充电站推荐的多阶段MDP模型

摘要

为了最大限度地缩短车辆的总充电时间和平衡充电站的负载，越来越需要充电站推荐。为了满足这一需求，我们将推荐问题建模为马尔可夫决策过程(MDP)问题。然而，传统的MDP模型存在“维度诅咒”的问题。针对这一问题，我们提出了MDP的一种扩展：多阶段MDP，将MDP的状态转换分解为多个阶段，以减少状态空间和状态转换的复杂度。这是通过引入MDP中定义的正常状态以外的两种状态来实现的：后决策状态和中间决策状态。在此基础上，提出了一种基于在线学习的多阶段MDP模型求解算法。由于减少了状态空间和状态转移的复杂度，该在线算法能够快速收敛。通过与基于博弈论的推荐机制和基于Q-学习的推荐机制的比较，仿真结果表明本文提出的推荐方案具有良好的性能。

1. 引言

电动汽车的普及率不断提高，使得电动汽车充电成为一个关键问题。安装在家里的充电桩只能解决部分问题，由于充电时间长，车辆无法到达，如电动出租车、Uber电动汽车。因此，越来越多的支持电动汽车加油的公共基础设施开始实施并商业化。然而，随着电动汽车普及率的不断提高，充电需求与现有充电基础设施容量之间的差距始终存在。

已经确定了与充电问题相关的几个挑战，包括充电事件的路由优化[1]、范围估计[2]、充电站位置[3]、充电调度[4]和充电站（CS）建议[5]等。在这些挑战中，充电调度和充电建议是两个重要的挑战。充电计划是为电动汽车安排合适的充电时间段，而充电建议是在电动汽车请求充电时向其推荐最佳CS。这两个挑战在某种程度上是相似的，因此在文献中，收费调度被分类为时间调度，而收费建议被分类为空间调度[6]。

充电调度的主要目标是减轻电动汽车对电网的影响，同时保持可接受的用户满意度，而CS建议的目标是最小化包括驾驶时间、排队时间和充电时间在内的总充电时间。从用户的角度来看，收费的体验质量（QoE）是主要关注点。因此，推荐具有最佳CS的电动汽车以通过最小化总充电时间来改善用户QoE至关重要。此外，CS建议或多或少会影响其他问题。例如，它影响电动汽车的布线优化[1]和CS布局[7]。

为了提高CS推荐的性能，目前的工作主要集中在两个方面。一个是设计一个建议架构/方案，帮助电动汽车选择CS并完成充电[8]，[9]，另一个是建议建模。在我们的工作中，我们重点研究了推荐模型，它对决定CS推荐的性能起着至关重要的作用。

由于CS推荐可以看作是一个分配问题（将一个EV分配给一个CS），因此相关的建模方法可以用于CS推荐。事实上，智能交通系统的许多问题，如出租车调度和停车诱导等，也是一个分配问题。因此，在这些领域提出的模型/算法可以纳入CS建议，如出租车调度的多重优化模型[10]、[11]和停车诱导的二次分配问题模型[12]。目前，动态建模引起了该领域的广泛关注，如基于博弈论的建模[6]、[13]、动态资源分配[14]等。

尽管这些工作对CS推荐模型的建模做出了很大的贡献，但在建模复杂度和效率方面仍存在一些不足。静态优化模型，如[15]中使用的线性规划(LP)和[16]中使用的混合整数规划(IP)，可以实现全局（或近全局）优化（例如，最小化收费时间或最大化收入），但不适用于推荐场景。原因之一是这些建模不能捕捉CS建议的动态特性。二是它们通常需要大量的计算，因此会产生大量的延迟。动态建模，如[13]中使用的博弈论，确实反映了推荐的动态特征。然而，以往的动态建模要么实现了一步优化，而没有对未来的预测，要么无法进行智能推荐。这促使我们设计一个基于扩展马尔可夫决策过程(MDP)模型的动态智能CS推荐系统。

在我们的CS推荐系统中，当收到电动汽车的充电请求时，系统需要决定电动汽车应该被引导到哪个CS。该决策是基于系统的当前状态做出的，其中包括例如CSS的排队信息和距离信息。一个最优决策必然会使总的充电时间最小化，总的充电时间包括开车时间、排队时间和充电时间。对于每一个决定，我们让系统获得一个与总收费时间成反比的奖励。同时，系统演化到下一个状态。因此，整个过程可以被建模为一个MDP问题，其中的关键成分是一组决策环境、系统状态、可用行动、奖励和依赖于状态/行动的转移概率[17]。

