测为 6.64s，而使用连接顺序 2：$T_3 \bowtie T_2 \bowtie T_6 \bowtie T_4 \bowtie T_5 \bowtie T_1$ 所需的查询时间实测为 0.67s，查询时间对比图如图3-2所示。由此可见，在基本图模式 (BGP) 中的三元组模式的连接顺序具有非常大的优化空间，好的连接顺序将比差的连接顺序效率提升 9 倍左右。同时，连接顺序的空间复杂度是 $(n)!$，n 是查询中三元组模式的个数，所以较大的解空间也将给查询优化带来很大的优化空间。

此优化问题的输入为 SPARQL 查询中的基本图模式 (BGP)，例如：$T_1 \wedge T_2 \wedge T_3 \wedge T_4 \wedge T_5 \wedge T_6$；输出为一个优化后的三元组模式连接顺序，例如：$T_3 \bowtie T_2 \bowtie T_6 \bowtie T_4 \bowtie T_5 \bowtie T_1$。

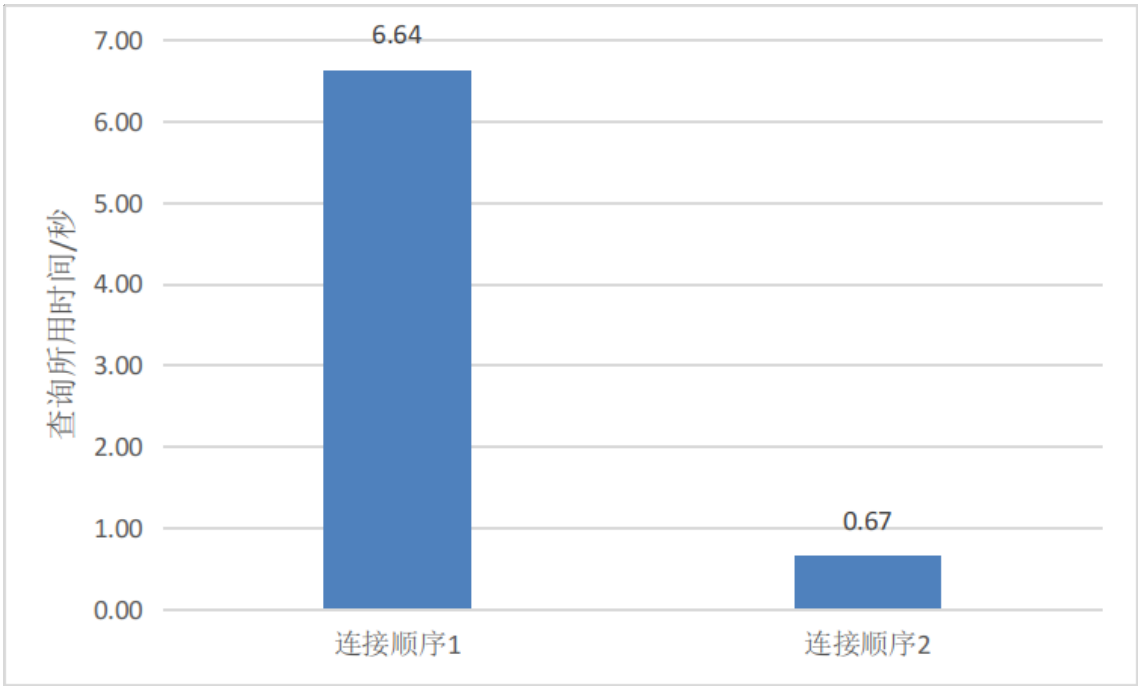

图 3-2　不同连接顺序所用查询时间

3.2 马尔可夫决策过程定义和特征化

为了能够把一个优化问题表述为一个强化学习问题，我们需要把问题形式化为一个马尔可夫决策过程 (MDP)。而在第二章预备知识中，我们介绍过，一个马尔可夫决策过程可以形式化的表示为五元组2-1，所以在本小节中，我们将依次介绍每一个元素在此优化方法中的具体定义，举出相应的例子，之后再根据定义给出其相应的编码方式以便于计算机的识别和处理，即对每个元素进行特征化处理。

3.2.1 状态的定义和特征化

3.2.1.1 状态的定义

S 代表学习者所有可能的状态，对任意时刻 t 的状态 S_t 都满足 $S_t \in S$。在我们的优化问题中，我们用 S_t 来表示在 t 时刻基本图模式 (BGP) 的连接状态，S_0 是初始状态，表示还没有开始做连接。

以查询3-1为例子，这个查询中包括 6 个三元组模式 (Triple)：$T_1, T_2, T_3, T_4, T_5, T_6$。

以下过程将展示状态是如何随时间变化的：

1. 初始时的状态 $S_0 = \phi$，表示所有的三元组模式都没有连接。

2. 如果在 t_1 时刻，已经得到了三元组模式 T_1 的查询中间结果，则此时的状态可以表示为 $S_{t1} = T_1$；

3. 如果在 t_2 时刻，三元组模式 T_1 的查询中间结果与 T_3 的查询中间结果进行了连接操作，则此时的状态可以表示为 $S_{t2} = T_1 \bowtie T_3$；

4. 如果在 t_3 时刻，前面的连接中间结果又与三元组模式 T_2 的查询中间结果进行了连接操作，则此时的状态可以表示为 $S_{t3} = T_1 \bowtie T_3 \bowtie T_2$；

5. 以此类推，如果最终查询结束时，这 6 个三元组模式 (Triple) 查询中间结果的连接顺序是 $T_1 \bowtie T_3 \bowtie T_2 \bowtie T_5 \bowtie T_4 \bowtie T_6$，则在查询处理结束的时刻 t_6 时，状态的表示为 $S_{t6} = T_1 \bowtie T_3 \bowtie T_2 \bowtie T_5 \bowtie T_4 \bowtie T_6$。

