强化学习在资源最优调配问题中的应用

一，摘要

近年来强化学习应用广泛，在解决许多问题时都起到了非常好的效果，本文介绍强化学习在最优资源调配问题中的应用，并通过介绍最优资源调配问题的实例---打车平台的订单调配问题等来体现强化学习在其中的作用。

二，引言

2.1 资源最优调配问题

资源最优调配问题指的是通过高效管理资源来提高整体收益。资源最优调配问题体现在生活的方方面面，常见的资源最优调配问题有：打车平台的派单问题、供应管理问题、水资源供需问题等。传统上，解决此类问题常用运筹学方法，如动态规划、组合优化、图论、对策论等。传统的运筹学在解决资源最优调配问题时取得了一定的效果。然而该类方法在解决资源最优调配问题时具有局限性，原因在于：实际中的资源最优调配问题存在场景复杂、规模巨大、不确定性强这几个特点。“场景复杂”指的是实际问题中的约束，例如多平台打车问题中未满足客户需求而带来的用户流失等问题，无法用运筹学方法直接刻画，近似建模会使模型的客观性下降；“规模巨大”指的是实际的资源最优调配问题往往涉及到近千万的用户， 因此在建模时会出现节点过多、依赖复杂、约束过多等现象，造成求解困难；“不确定性强”指的是资源最优调配问题需要根据现有的策略去对未来的情况进行资源调配，因此需要建立预测模型，随着预测的时间增加，预测的精度也会逐渐降低，可能会得到低效解。[1]

2.2 强化学习

强化学习是一种机器学习方法，指智能体以“试错”的方式进行学习，通过与环境进行交互获得的奖赏指导行为，目标是使智能体获得最大奖赏。强化学习的五个要素分别是环境、智能体、状态、动作空间、奖励。强化学习大致可分为两类基于值的方法、基于策略的方法。基于值的方法常用的算法有蒙特卡洛法、TD 算法、DQN 算法、DDQN 算法和 Dueling DQN 算法。蒙特卡罗算法值利用采样若干完整的状态序列来计算并优化最优动作价值函数，并用贪婪法更新策略；TD 算法相比与蒙特卡罗算法的区别在于用结合了自举法；DQN 用神经，网络来近似表示价值函数；DDQN 会设定两个 Q 网络，用 Q 网络去拟合被固定的 n 目标 Q 网络，这样做的原因是传统的 DQN 会高估 Q 值的大小。基于策略的方法常用的算法有蒙特卡洛策略梯度算法。蒙特卡洛策略梯度算法是指对每个采样 的蒙特卡洛序列，计算序列中每个时间位置 t 的状态价值，并用梯度上升法更新策略函数的参数，循环优化更新。此外，按照是否已知状态-动作条件概率，可分为有模型方法和无模型方法。有模型的方法虽然在学习过程中效率高，但其缺陷在于依赖元素过多、复杂度高、难于建模。而无模型的方法不用了解系统的动力学策略，但其缺陷在与学习系统动力学的难度是巨大的。

强化学习的应用是十分广泛的。强化学习的发展历程可以从二十实际八十年代说起，它借鉴了动物学习的试错法，并由于最优控制方案的发展形成了现代强化学习。最优控制本是控制器优化动态系统的一种指标，Richard Bellman 提出了动态规划的方法作为最优控制的解决方法，组成了现代强化学习理论和算法的基本元素。强化学习融合了计算机科学、神经科学、心理学、经济学、数学、工程等，因此它在多个领域都有应用。例如，医疗、教育、自动驾驶、机器人、游戏等领域。强化学习在各领域应用一般要经过问题定义、数据准备、特征工程、表征选择、算法选择、实验调优、部署调优这几个步骤才能在落地时取得较好的效果。以强化学习在游戏领域为例，无论是 DeepMind 研发的 AlphaGo Zero 在围棋领域

超越人类，还是 OpenAI 研发的 Dota Five 在 Dota 游戏上达到人类顶尖玩家水平，亦或是 AlphaStar 在星际争霸游戏上取得的成果，都表明了强化学习的强大效果。以强化学习在自动驾驶领域的应用为例，强化学习可应用在路径优化、高级驾驶策略开发、控制器优化、交叉口合并与分割策略等任务中，目前国内的蔚来汽车公司与国外的特斯拉在自动驾驶的落地上取得了一定成果。[2]