根据以上分析，MDP模型很好地抓住了CS推荐的特点。然而，MDP有“维度诅咒”[18]的问题，在CS推荐方案中，这一问题变得更加严重，因为存在较大的状态/动作空间。此外，由于MDP建模需要对模型的所有概率分布有先验知识，这在实际中是不可能的。为了解决这些问题，我们引入了决策后状态(PDS)[19]，并在模型中引入了一种新的状态，称为中间决策状态(IDS)，我们称之为多阶段MDP(MMDP)模型。利用这两个附加状态，将MDP的状态转换分解为多个阶段，从而大大降低了状态空间和状态转换的复杂性。在此基础上，提出了一种基于在线学习的MMDP模型求解算法。

本文的主要贡献如下：

- 提出一种新的MDP模型——MMDP。在该模型中，除了MDP中定义的正常状态外，还引入了两种附加状态PDS和IDS。
- 将CS推荐问题描述为一个MMDP模型。与传统的MDP模型相比，MMDP在状态空间和状态转移复杂度方面都有明显的降低。这使得我们的命题对于存在“维数诅咒”问题的CS建议是有效的。
- 提出了一种基于在线学习的MMDP模型求解算法。理论分析和仿真评估都表明，基于在线学习的算法收敛速度快，在总体充电时间和CSs负载均衡方面具有良好的充电性能。

本文的其余部分组织如下。第二节简要总结了相关工作。第三节描述了使用MDP的电动汽车推荐问题。第四节通过介绍PDS和IDS来阐述我们的MMDP模型。第五节介绍了求解MMDP模型的在线学习算法。第六节分析了本文命题的趋同性。第七节通过仿真对性能进行了评价。第八节总结了本文的工作并提出了展望。

2. 相关工作

CS布局（部署）影响调度和再指挥性能，已被广泛研究。可以充分利用现有加油站进行CS建设[20]，也可以寻求新的CS设置方案[21]。CS布局问题可以看作是一个在充电需求、续航里程等多个约束条件下的优化问题，并被断言为NP完全问题[3]、[21]。为了解决这个优化问题，Albert Y.S.林等人。[3]提出了迭代混合整数线性规划(MILP)和贪婪法等解决方案，以最小化建设成本。[21]设计了两个基于贪婪的多目标同时优化算法。CS的部署也可以被认为是一个博弈问题，Luo等人。[22]将其表述为一个贝叶斯博弈，旨在实现CS所有者利润、用户满意度和电网可靠性之间的平衡。

充电调度解决了电动汽车在CS或公园停车接受检查时应该何时充电的问题。适当的电动汽车充电调度可以维持电网的稳定，保证电力供需平衡[23]。Mukherjee和Gupta[4]将这些工作总结为单向充电和双向充电两大类。前者是指电网到车辆(G2V)，电动汽车只从电网充电；后者是指车辆到电网(V2G)，电力流动是双向的。

将以往的调度工作分为车站/停车级调度和网格级调度两大类。对于车站/停车层，调度在每个CS或停车处单独执行。在[24]中，为了使平均排队长度（即充电时间）最小化，一个CS确定排队中待服务的电动汽车的数量和待分配用于充电的可再生能源的数量。作者将其建模为约束MDP问题。在文献[25]中，作者设计了一个最优定价方案来指导和协调电动汽车在CS中的充电过程。该问题被建模为一个凸优化问题，目标是最小化服务丢失率。在[26]中，作者将调度问题建模为多模式近似动态规划(MM-ADP)，目标是最小化收费费用。在[27]中讨论了停车库下的调度问题。在实现运营商利润最大化的同时，为计费用户提供满意的服务。

网格级调度的目标是实现全局优化，而不是单个控制系统的优化。在文献[28]中，作者在考虑电网约束的情况下，设计了一个使发电机运行费用最小的最优计费控制计划。另一方面，收费成本最小化被认为是一个目标[29]，[30]。文[30]将充电调度问题描述为一个约束MDP问题，其目的是在保证电动汽车充满电的前提下，使充电费用最小化。针对该模型，提出了一种基于安全深度强化学习(SDRL)的无模型求解方法。在[31]中，作者的目标是协调电动汽车调度以缓解有功功率波动。