3.2.1.2 特征化

在将以上描述的状态定义进行特征化时，我们面临着两个挑战：

1. 如何将一个三元组模式表示为一个编码，如何通过编码来体现这些三元组模式中间查询结果的连接顺序；

2. 每一个 SPARQL 查询都有不同的基本图模式 (BGP)，即如果有两个不同的 SPARQL 查询，则它们的三元组模式和总个数都不尽相同。而对于强化学习算法来说，其输入数据（状态的特征编码）的维度需要维持衡定，所以如何将个数变化的三元组模式集特征化为一个长度固定的编码是比较困难的。

为了能够解决上面的两个问题，我们需要先介绍三元组模式的两个种类：

1. 类型三元组模式 (Type triple)。这种三元组的特征是其属性固定为 Type 的 URI: <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>。类型三元组模式是用来描述三元组 (s, p, o) 中主体 (s) 的类型是客体 (o) 的一类三元组模式，其形式化表述为：

$$Type\ triple \in \{(RDF - T \cup V) \times RDF : Type \times (RDF - T \cup V))\}$$

其中 V 表示变量，$RDF : Type$ 代表 type 的 URI：<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>。

2. 属性三元组模式 (Property triple)。这种三元组是所有三元组模式中除去类型三元组后的部分。其用于表示一个三元组 (s, p, o) 中主体 (s) 的某属性 (p) 是客体 (o)。其形式化表述为：

$$Property\ triple \in \{Triple\} - \{Type\ triple\}$$

其中 $\{Triple\}$ 表示三元组模式的集合，$\{Type\ triple\}$ 表示类型三元组模式的集合。

例如，三元组模式 T_1：" ?X rdf:type ub:GraduateStudent " 是一个类型三元组模式，它的查询语义是查询所有类型是研究生 (Graduate) 的主体 (subject)。三元组模式 T_2：" ?X ub:memberOf ?Z " 是一个属性三元组模式，它的查询语义是查询所有符合主体 (subject) 是客体 (object) 的成员 (member) 的三元组。

根据以上的定义，我们首先对输入的 SPARQL 查询中的三元组模式进行映射。

映射规则如下所示：

$$T_i \rightarrow o, if T_i \in \{Type\ triple\} \& T_i = (s, rdf:type, o) \tag{3-1}$$

$$T_i \rightarrow p, if T_i \in \{Property\ triple\} \& T_i = (s, p, o) \tag{3-2}$$

对于一个三元组模式，如果它是一个类型三元组模式，则将其映射为它的客体 (subject)；如果它是一个属性三元组模式，则将其映射为它的属性 (property)。这样做映射是因为对于一个属性三元组模式来说，它的客体 (subject) 最能够反映这个三元组模式的特征，是一个三元组模式最重要的信息，所以用客体来唯一代表三元组模式是合适的。同理，对于属性三元组模式来说，属性 (property) 是最能反映一个属性三元组模式的信息，所以我们用属性来做属性三元组模式的映射结果。除此之外，在 Jena 图数据库中，一个类型三元组数据是以类型值（客体）作为索引项的，一个属性三元组数据是以属性作为索引项的，所以这样做映射还可以把图数据库的索引也加入到特征中，使得算法的输入数据具有更多的信息。

例如，三元组模式 T_1：" ?X rdf:type ub:GraduateStudent " 是一个类型三元组模式，根据映射规则3-1，将其映射为 ub:GraduateStudent。三元组模式 T_2：" ?X ub:memberOf ?Z " 是一个属性三元组模式，根据映射规则3-2，所以将其映射为 ub:memberOf。

在获得映射的结果后，我们将每一个映射结果都作为一个单独的特征位，用一个二进制位来表示，0 表示该三元组模式的中间查询结果还未被连接，1 表示该三元组模式的中间查询结果已经被连接。但是这样还是无法解决不同的基本图模式在特征化后得到的向量维度不统一问题。为了解决这个问题，我们需要在查询优化系统初始化时，就确定好特征化结果的向量维度，将所有查询中可能出现的三元组模式映射结果都罗列出来，形成一个三元组模式映射结果全集 M，然后将它们特征化为一个 one-hot 向量，这个向量的长度是全集 M 的大小。这样在之后的查询中，每个查询的三元组模式都会是初始时的三元组模式全集 M 的子集，每个查询的特征化结果就都可以变成一个定长向量了。

我们为了在系统初始化时刻获取三元组模式映射结果全集 M，需要在系统导入 RDF 数据时，对导入的数据进行统计。对导入的每一条 RDF 数据按照映射规则3-2和3-1进行映射处理，然后收集所有的映射结果形成全集 M。算法伪代码见算法3-1。

例如，在系统开始时导入了如表格3-4所示的 RDF 数据，则在导入时，进行如下统计操作：初始时，集合 MT 和 MP 都是空的，在导入三元组 <stu-

算法 3-1 统计三元组模式映射结果算法

Input: RDF 三元组 T

Output: 类型三元组模式映射结果全集 MT，属性三元组模式映射结果全集 MP

```
初始化类型三元组映射结果集合 MT=ϕ;
初始化属性三元组映射结果集合 MP=ϕ;
for triple(s_i, p_i, o_i) in T do
    if triple 是类型三元组 and o_i 不在集合 MT 中 then
        将 o_i 加入到集合 MT 中;
    end
    else if triple 是属性三元组 and p_i 不在集合 MP 中 then
        将 p_i 加入到集合 MP 中;
    end
end
```