多智能体强化学习已被应用于协同决策支持系统等领域。与单智能体强化学习(RL)不同，多智能体 RL 需要智能体学习与他人合作。通常不可能知道其他策略，因为所有代理的学习过程都是同步进行的。对于每个代理，环境是非平稳的。将独立的强化学习方法直接应用于多智能体环境中是有问题的。仅提出了几种方法来缓解或解决这个问题，包括共享策略参数，与其他代理的策略参数训练 q 函数，集中训练和对手建模。此外，也有一些方法使用显式通信来为对等代理建立一个相对平稳的环境。在大规模的多智能体系统中，非平稳问题将被放大。为了解决这一问题，提出了一种新的方法，通过在多智能体强化学习中应用平均场理论，将多智能体学习转换为双人随机博弈，使其在大规模场景中成为可能。由于平均 MARL 方法只考虑了状态/动作输入的平均值，因此它忽略了代理交互。我们提出的方法提供了另一种方法来实现大规模多代理学习的方法，并保留代理之间的交互，使代理从下一个时刻接收全局反馈，并及时调整其策略。[3]

三，实例

打车平台的调度问题是打车平台的派单问题和大规模车辆管理问题的统称，指的是在线打车平台通过重新分配交通资源（将订单分配给车辆）来缓解交通拥塞问题并提高平台整体的总收益。大规模在线打车平台，例如滴滴出行和 Uber，利用无线通信工具与全球定位系统将未利用的车辆分配给需要的乘客并给司机提供路线规划等信息，极大地改变了人们的出行与生活方式。[4]

打车平台的调度问题关键之处在于平衡供需，即平衡乘客的订单和司机是否可以接单。举一个简单的例子，针对打车高峰区域的乘客打不到车，而大量司机在订单稀疏区域接单的现象，可以通过系统引导司机到打车需求量大的地点，既可增加订单成交率，又可提高社会交通效率。

3.1 打车平台的派单问题

3.1.1 问题概述

打车平台的派单问题即将平台内收到订单分配给平台内的司机。传统的派单算法，例如就近派单算法、先到先服务算法，只关注用户的即时满意度，这些方法易于实施和管理，但长远来看会导致次优结果。订单调度旨在找到司机和订单之间的最佳匹配，这对按需叫车服务显然至关重要。每辆出租车配备传感和通讯工具，定期将其地理坐标和占用状况上传到平台。在另一方面，当乘客“按需”生成请求时，他或她会立即在平台上下订单。在过去，滴滴依赖的是一种简单但仍然成功的在线订单调度策略。特别是，在每个短时间段（例如 1 秒或 2 秒）内，平台的决策中心首先收集所有可用的驱动程序和活动订单，然后基于组合优化算法进行匹配。

3.1.2 解决方法

本文所介绍的方法基于 Xu 的工作[5]，在大规模的点播叫车平台上提出了一种新的订单调度算法。传统的订单调度方法通常关注即时客户满意度，而本文所提出的算法旨在提供一种更有效的方法来优化资源利用和用户体验。特别地，我们将订单调度建模为一个大规模的顺序决策问题，其中向驾驶员分配订单的决策是由一个集中式算法以协调的方式决定的。

通过学习和规划的方式解决问题：1)基于历史数据，首先将需求和供应模式总结为时空量化，每个量化都表明驾驶员处于特定状态的期望值；2)实时进行规划步骤，其中每个驾驶员顺序对考虑即时奖励和未来收益，然后使用组合优化算法解决调度。通过广泛的离线实验和在线 AB 测试，该方法显著提高了平台的效率，并已成功部署到滴滴出行的生产系统中。

首先对派单问题进行建模。将该问题建模成一个 MDP 问题，首先将地域进行网格化划分，状态是由时间和空间组成的二维拼接向量，动作空间是接单或者不接单，奖励是订单的价格，目标是最大化全局总收益。

算法分为两个部分：离线学习和在线规划。在离线学习部分，利用历史数据，用强化学习的方法（TD 算法）不断更新，求解最终的全局状态价值。在在线规划部分，将司乘匹配问题建模成二分图匹配问题，将优势函数作为边权，利用二分图匹配算法（KM 算法），进行司乘匹配。KM 算法流程如下：首先进行初始化，为所有顶点赋值，将左边的顶点赋值为最大权重，右边顶点赋值为 0；其次按序依次进行更新，只与权重相同的边匹配，若找不到边匹配，则对冲突的边左边顶点-1， 右边顶点+1，再次匹配；若依然冲突，重复上述操作。