CS推荐是决定用户体验质量(QoE)的重要指标，近年来受到了广泛的关注。本文从推荐系统/方案设计和推荐建模两个方面对前人的工作进行了综述。

文献[32]提出了一个由EVs、CSs和全局聚合器(GA)组成的推荐系统，其中GA是一个集中的实体，它利用CSs的条件信息和EVs的计费预约进行CS推荐。本文在文[8]中进一步扩展了这一工作，提出了一种由电动汽车、移动边缘计算(MEC)和云计算(CC)组成的电动汽车充电分层体系结构。文[5]研究了电动汽车出租车CS推荐系统。该系统包括三个核心模块：状态推理、等待时间计算和CS推荐。该系统根据电动出租车的充值意图，计算每个CS的等待时间，并为出租车选择最佳的等待时间。

在[9]中，作者定义了一个两级推荐方案，其中较低的一级是充电率控制，它处理如何将CS的电源分配给电动汽车，而较高的一级是CS推荐，它分配了一个最佳站点。[33]中提出了一种充电管理方案，可为异构电动汽车提供抢占式充电服务。

推荐模型在确定总体充电性能方面起着至关重要的作用，这也是本文研究的重点。以前的工作将CS推荐问题描述为博弈论、LP问题、广义指派问题等。

在文献[6]中，推荐问题被建模为一个博弈论，其中每个EV都是一个参与者，在本文中使用了时间触发策略。即在每个预定义的时间瞬间，所有有充电需求的电动汽车选择一个能够最小化循环行驶和排队时间的CS。在此过程中，如果一个EV（如EV1)选择了一个使前一个策略（如EV2)变得非最优的策略，那么EV2必须重新选择它的策略。利用归纳法证明了该博弈存在纳什均衡。文献[13]的作者将推荐问题建模为一个Stackelberg博弈，考虑了收费价格、排队时间和CSS之间的距离。然而，博弈模型并不能保证获得推荐的最优解。

在文献[14]中，作者将CS推荐问题看作一个在线动态资源分配问题，并将其转化为一个具有公平性约束的Pareto优化问题。然后采用基于贪心的算法对模型进行求解。

LP（或IP）也用于CS推荐。Ni等人[15]研究了CS关于电池交换的建议。为了最大限度地提高所有CSs的总收入，推荐系统为用户推荐CSs列表，而不仅仅是一个CS，然后让用户选择一个。这样，可以减少推荐失败。然后将该推荐问题表述为LP问题，其中一些充满电的电池策略性地保留给能够接受更高价格的

未来客户。类似地，Tan等人[16]也在研究电池交换–ping CS，但他们将其建模为一个混合整数线性规划（MILP），并开发了一个广义benders decom–position算法来求解该模型。

如上所述，CS推荐问题可以看作是一个指派问题。孔等人。[9]将其建模为一个广义指派问题。然后，他们提出了两个近似算法。一种算法是具有多项式复杂度的3–逼近，另一种算法是使用完全多项式时间逼近格式的(2+)逼近。文献[34]将车辆–CS分配问题建模为MILP。作者提出了一种代理辅助优化方法来求解所建立的模型。

由于推荐问题和调度问题紧密联系在一起，以往的工作也同时解决了这两个问题。在[35]中，提出的系统对每个收费请求进行重新计算和调度，以使总收入最大化。将该问题转化为一个优化问题，并采用一种称为投标–价格控制的启发式方法求解该模型。在[36]中，作者的目标是同时优化调度（充电时间）和推荐。他们通过最小化充电成本来安排电动汽车的充电时间，并通过最小化充电时间来推荐CS。[37]的作者使用博弈论方法设计了多组公共电动汽车竞争容量有限的充电站的时空调度。

由于推荐问题和调度问题紧密联系在一起，以往的工作也同时解决了这两个问题。在[35]中，提出的系统对每个收费请求进行重新计算和调度，以使总收入最大化。将该问题转化为一个优化问题，并采用一种称为投标–价格控制的启发式方法求解该模型。在[36]中，作者的目标是同时优化调度（充电时间）和推荐。他们通过最小化充电成本来安排电动汽车的充电时间，并通过最小化充电时间来推荐CS。[37]的作者使用博弈论方法设计了多组公共电动汽车竞争容量有限的充电站的时空调度。

3. 问题描述

我们考虑的EV充电基础设施的控制和管理的推荐系统。推荐系统的目的是通过对每个计费请求进行最佳推荐，使总计费时间最小化，并实现CSs之间的负载平衡。我们将电动汽车充电过程描述如下（图1）。