表 3-4　RDF 数据

RDF 三元组	类型
<student1,rdf:type,GraduateStudent>	类型三元组
<university0,rdf:type,University>	类型三元组
<student1,takesCourse,course1>	属性三元组
<student1,memberOf,department>	属性三元组
<course1,rdf:type,Course>	类型三元组
<department1,rdf:type,Department>	类型三元组
<department1,subOrganizationOf,university0>	属性三元组

dent1,rdf:type,GraduateStudent> 时，由于它是类型三元组，根据映射规则3-1，将 GraduateStudent 加入到集合 MT 中；三元组 <university0,rdf:type,University> 是类型三元组，则将 University 加入到集合 MT 中；三元组 <student1,takesCourse,course1> 是属性三元组，根据映射规则3-2，将 takeCourse 加入到集合 MP 中；三元组 <student1,memberOf,department> 是属性三元组，则将 memberOf 加入到集合 MP 中；三元组 <course1,rdf:type,Course> 是类型三元组，则将 Course 加入到集合 MT 中；三元组 <department1,rdf:type,Department> 是类型三元组，则将 Department 加入到集合 MT 中；三元组 <department1,subOrganizationOf,university0> 是属性三元组，则将 subOrganizationOf 加入到集合 MP 中。

最后，类型三元组映射结果集合 MT={GraduateStudent, University, Course, Department}。属性三元组映射结果集合 MP={takesCourse, memberOf, subOrganiza-

tionOf}。在获得三元组映射结果全集后，将集合中的元素做如下特征化：

GraduateStudent	University	Course	Department	takesCourse	memberOf	subOrganizationOf
0	0	0	0	0	0	0

因为向量中所有二进制位都是 0，所以此时以上编码状态处于初始状态 S_0。对于在 t_3 时刻，状态为 $S_{t3} = T_1 \bowtie T_3 \bowtie T_2$，即三元组模式 T_1, T_2, T_3 的中间查询结果已经被连接，这三个三元组模式的映射结果 GraduateStudent,memberOf,Department 对应的特征位将被设置为 1，其特征向量如下：

GraduateStudent	University	Course	Department	takesCourse	memberOf	subOrganizationOf
1	0	0	1	0	1	0

3.2.2 动作的定义和特征化

A 表示有限动作集（动作空间），对任意时刻 t 的动作 A_t，都满足 $A_t \in A$。在我们的优化问题中，在某个连接状态下，所有可能的连接动作构成了有限动作集。在 t 时刻，其连接状态为 S_t，则此时，所有可能的连接动作就是整个动作空间 A，而学习者会从动作集中选择一个动作 A_i 去实际执行。动作以选择连接的三元组模式中间结果的编号作为编码。

再以查询3-1为例子，这个查询中包括 6 个三元组模式 (Triple)：$T_1, T_2, T_3, T_4, T_5, T_6$。

以下过程将展示动作集和动作是如何随时间变化的：

1. 初始时的状态 $S_0 = \phi$，表示所有的三元组模式都没有连接，此时，可以从剩余的 6 个三元组模式中任选一个开始执行实际查询，与剩余的 6 个三元组模式进行连接操作构成了动作集，动作集大小为 6，假设学习者选择首先执行三元组模式 T_1 得到其中间查询结果，则它选择的动作是与三元组模式 T_1 的中间查询结果做连接操作，这个动作的编码是三元组模式 T_1 的编号 1。

2. 如果在 t_1 时刻的状态是 $S_{t1} = T_1$，此时，可以从剩余的 5 个三元组模式中间结果 T_2, T_3, T_4, T_5, T_6 中任选一个进行连接操作，动作集大小为 5，假设学习者选择与 T_3 的查询中间结果进行连接操作，这个动作的编码就是三元组模式 T_3 的编号 3。

3. 如果在 t_2 时刻，连接状态为 $S_{t2} = T_1 \bowtie T_3$，此时，可以从剩余的 4 个三元组模式中间结果 T_2, T_4, T_5, T_6 中任选一个进行连接操作，动作集大小为 4，假设学习者选择与 T_2 的查询中间结果进行连接操作，这个动作的编码就是三元组模式 T_2

的编号 2。

4. 如果在 t_3 时刻，连接状态为 $S_{t3} = T_1 \bowtie T_3 \bowtie T_2$，此时，可以从剩余的 3 个三元组模式中间结果 T_4, T_5, T_6 中任选一个进行连接操作，动作集大小为 3，假设学习者选择与 T_5 的查询中间结果进行连接操作，这个动作的编码就是三元组模式 T_5 的编号 5。

5. 以此类推，在每个状态下，选择从动作集中选择一个动作执行，直到最后的状态是所有的三元组模式的中间结果都得到连接，即得到了最终的查询结果为止。

状态转换的规则是在一个状态 S_t 下，如果学习者选择了动作 a_i，则将状态 S_t 的编码中三元组模式 T_i 对应的特征位置 1，状态由 S_t 转换成了状态 S_{t+1}。

3.2.3 奖励定义

奖励 (reward) 是学习者执行一个动作后环境给予的回馈信息，用于告知学习者这个动作执行的结果，来帮助学习者学习这个动作带来的价值。在我们的查询优化问题中，在查询优化器生成一个查询计划后，交给了底层 RDF 存储去执行查询计划，查询优化器将通过执行查询的代价来学习该查询计划的好坏，一个查询计划执行的代价越小，说明该查询计划的效率越高，越应该获得更高的奖励值。

我们首先想到将查询计划执行时间 t 的负数作为奖励值，因为查询时间 t 越小，说明查询的效率越高，这样学习者获得的奖励-t 就越大。但是这样的话还存在两个问题：

1. 在实际的执行过程中，查询计划的执行时间变化幅度很大，其范围在 100 到 10^6 之间，而深度强化学习方法通常能够处理的奖励值范围是-10 到 10 之间，幅度过大的奖励值对模型的训练会造成极大的干扰。