3.1.3 实验结果

文章[5]设计了模拟器进行了模拟实验。这里使用的模拟器是为叫打平台进行物理 世界的全面建模。一个典型的示例是从历史数据中模拟特定的一天。为此，我们首先将此日期实际发生的订单作为需求返回。每个驾驶员根据其第一次进入平台的位置和时间进行初始化。驾驶员之后的行为完全由模拟器决定，要么是根据用户定义的订单调度算法（服务订单），要么是使用从历史数据拟合的模型（空闲运动和离线/在线操作）。模拟器经过仔细校准，模拟结果和真实指标之间的差异在大多数情况下，在响应率和总 GMV 方面，这个比例都在 2%以内。对全局 GMV 的改进来自于几个变化。例如，在某些城市，使用 MDP 算法，在高峰时段，对前往高价值地区的订单的响应率可以提高多达 10%。因此，更多的司机可以留下来或转移到高价值地区，从而能够在未来服务更多的潜在订单。同时，在相同 的条件下，MDP 优先考虑长途旅行的订单，因为他们的即时奖励更大；我们发现它更符合司机的意图。

表 1 用模拟器在四个城市中实验的对比结果

城市	MDP V0		MDP V 收敛		
指标	gmv	速率	gmv	速率	收敛的天数
城市 C	+0.6%	+0.5 pp	+0.8 %	+0.9p p	4
城市 D	+0.9%	+1.0 pp	+1.4 %	+1.5p p	8
城市 E	-0.1%	+0.1 pp	+0.5 %	+0.5p p	13
城市 F	+0.8%	+0.7 pp	+1.2 %	+1.1p p	7

文章[5]在真实的城市中进行了 A/B 测试。A/B 测试被广泛用于在现实系统中使用两个或多 个变体进行控制实验。传统上，在 web 分析中，可以做一个简单的流量分配策略，为两个比较的变体将用户分成 50/50 或 90/10 组。然而，由于匹配问题中的网络效应，这种设计并不适合我们的问题。想象一下，如果我们将驱动程序分成多个组，一个订单可能会连接到多个组中的驱动程序。如何为使用不同方法计算的边权值的顺序选择最佳匹配是令人困惑的。在实践中，我们采用了一个定制的 a/B 测试设计，根据大的时间切片（3 或 6 个小时）来划分流量。例如，三个小时的分割将第一天的前三个小时设置为 A 和 B 运行的三个小时。然后是订单在第 2 天反转。这样的实验将持续两周，以消除每天的差异。我们选择较大的时间切片来观察由订单调度方法产生的长期影响。

实验设置和结果。该方法已进行了多轮的 A/B 测试，覆盖了中国的多个城市。MDP 值函数是

基于上个月的历史数据，采用动态规划进行计算的。我们分别计算工作日和周末的一个值。对于超参数，我们使用贴现因子γ=0.9。其次，将所提出的 MDP 方法与基于基线距离的方法进行了比较。需要注意的是，我们没有与[24]进行比较，因为滴滴出行已经完成了从[24]传统的“驾驶员选择顺序模式”到本文研究的新的“平台分配顺序模式”的转换，该模式本身已经带来了效率的显著提高。实验结果表明，MDP 方法在所有城市的性能提高是一致的，全球 GMV 和完成率的提高在 0.5%~5%之间。与之前的发现一致，MDP 方法在高阶驱动比的城市中获得了最好的性能增益。同时，平均调度时间与基线方法几乎相同，说明用户体验牺牲不大。鉴于这些好的结果，该算法已成功部署在滴滴出行的在线调度系统中，覆盖 20 多个大城市，每天为中国数百万次出行提供服务。