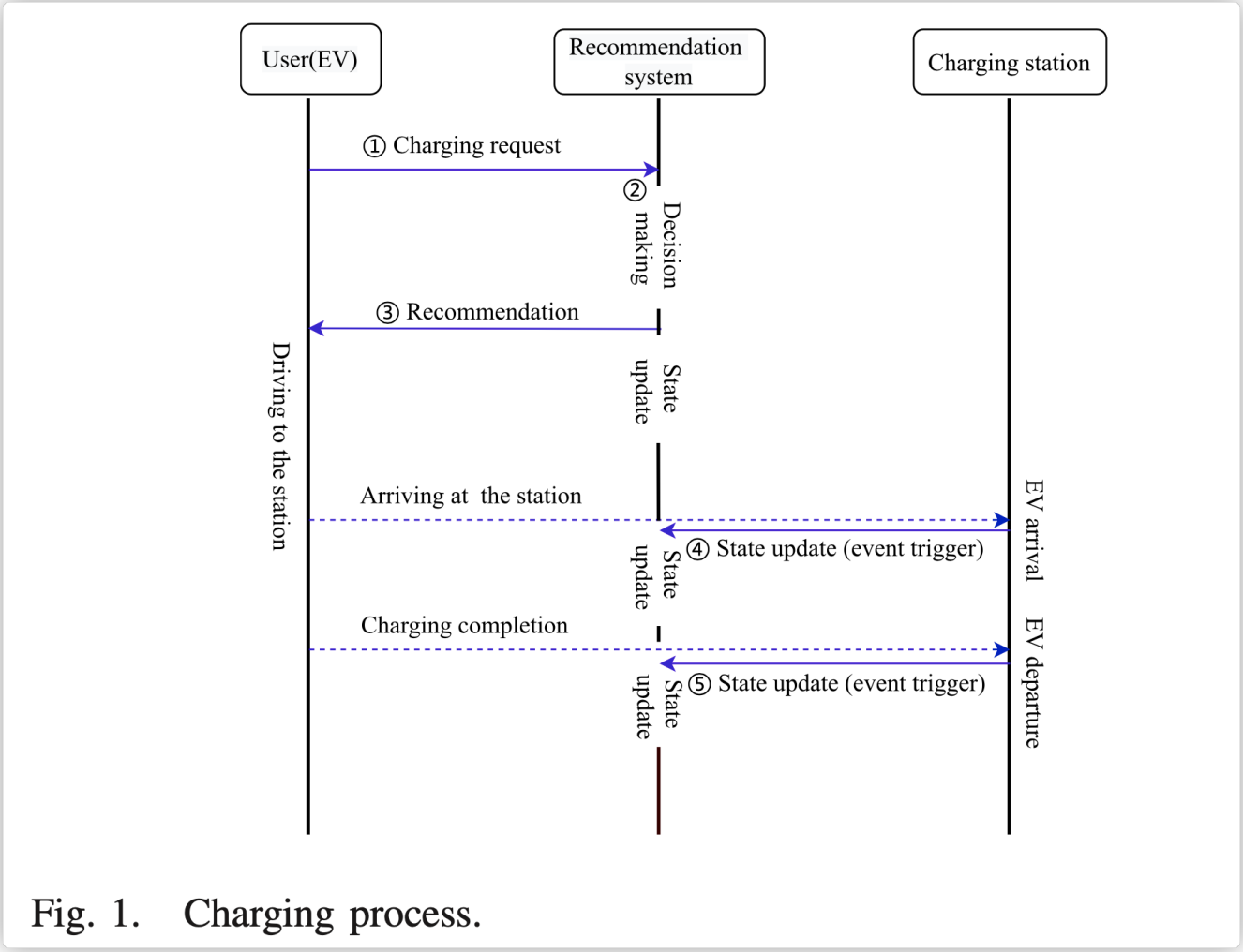


Fig. 1. Charging process.

- 当EV耗尽其能量时，驾驶员通过例如APP向推荐系统发送充电请求；
- 在接收到该请求时，推荐系统根据系统的状态做出决定（选择CS）（状态定义请参阅第三节）；
- 推荐系统向请求EV回复推荐消息并更新状态；
- 当电动汽车到达推荐的CS时，CS通知推荐系统，由推荐系统更新状态；
- 当电动汽车完成充电时，CS再次通知推荐系统更新状态。