2. 在实际训练模型时，由于连接顺序的可能情况很多，模型不可能探索所有可能的连接顺序，对于没有被强化学习探索过的连接顺序，由于查询计划没有被实际被执行过，所以对应的奖励值是 0。而所有已经被探索过的连接顺序的奖励值都是小于 0 的，这样就造成模型会认为奖励值为 0 的连接顺序更优，模型最终总会把没有被探索过的连接顺序作为最优解，模型无法正常的运作。

为了解决第一个问题，我们可以通过对奖励进行归一化，来避免奖励值出现较大的变化幅度，而对于第二个问题，我们可以通过对奖励值进行一些变换使得所有的奖励值都是大于 0，并且能够满足奖励值随着查询代价的减小而增大。综上两个目的和方法，我们将奖励值定义如下：

$$R(S = s_t, A = a_i) = e^{-t} * 10 \tag{3-3}$$

其中 t 是查询计划的执行时间。通过以上定义，我们将奖励值的范围限定为 0 到 10 之间，来满足深度强化学习常用的奖励数值范围。如图3-3展现了奖励值随查询时间的变化关系，随着查询时间的变长，奖励值会逐渐趋向于 0，查询时间越接近 0，奖励值就越接近最大值 10，这样的对应关系就可以解决上文提到的两个问题。

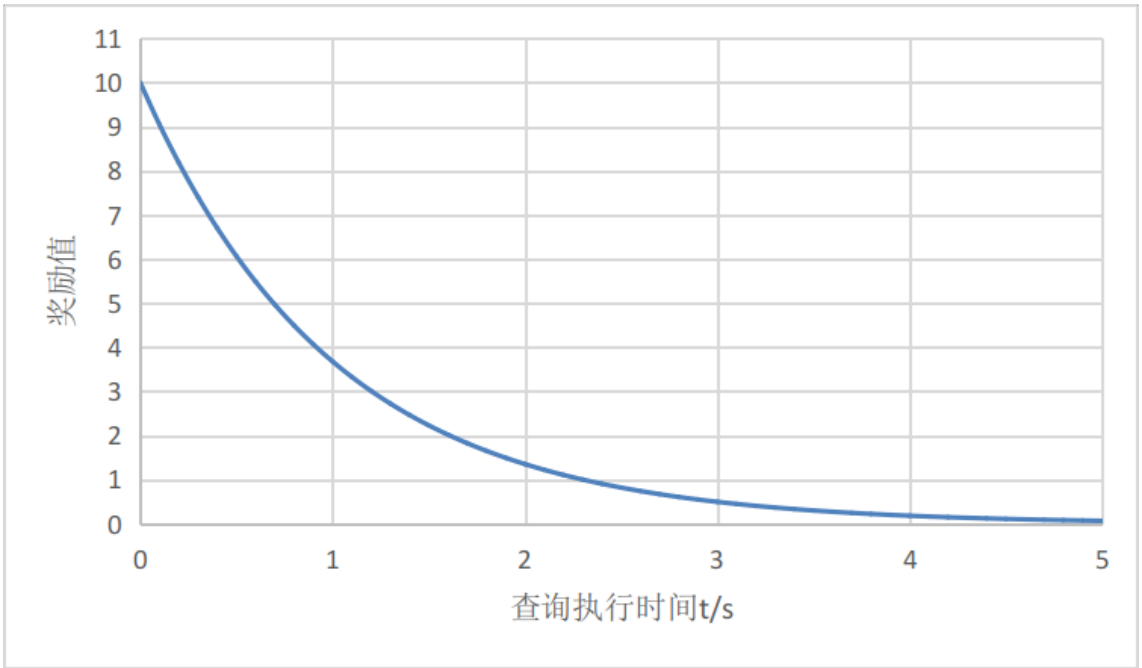

图 3-3　奖励值随查询时间的变化

3.3 本章小结

在这一章节中，我们提出了一种基于强化学习的方法来对 SPARQL 查询中基本图模式查询进行优化，该方法能够优化基本图模式中的三元组模式中间查询结果的连接顺序，从而到达提高查询效率的目的。我们首先对 SPARQL 的基本图模式优化问题进行了详细的描述，展现了其较大的优化空间，确定了该优化问题的输出输出。之后，我们从马尔可夫决策过程的五元组（状态，动作，策略，奖励，初始状态）开始，分别详细的介绍了状态、初始状态、动作的定义和特征化方法，然后通过举例的方式描述了状态之间的转换方式，最后介绍了奖励值的计算方法。

第 4 章 优化方法的实现和实验结果

在这章中，首先我们将基于 Apache Jena 开源图数据库，介绍基于强化学习的 SPARQL 优化方法的具体实现，然后介绍搭建的实验环境的参数，最后，我们使用基准测试数据 LUBM 来验证方法的有效性并进行不同算法之间的效果比较。

4.1 优化方法的实现

我们的算法基于 Q 学习算法和深度 Q 学习算法，但是我们又根据实际的问题情景，对两个算法进行了一些改进，以达到更高的效率和准确度。以下内容我们将对使用的框架，算法的改进和整体模块的实现进行介绍。

4.1.1 Q 学习算法实现

在第二章中，我们已经介绍过基本的 Q 学习算法，在这里我们将描述我们对数据结构的特殊设计。

Q 学习算法需要一个 Q 值表来存储 (状态，动作) 与 Q 值的映射，但是考虑到基本图模式连接状态的空间复杂度是 $n!$，动作最多有 n 个，则如果使用二维矩阵来存储 Q 值需要 $n * n!$ 的存储空间，n 是三元组模式映射结果全集 M 的大小，这样的存储方式需要的空间极大，在 n 值为 13 时就需要大约 60GB 内存，同时内存利用率也极低。因为连接状态空间极大，强化学习模型经过大量训练也只能探索极小的一部分空间，所以 Q 值表其实是一个稀疏矩阵。为了提高空间利用率，我们可以用 HashMap 来存储 (状态，动作) 与 Q 值的映射。我们用于存储 Q 值表的数据结构为 Map<List<String>, Map<String, Double>>，其中：