3.1.4 总结

这是一种新的订单调度算法，旨在优化平台的长期效率，并满足客户的即时需求。为此，我们将订单调度建模为一个顺序决策问题，其中每个驱动器对的值被计算为一个即时奖励（订单的效用）和从历史数据中学习到的长期期望值的和。然后，以集中和协调的方式确定多个驱动程序和订单之间的匹配。在模拟器和在线 A/B 测试上的实验表明了该方法的有效性，与基于基线距离的方法相比，在总收入和订单完成率方面都有了显著的提高。基于这些结果，该方法已被部署到滴滴出行的实际调度系统中。在未来的工作中，我们感兴趣的是研究订单调度的深度强化学习方法，直接使用原始 GPS 系数作为输入，以消除区域分割的边界效应，并纳入更丰富的实时特征。在一个强大的动机下，将订单调度和司机调度连接在一起，以平衡一个统一框架内的供需。同时，这也有利于研究该方法在物流领域的其他组合优化问题中的应用，如减少能源使用、供应链管理等在运输中的应用。我们正在积极地寻求解决这些任务的潜在解决方案。

3.2 大规模车辆管理问题

3.2.1 问题概述

大规模车辆管理问题是指提前重分配可用车辆来平衡不同地区车辆需求与供应之间的差别。传统的监督学习方法很难捕捉和建模供需关系的变化。而强化学习可以通过与复杂环境交互来学习动态，因此适宜用来解决大规模车辆管理问题。然而深度强化学习在处理大规模车辆调度问题时，会存在以下几个问题：首先是问题设置的可行性。框架是由奖励驱动的，这意味着从策略中采取的一系列行动仅由来自环境的奖励信号来评估。定义代理、奖励和行动空间对 RL 至关重要。如果我们使用集中式代理对分配策略进行建模，那么操作空间可能会非常大，因为一个操 作需要决定从每个位置重新定位到其附近位置的可用车辆的数量。此外，该政策还受到可行性限制，强制执行重新定位的车辆数量不需要大于当前可用的车辆数量。据我们所知，这种高维精确约束满意度策略优化在 DRL 中是不可计算处理的：将它应用于一个非常小规模的问题可能已经导致较高的计算成本。其次，智能体规模过大。视为一个代理。多智能体配方确实减轻了动作空间维数的诅咒。然而，这样的设置每次都会创建数千个与环境交互的代理。使用 DRL 训练大量的代理再次具有挑战性：每个代理的环境都是非平稳的，因为其他代理同时也在学习和影响环境。由于计算成本高，大多数现有的研究只允许在一小部分代理之间进行协调。再者，是状态和动作空间的依赖性。促进大规模代理之间的协调 仍然是一项具有挑战性的任务。由于每个代理通常学习自己的随时间变化的策略或操作值函数，因此很难为大量代理进行协调。此外，行动空间是随着时间的推移而动态变化的，因为代理正在导航到不同的位置，而可行的行动的数量取决于该位置的地理环境。[4]

3.2.2 解决方法

在大规模车辆管理问题中，比较难解的点有如下几个：首先，智能体的决策会对其他智能体的环境产生影响，这种影响是动态的、高维且复杂的，难以直接学习。例如，在滴滴打车订单的供需调度上，如果调度过多的车辆到热门区域，将导致车辆太多而使热门区域无单可接。其次，若以单个车辆作为智能体，则智能体量级过大，可达千万的级别。再者，多个智能体的动作空间进行协同组合，若智能体的动作空间过于复杂，则会导致量级过大，难以学习。

为应用强化学习解决大规模车辆管理问题，首先对问题进行建模。

1，区域划分：将地图划分成若干个相等的六变形，则每个六边形周围有六个六边形。

2，Agent：一辆车看作一个 Agent，处于相同区域的 agent 看作一个智能体群，具有相同的决策，将从单车辆的调度问题，转换为对车辆群的调度，那么最多仅有 N 个智能体群。

3，State：网格内的特征信息（订单数、可用车辆数），网格的独热编码，时间的独热编码的合并向量。

4，Action：相邻限制，只允许智能体群往相邻六个格子移动，或者原地不动，（减少车辆空跑带来的消耗）。

5，Reward：移动到当前网格的收益 = 网格内的订单总额 / 网格内的空闲车辆数。

为解决前述问题，文章[4]的解决方法如下：首先，减少智能体数量，将对车辆的调度转换为对车辆群的调度，则同一时间最多有 N 个（网格个数）的智能体。其次，对智能体群的动作空间进行邻接限制，最多只有 7 种可能。状态转移概率收到邻接和联合的限制。所谓联合：若达能所处网格周围的 6 个格子，其 q 值不如呆在原地，那么对其截断为 0。