用Voronoi图划分整个充电区Z（图2）。对于CS集$\{c_1, c_2, \ldots, c_{n-1}, c_n\}$，其中$n$是区域中CS的数目。$c_i$站的区域$Ri$定义为：

$$R_i = \{l \in Z | d(l, c_i) < d(l, c_j), j \leq n, j \neq i\}$$

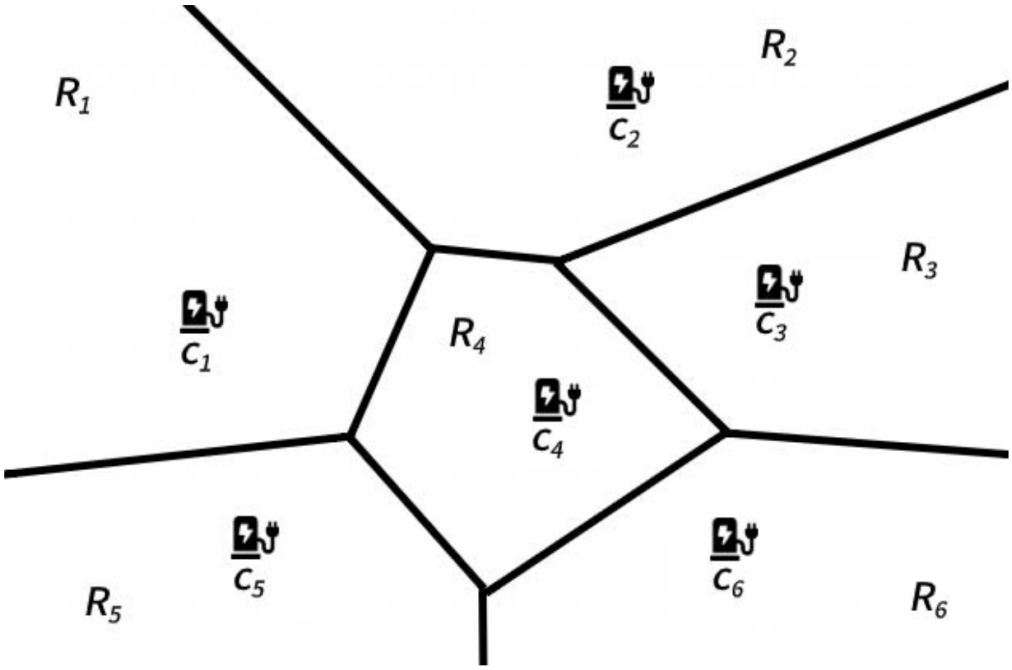


Fig. 2. Voronoi diagram.

与以往的推荐通常在固定的时间间隔（时间触发器）触发不同，我们的推荐是由计费请求（事件触发器）触发的。显然，事件触发机制比时间触发机制有优势，因为它可以立即接收推荐，而 后者应该等待触发时间。在我们的提议中，时间范围由EV的充电请求来划分，如图3所示：

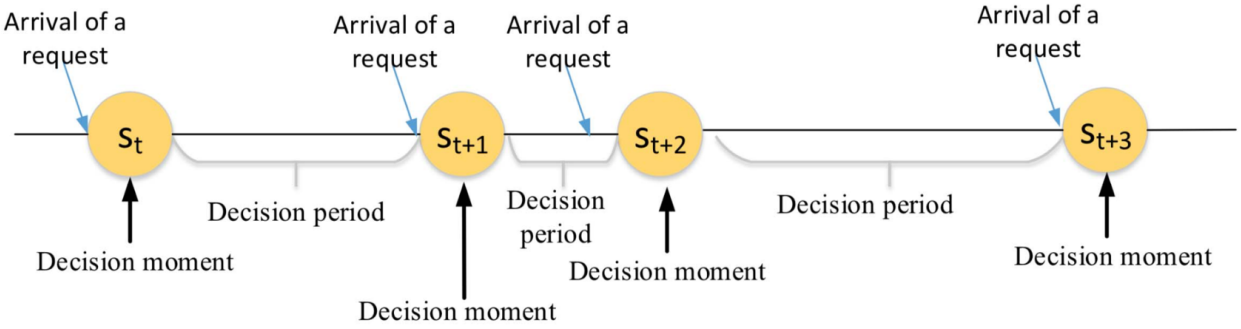


Fig. 3. Decision periods.