- List<String> 用于存储连接状态。String 存储的是三元组模式根据映射规则3-1和3-2得到的结果，List<String> 中元素的顺序代表当前状态下三元组模式中间结果的连接顺序。
- Map<String,Double> 用于存储动作和 Q 值的映射。String 存储的也是三元组模式的映射结果，但是它代表在当前状态下选择连接的三元组模式，是代表的连接动作，Double 类型用于存储 Q 值，Map 数据结果用于存储动作选择和 Q 值的映射关系。
- Map<List,Map> 用于存储状态与动作的映射关系。

4.1.2 深度 Q 学习算法 (DQN) 实现

由于我们选用的图数据库是 Apache Jena，它是使用 Java 语言实现的，所以我

们在实现深度 Q 学习算法时，为了方便与数据库进行交互，选择了一个由 Java 语言实现的强化学习库 RL4J(Reinforcement Learning for Java)，它的神经网络部分是由 DeepLearning4J 来完成的。我们对 RL4J 中的 MDP 类 (负责马尔可夫决策过程) 进行了重写，将状态，动作，状态转换，奖励和初始化操作根据章节三的方法进行了实现。特别的是，由于我们的动作定义的特殊性，我们需要对框架中的动作空间部分进行定制。与框架中的实现不同的是，我们的方法在每次状态更新后，合法的动作都会减少 1 个，动作空间需要更新，然后学习者再从更新的动作空间中选择一个动作去执行。

4.1.3 模块实现

在第二章中，我们介绍了一个普遍的 SPARQL 查询处理器结构如图2-3所示，其结构特点是从用户发出查询到用户收到查询的结构，所有的执行过程和信息传递是单向的。本方法实现的结构图如图4-1所示，强化学习优化器模块从语法解析器中获得基本图模式 (BGP)，然后使用强化学习算法确定出连接顺序，然后将结果交给查询计划执行模块去执行，不同于普遍 SPARQL 查询结构的是，我们的实现中，在 RDF 存储模块执行查询计划后，不仅会将查询结果返回给用户，还会将统计的查询代价返回给强化学习优化器，作为反馈，强化学习优化器模块将根据查询代价计算动作奖励，然后更新算法的参数。

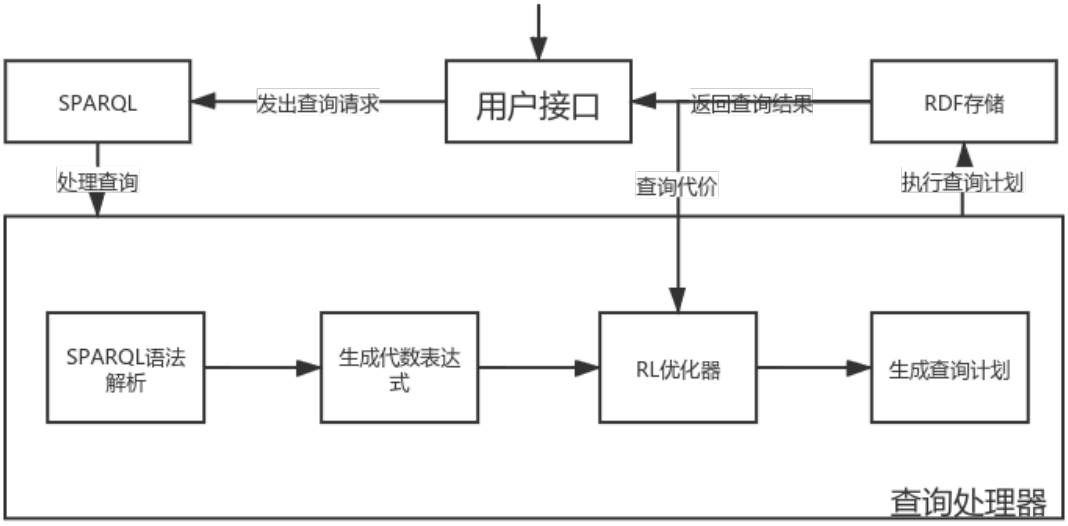

图 4-1　改进的 SPARQL 查询器结构图

在模型初始运行时，由于模型之前没有学习过如何安排基本图模式的连接顺序，模型给出的连接方案可能会很差，在模型刚开始运行时的查询代价可能会较高。但是随着处理的查询越来越多，模型从查询中积累了越来越多的经验，模型

给出的连接计划会越来越好，这一过程也是强化学习模型的训练过程。在实际运用中，我们的模块实现提供了两种使用方式：

1. 离线训练方式。在模型初始化时，如果 SPARQL 查询处理器没有查询需要处理，系统可以自行离线训练，通过运行一个用于训练的查询集，从而快速的让模型学习到大量的查询优化经验，从而在之后 SPARQL 查询处理器处理真正的查询时，能够给出高效的查询计划来满足实际的效率需求。

2. 在实际查询流中学习。在模型初始化时，SPARQL 查询处理器接收到了用户发送的实际查询请求，这样系统直接使用查询优化器提供的查询计划，随着处理查询的增多，查询优化器可以在不断处理实际查询的过程中不断学习，从而使得模型提供的查询计划越来越高效。

对于第一种方式，模型在实际使用时就可以提供高效的查询计划，使用效果较好；对于第二种方式，在优化模块使用的初期，给出的查询计划可能比较差，对实际业务需求可能会造成一定的影响，这就是模型的冷启动问题。在以后的进一步研究中，我们可以尝试在强化学习优化器中结合传统的查询优化方法来解决冷启动问题。例如在模块使用初期，更多的依靠传统的优化方法来产生查询计划，而在优化器积累了足够的经验后，就完全取代传统的优化方法，这样既可以实现高效率查询，又能够在不断的查询中不断提高查询效率。