3.2.3 实验结果

文章[4]采用了多种不同的算法在全局 GMV 和订单响应速率的性能表现方面进行对比。对以下几种算法进行简要介绍：

- Simulation：基线算法模拟了真实场景，没有任何车队管理策略。
- Diffusion：该方法将可用车辆随机扩散到相邻网格中。
- Rule-based：该方法存储了一个 T×N 的 V 值表格。
- Value-based：该方法基于策略评估动态更新 V 值表格。
- T-Q learning：标准化的独立 Q-learnig 学习。
- T-SARSA：独立的 sarsa 学习算法。
- cDQN：具有邻接和联合限制的 DQN 算法。
- cA2C：具有邻接和联合限制的 A2C 算法。

表 2 不同方法在 GMV 和订单响应速率方面的表现

	100% 初始车辆		90% 初始车辆		10% 初始车辆	
	标准化的 GMV	订单响应速率	标准化的 GMV	订单响应速率	标准化的 GMV	订单响应速率
Simulation	100.00 ±0.60	81.80% ±0.37%	98.81 ±0.50	80.64% ±0.37%	92.78 ±0.79	70.29% ±0.64%
Diffusion	105.68 ±0.64	86.48% ±0.54%	104.44 ±0.57	84.93% ±0.49%	99.00 ±0.51	74.51% ±0.28%
Rule-based	108.49 ±0.40	90.19% ±0.33%	107.38 ±0.55	88.70% ±0.48%	100.08 ±0.50	75.58% ±0.36%
Value-Iter	110.29 ±0.70	90.14% ±0.62%	109.50 ±0.68	89.59% ±0.69%	102.60 ±0.61	77.17% ±0.53%
T-Q learning	108.78 ±0.51	90.06% ±0.38%	107.71 ±0.42	89.11% ±0.42%	100.07 ±0.55	75.57% ±0.40%
T-SARSA	109.12 ±0.49	90.18% ±0.38%	107.69 ±0.49	88.68% ±0.42%	99.83 ±0.50	75.40% ±0.44%
DQN	114.06 ±0.66	93.01% ±0.20%	113.19 ±0.60	91.99% ±0.30%	103.80 ±0.96	77.03% ±0.23%
cDQN	115.19 ±0.46	94.77% ±0.32%	**114.29** ±0.66	94.00% ±0.53%	105.29 ±0.70	79.28% ±0.58%
cA2C	**115.27** ±0.70	94.99% ±0.48%	113.85 ±0.69	93.99% ±0.47%	**105.62** ±0.66	79.57% ±0.51%

3.2.4 分析总结

在文章[4]中，我们首先制定了大规模的舰队管理模型，将问题转化为一个可行的深度强化学习的设置。在此基础上，我们提出了上下文多智能体强化学习框架---cDQN，并开发了 cA2C，两者都实现了大规模代理在车队管理问题上的协 调。通过利用集中化价值网络和嵌入上下文信息的分散化策略执 行，CA2C 具有灵活性和效率。它能够适应不同的动作空间 端到端培训范式。利用滴滴出行提供的真实数据，开发并校准了 一个模拟器，作为我们的培训和评估平台。在模拟器不同设置下的广泛实证研究证明了所提出的框架的有效性。

四，总结

强化学习在资源最优调配问题中取得了很多重要突破，但人有许多问题有待解决，例如在对资源最优调配问题建模并用强化学习算法求解时，需要大量历史数据用于训练，这对于小规模应用场景来说门槛较高；其次，训练强化学习算法需要强大的计算资源，且由于现实问题动态变化，需要时常更新模拟器，这意味着巨大计算成本。[1]

参考文献

【1】WANG J Y, WEI X R, SHI W L, et al. Applications of reinforcement learning in the field of resource optimization[J].BigDataResearch,2021,7(5): 138-149.

【2】Mondal A K, Jamali N. A survey of reinforcement learning techniques: strategies, recent development, and future directions[J]. arXiv preprint arXiv:2001.06921, 2020.

【3】Zhou M, Jin J, Zhang W, et al. Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching[C]//Proceedings of the 28th ACM International Conference on Information and Knowledge Management. 2019: 2645-2653.

【4】Lin K, Zhao R, Xu Z, et al. Efficient large-scale fleet management via multi-agent deep reinforcement learning[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 1774-1783.

【5】Xu Z, Li Z, Guan Q, et al. Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach[C]//Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018: 905-913.