充电请求到达的随机性使得决策周期（两个连续到达之间的间隔）发生变化。在每个决策时刻t，系统接收充电请求，并对CS建议做出决策，即请求电动汽车应驱动至哪个CS。在收到此建议后，车辆将花费τ_d的行驶时间与τ_w的排队时间（如果有可用的充电桩，$\tau_w = 0$）和τ_c的充电时间。

CS推荐（决策）是根据系统的状态信息做出的。它包括**CS容量**、**排队长度信息**、**请求电动汽车的信息**、**先前推荐但仍在路上的电动汽车的信息（到车站电动汽车）**。我们将系统状态定义为$S \triangleq (M,\ X,\ V)$，其中M表示CS的状态，X表示到站EVs的状态，V表示请求EV的状态。

- $M = (m_1, m_2, \ldots, m_n)$是一个n元组，其元素$m_i$为整数，$m_i \in [-k_i, \infty)$，$i \in [1, n]$。其中$k_i$是CS处的充电桩数。$M = 0$表示没有等待充电的电动汽车，但所有的充电桩都被占用了；$M < 0$表示有M个充电桩可用；$M > 0$表示（队列中）等待充电的电动汽车数。

- $X = (x_1, x_2, \ldots, x_n)$也是一个n元组，它的元素$x_i \in [0, \infty)$表示已经分配给CS i 但仍在路上的EVs的个数。

- $V = (l, e)$是一个二元组，其中l表示电动汽车请求充电时的位置；e代表需要充电的电量。我们不使用坐标，而是设置代表归属区域R_i的位置信息$l \in [1, n]$。

在时刻t收到充电请求后，系统应根据其当前状态s_t，$s_t \in S$作出决定。 即采取应向请求车辆推荐CS的行动。因此，我们定义了动作集$A \triangleq [1, n]$和动作$a \in A$。$a = i$表示向请求EV推荐CS i。在采取动作$a = i$之后，系统以概率$p(s_{t+1}|s_t, a)$移动到下一个状态s_{t+1}。系统将获得奖励$r_a(s_t)$，其定义为总充电时间的倒数，总充电时间是从车辆请求充电到完成充电的时间间隔。

$$r_a(s_t) = \frac{1}{\tau_d + \tau_\omega + \tau_c}$$

MDP中的策略是一个映射$\pi : S \to A$。如果使用策略π，并且状态S是初始状态，则状态值函数$V^\pi(s)$是期望的总收益。

$$v^\pi(s) = E_s^\pi\{\sum_{t=1}^{\infty} \lambda^{t-1} r_a(s_t)\}, s_t \in S, a_t \in \mathcal{A} \qquad (1)$$

其中$s_1 = s$，$\lambda(< 1)$是一个不变的折扣因子。设$v^*(s)$表示在最优策略下对状态s的奖励，理论上可以通过递归求解下面的Bellman方程得到它：

$$v^*(s) = \max_{a \in \mathbb{A}}\{r_a(s) + \lambda \sum_{s'} p(s'|s, a) v^*(s')\} \qquad (2)$$

其中s'是s的下一个状态。

然而，在我们的场景中求解上述Bellman（布尔曼）方程具有挑战性。挑战来自三个方面。首先确定$r_a(s)$，它依赖于行车时间、排队时间和充电时间。即使我们可以假设驾驶时间和充电时间遵循一个I.I.D（独立同分布），计算电动汽车充电完成时间的期望是非常复杂的，因为电动汽车的排队时间受到其他电动汽车的影响，这些电动汽车也是以前推荐给本CS的，但可能在任何时候到达，即在本电动汽车之前或之后。

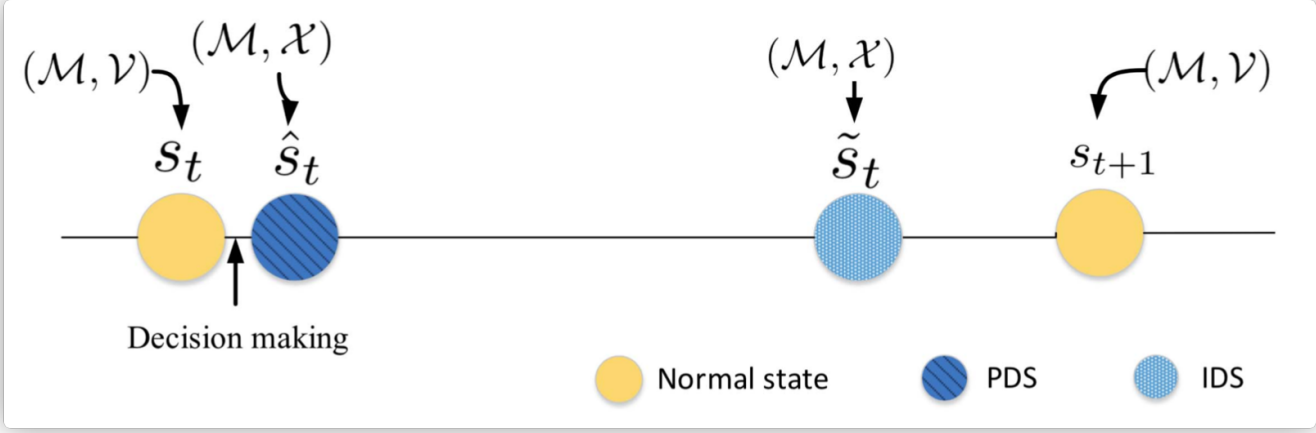
第二个是确定下一个状态。充电请求的到达和电动汽车的到达/离开站点的随机过程使得下一个状态（如s_t+1）非常不确定。例如，在t+1时刻，先前的车辆（需要在t、t−1,...充电）可能仍然在路上，或者已经在排队，或者正在充电，或者已经完成充电。因此，下一个状态的确定具有挑战性。