4.2 实验环境

本文的实验是在一台笔记本电脑上进行的，该电脑的 CPU 是双核 Intel Core i7 6500U(2.5 GHz - 3.10 GHz)，8GB 内存；显卡是 Nvidia GeForce 940MX(1122 MHz - 1242 MHz)，显存 2GB，显卡驱动版本是 440.82，CUDA 版本是 10.2；电脑运行的系统是 Ubuntu 18.04。

我们使用的 Java 是 OpenJDK1.8.0，使用的 Apache Jena 版本是 3.14.0，使用的 RDF 底层存储是 TDB。我们使用的标准测试数据是 LUBM[24]，这个基准测试是由 Lehigh 大学发起，旨在以一种标准且系统的方式进行语义 Web 存储库的评估工作。LUBM 由大学领域的本体组成，可自定义和可重复的合成数据，包括 14 个标准 SPARQL 查询。由于实验环境的计算能力有限，而且 LUBM 数据集过于庞大，训练时间过于长，我们从 LUBM 的全部数据中随机抽样了 10% 的数据进行实验。

4.3 实验结果

在这一部分，我们将展示和分析实验结果，包括强化学习算法的收敛性检验，两种强化学习算法的优化效果及其与其他方法的效果比较。

在模块的实现中，我们提到了模块需要在多次的查询处理中不断的学习，从而实现查询效率的不断提高，但是这个功能的实现依赖于强化学习算法的收敛性，即算法能否在不断的训练中趋向于产生效率更高的查询计划。

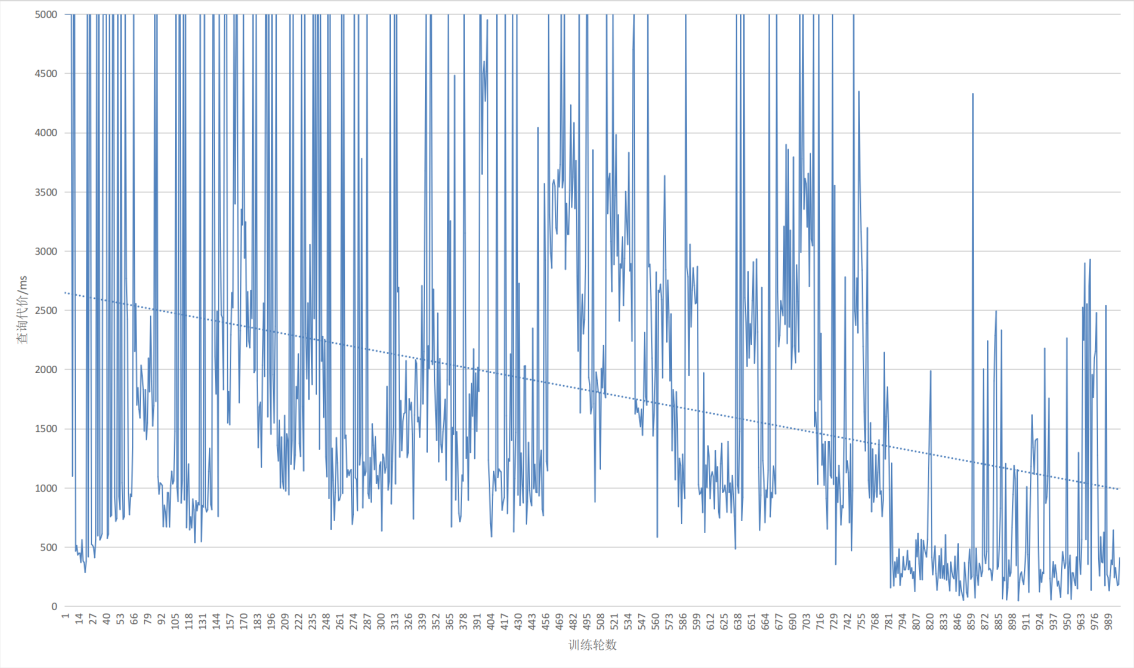

图 4-2　查询代价随训练轮数变化折线图

如图4-2，是查询代价随模型训练轮数的折线图，图中的蓝色直线是对折线进行线性拟合的结果。从折线的变化来看，随着训练轮数的增多，查询代价在波动着下降，特别是在训练次数为 700 次之后，查询代价显著下降，之后的大部分的查询代价都较小。从线性拟合结果来看，折线是一个明显的递减趋势，也与折线的变化趋势相符。从图中可以表明强化学习方法具有收敛性。

我们将 LUBM 的 14 个查询随机分为两组，一组有 8 个查询，作为训练查询集，一组有 6 个查询，作为测试查询集。我们在相同的 RDF 数据集下，Q 学习算法和 DQN 算法都迭代训练 100 轮，得到了在训练查询集下的查询代价，然后使用两个算法各自训练好的模型运行测试查询集，得到了各自的查询代价，之后使用 Jena 自带的基本图模式查询优化器分别运行训练查询集和测试查询集，记录了各自的查询代价。

如图4-3所示是三种查询优化器在训练查询集上的查询代价，可以看出不同的查询的查询代价差异较大，同时，可以看出在查询代价较大的查询上，Q 学习算法和 DQN 的代价都要小于 Jena 自带的 BGP 查询优化器，但是在查询代价较小的查询上，三种查询优化器的查询代价相差不大，三种查询优化器基本持平。从训练查询集上可以看出，在总体查询代价较大时，Q 学习算法和 DQN 算法都能显著提

升查询效率，同时，在大部分查询中，DQN 算法的效果略好于 Q 学习算法。

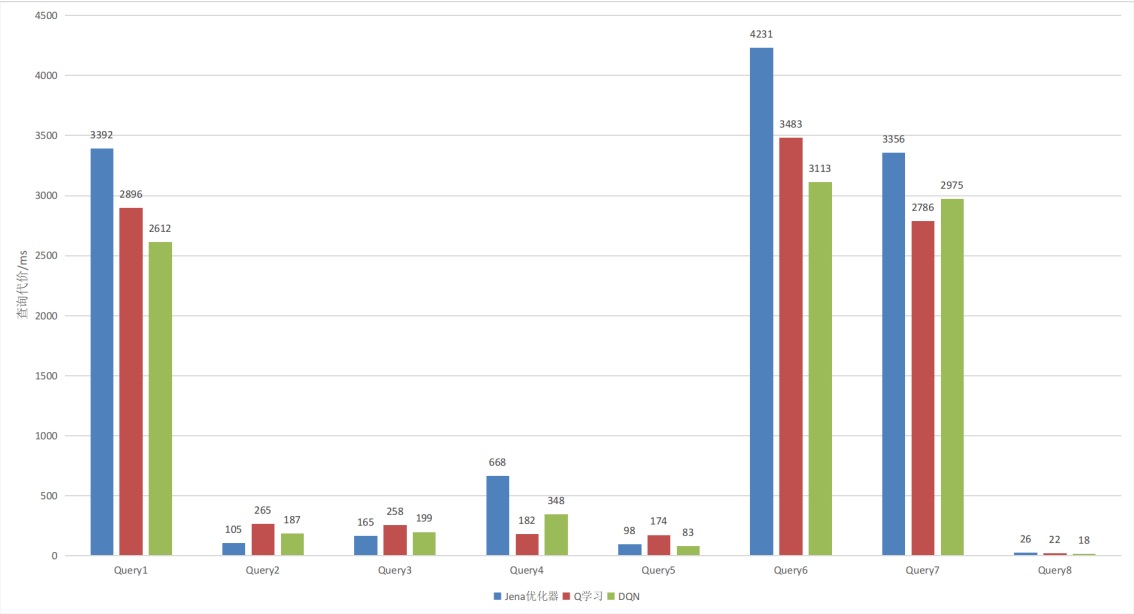

图 4-3　三种查询优化器在训练查询集上的查询代价

如图4-4所示的是三种查询优化器在测试查询集上的查询代价，从图中可以看出三种查询优化器的查询代价情况与在训练查询集上比较相似，但是 DQN 算法在测试集上的表现不如在训练集上，表明其可能存在一些过拟合情况。综上实验结果可以得出，在大多数情况下，强化学习的方法能够有效提高查询效率。

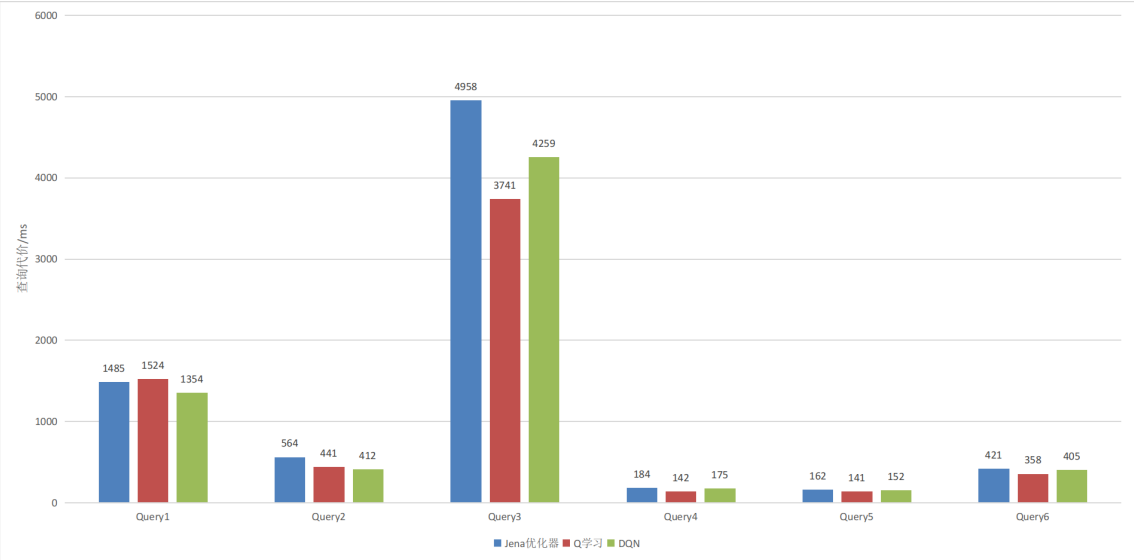

图 4-4　三种查询优化器在测试查询集上的查询代价

4.4 本章小结

在本章中，我们首先介绍了强化学习方法的实现，包括两种强化学习算法的

实现以及优化模块整体结构的实现，特别介绍了我们根据实际问题情景对算法的改进部分。之后介绍了实验的环境，包括 CPU、GPU、操作系统和基准测试。最后，我们给出了实验中两种算法的实测结果，通过与 Jena 自带优化器查询效率进行比较，来验证本文优化方法的有效性。

结　论

本文针对 SPARQL 查询中的基本图模式，使用强化学习方法来优化其中三元组模式的连接顺序。本文将强化学习问题转换为马尔可夫决策过程 (MDP)，定义了马尔可夫决策过程的五元组 (状态，动作，策略，奖励，初始状态)，提出了通过对三元组模式进行分类来提取其关键信息和特征化的方法，建立了一个奖励的计算方法，既能符合强化学习对奖励值的范围要求，也能正确表现奖励值随查询代价的变化趋势。本文基于 Jena 图数据库实现了强化学习的优化模型，使用 LUBM 基准测试数据对模型进行了有效性测试，得出结论：通过本文提出的强化学习方法可以显著提升 SPARQL 的查询效率。今后的研究可以在本文提出的方法基础上，进一步缩短训练模型需要的时间，扩展方法的使用范围和优化在实际应用场景下的表现。

参考文献

[1] Schreiber G, Raimond Y. RDF 1.1 Primer[J/OL], 2014. https://www.w3.org/TR/rdf11-primer/.