三是系统状态空间大。系统状态由（2n+2）个元素组成。如果n较大，并且充电能量元e（连续值）以细粒度方式离散，则状态空间非常大。

为了解决这些问题，我们提出的关键思想是将决策周期划分为多个阶段，以简化状态转换和状态空间复杂度。为此，我们使用PDS并在MDP中引入一个名为IDS的新状态来形成MMDP。在决策周期中，即从一个决策时刻到下一个决策时刻，系统从正常状态过渡到PDS，然后过渡到IDS，最后过渡到下一个正常状态。针对MMDP模型的求解问题，提出了一种在线学习算法。

4. 系统模型

一个决策周期内的三个阶段在我们的提议中被定义，如图4所示：



第一阶段介于正常状态s和PDS之间（我们用$\hat{s}$表示PDS）；第二阶段在PDS和IDS之间（我们使用$\tilde{s}$表示IDS）；第三阶段是IDS和下一个正常状态之间的阶段。为了降低状态空间的复杂性，我们让不同的状态取$(M,\ X,\ V)$的子集。正常状态重新定义为$\mathbb{S} \triangleq (M,V)$。这里，我们省略了到站电动汽车的状态，但让它在PDS和IDS中考虑。PDS和IDS定义相同，即$\hat{S} \triangleq (M,\ X)$，$\tilde{S} \triangleq (M,\ X)$。

在我们的模型中，PDS是决策后的即时状态。例如，在三个CSs(c1，c2，c3)的场景中，我们假设当前到站的电动汽车信息为$x_t = (3,2,0)$，表示分别有3，2和0辆电动汽车在前往c1、c2和c3的途中。决策时刻的正常状态为$s_t = ((2,3,3),(2,5))$。根据该状态，系统向当前充电请求推荐c3。根据该判决，PDS被设置为$\hat{s} = ((2,3,3),(3,2,1))$。根据这个定义，我们省略了关于请求EV $\mathcal{V}$的状态信息，

因为这些信息在做出决定后是无用的。通过引入PDS，可以在第一阶段限制作用对状态的影响。Bellman方程（2）改写为：

$$v^*(s) = \max_{a \in \mathcal{A}} \{r_a(s) + \lambda v^*(\hat{s})\} \qquad (3)$$

在上面的方程中，由于定义的PDS是一个动作的确定性状态，所以概率被移除了。

与EV请求触发的正常状态不同，IDS是由EV到达或离开CS触发的。在上面的两个CS场景中，假设一辆电动汽车（前面已经推荐过）到达c1。此时的状态是$\hat{s}_t^1 = ((3,3,3),(2,2,1))$。然后，在c2处的电动汽车完成充电并离开，因此$\tilde{s}_t^2 = ((3,2,3),(2,2,1))$。然后，一个新的充电请求（3,4）到达，系统进入正常状态$s_{t+1} = ((3,2,3),(3,4))$。

在我们的MMDP模型中，在每个决策阶段，我们只考虑下一个决策时刻之前的最后一个中间状态。因此，在上面的例子中，模型中只考虑了$\tilde{s}_t^2$并将其视为IDS。从PDS $\hat{s}$到IDS $\tilde{s}$的转换与动作无关。实际上，它取决于CSs和到站电动汽车的状态信息。这就是为什么两种状态都被定义为这些信息的组合。然后，我们定义PDS的价值函数如下：

$$v^*(\hat{s}) = \sum_{\tilde{s}} p(\tilde{s}|\hat{s}) v^*(\tilde{s}) \forall \tilde{s} \qquad (4)$$

其中，$p(\tilde{s}|\hat{s})$是从状态PDS到状态IDS的转移概率，它只取决于电动汽车到达和离开一个车站。因此，在第二阶段，电动汽车的到达和离开对状态的影响是有限的。