[2] Prud'hommeaux E, Seaborne A. SPARQL Query Language for RDF[J/OL], 2008. https://www.w3.org/TR/rdf-sparql-query/.

[3] 王鑫, 邹磊, 王朝坤, 等. 知识图谱数据管理研究综述[J]. 软件学报, 2019 : 2139-2147.

[4] Domingos P. Machine Learning for Data Management: Problems and Solutions[C/OL] // . 2018 : 629-629. http://dx.doi.org/10.1145/3183713.3199515.

[5] Kraska T, Alizadeh M, Beutel A, et al. SageDB: A Learned Database System[C] // . 2019.

[6] Damasio G, Corvinelli V, Godfrey P, et al. Guided automated learning for query workload re-optimization[J/OL]. Proceedings of the VLDB Endowment, 2019, 12 : 2010-2021. http://dx.doi.org/10.14778/3352063.3352120.

[7] Marcus R, Negi P, Mao H, et al. Neo: a learned query optimizer[J/OL]. Proceedings of the VLDB Endowment, 2019, 12 : 1705-1718. http://dx.doi.org/10.14778/3342263.3342644.

[8] Trummer I, Wang J, Maram D, et al. SkinnerDB: Regret-Bounded Query Evaluation via Reinforcement Learning[J/OL], 2019. http://dx.doi.org/10.1145/3299869.3300088.

[9] Marcus R, Papaemmanouil O. Deep Reinforcement Learning for Join Order Enumeration[C/OL] // . 2018 : 1-4. http://dx.doi.org/10.1145/3211954.3211957.

[10] Astrahan M, Blasgen M, Chamberlin D, et al. System R: Relational Approach to Database Management[J/OL]. ACM Trans. Database Syst., 1976, 1 : 97-137. http://dx.doi.org/10.1145/320455.320457.

[11] Graefe G, McKenna W. The Volcano Optimizer Generator[J], 1991 : 21.

[12] Krishnan S, Yang Z, Goldberg K, et al. Learning to Optimize Join Queries With Deep Reinforcement Learning[J], 2018.

[13] Marcus R, Papaemmanouil O. Towards a Hands-Free Query Optimizer through Deep Learning[J], 2018.

[14] Bernstein A, Kiefer C, Stocker M. OptARQ: A SPARQL optimization approach based on triple pattern selectivity estimation[J], 2007.

[15] Groppe J, Groppe S, Kolbaum J. Optimization of SPARQL by using coreSPARQL.[J], 2009 : 107-112.

[16] 徐雷. SPARQL 查询优化[J]. 现代图书情报技术, 2012 : 42-48.

[17] 郑子伟, 郑建秋. 基于精英蚁群算法的 SPARQL 优化算法[J]. 控制工程, 2017, 24 : 1439-1446.

[18] Bradford A. Reinforcement Learning: An Introduction Richard S. Sutton and Andrew G. Barto[J], 2020.

[19] Krishnan S, Yang Z, Goldberg K, et al. Learning to Optimize Join Queries With Deep Reinforcement Learning[J], 2018.

[20] 张震. 集装箱码头双场桥动态调度 Q 学习算法[D]. 辽宁 : 大连理工大学, 2019.

[21] Watkins C. Learning From Delayed Rewards[J], 1989.

[22] 褚建华, 李大字. Q-learning 强化学习算法改进及其应用研究[D]. 北京 : 北京化工大学, 2009.

[23] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning[J/OL]. Nature, 2015, 518 : 529-33. http://dx.doi.org/10.1038/nature14236.

[24] Guo Y, Pan Z, Heflin J. LUBM: a benchmark for OWL knowledge base systems[J/OL]. Web Semantics: Science, Services and Agents on the World Wide Web, 2005, 3 : 158-182. http://dx.doi.org/10.1016/j.websem.2005.06.005.

哈尔滨工业大学本科毕业设计（论文）原创性声明

本人郑重声明：在哈尔滨工业大学攻读学士学位期间，所提交的毕业设计（论文）《基于人工智能的 SPARQL 查询优化》，是本人在导师指导下独立进行研究工作所取得的成果。对本文的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明，其它未注明部分不包含他人已发表或撰写过的研究成果，不存在购买、由他人代写、剽窃和伪造数据等作假行为。

本人愿为此声明承担法律责任。

作者签名：冯运　　日期：2020 年 6 月 10 日

致　谢

美好的时光总是短暂，我转眼间从四年前入学时的憧憬变成了如今离别的种种不舍。

在四年里，感谢工大，不仅让我掌握了专业知识，更是帮助我提高了自己的认知，让我看到了未来的方向。

在 2020 年这样一个独特的年份迎来了毕业季，既是不幸也是幸运。不幸的是长达半年的居家生活让我们毕业生错过了很多仪式和离别，幸运的是我们有了一个独特的毕业回忆，让我们终生难忘。感谢在疫情期间奔赴抗疫前线的白衣天使，感谢导师王宏志教授在千里之外的精心指导，感谢实验室的郑磊学姐半年以来对我毕业设计的帮助，特别感谢室友陈进龙和挚友崔航在我毕业设计最困难的时刻，能够耐心倾听我的苦诉，陪我度过那段难忘的时光。

最后，我要特别感谢我的父母，你们在这居家的半年里，每天为我做饭，在困难的时候能为我提出建议，在我怠惰的时候，能够及时督促我，在我抑郁的时候能够耐心的开导我。感谢你们一直的陪伴和支持，让我能够有今天的成就！