最后，通过充电请求触发从IDS $\tilde{s}_t$到下一个正常状态s_{t+1}的转换。此转换保持CS的状态信息不变，即$s.m_i = \tilde{s}.m_i \forall i$。然后，PDS的价值函数定义为：

$$v^*(\tilde{s}) = \sum_{s} p(s|\tilde{s}) v^*(s)$$

$$s = \{x \in \mathbb{S} | x.m_1 = \tilde{s}.m_1, \ldots, x.m_n = \tilde{s}.m_n\} \qquad (5)$$

其中$p(\tilde{s}|\hat{s})$是从状态IDS到正常状态的转移概率，它只依赖于计费请求的到达。因此，在第三阶段，收费请求的到达对状态的影响是有限的。

在CS推荐系统中，状态信息的变化可以由事件（例如EV的到达）触发，并且是可观察的，这使得可以将MDP状态的转变分成多个阶段。MMDP模型的一个好处是降低了复杂性，因为所有类型的已定义状态$(\mathbb{S}, \hat{\mathbb{S}}, \tilde{\mathbb{S}})$都是原始状态$(\mathcal{S})$的子集。将简化的正常状态$((M,X,V))$简化为$(M,V))$，可能会降低推荐的准确性。然而，公式化的Bellman方程（3）的右侧包括所有状态信息，即CSs的状态信息、请求EV的状态信息和到达站EV的状态信息。有了所有这些必要的信息，推荐系统就有可能做出最优推荐。通过将决策周期划分为三个阶段，将原Bellman方程（2）分解为三个方程(3)、(4)和（5）。与Bellman方程（2）相比，在Bellman方程（2）中，执行最大化需要知道概率p(ss，a)，而这几乎是不可能获得的，我们的命题消除了最大化操作中的转移概率（Bellman方程(3)），并简化了转移概率（（4）和（5））。利用这个多阶段模型，我们可以设计一个高效的在线算法来获得最优解。

5. 在线算法

在我们的建议中，系统应该了解实时状态信息。当事件触发时，CSs和EV会对此进行更新，例如，EV到达/离开CS或充电请求到达。例如，当EV到达CS时，CS向系统发送此事件。后者更新到站点EV的状态信息和CS的排队信息。

通过将状态转移分解为三个阶段，我们提出的MMDP模型可以通过基于在线学习的算法简单而精确地求解。反过来，在线算法不需要先验的转移概率知识，很好地符合CS推荐的MDP模型。

5.1 基于在线学习的算法

基于在线学习的算法使系统能够迭代更新每个状态的值函数，并最终收敛到最优解。首先，我们将（3）改写为迭代形式：

$$v_{t+1}(s) = \max_{a \in \mathbb{A}} \{ r_{t+1}(s,a) + \lambda v_t(\hat{s}) \} \qquad (6)$$

然后，我们需要从（4）和（5）中删除转移概率，这是一个不重要的工作，因为概率已经移到最大运算之外。

设ρ_t为具有以下属性的正更新序列：

$$\sum_{t=0}^{\infty} \rho_t = \infty; \quad \sum_{t=0}^{\infty} (\rho_t)^2 < \infty \qquad (7)$$

然后，根据随机逼近理论[38]，PDS $\hat{s}$（4）的值函数可更新为：

$$v_{t+1}(\hat{s}) = (1-\rho_t) v_t(\hat{s}) + \rho_t v_{t+1}(\tilde{s}) \forall \tilde{s} \qquad (8)$$

IDS $\tilde{s}$ (5) 的值函数可以更新为：

$$v_{t+1}(\tilde{s}) = (1-\rho_t) v_t(\tilde{s}) + \rho_t v_{t+1}(s) \forall s \qquad (9)$$

算法1描述了CS推荐的在线算法。在每个决策时刻i，系统观察当前状态s_i（步骤4），并为此状态选择最佳动作（步骤5）。基于所选择的动作，系统可以确定PDS $\hat{s_i}$（步骤6）。之后，系统继续观察系统状态，直到下一个决策时刻，并记录IDS $\tilde{s_i}$（步骤7，提醒IDS是下一个决策时刻之前的最后一个中间状态）。在此决策期间，如果系统接收到任何充电终止信息，它将计算总充电时间并更新相关即时奖励$r_a(s)$（步骤8）。使用在线算法，我们在第三节中讨论$r_a(s)$的确定的挑战与简单的解决方法。

Algorithm 1 Online Learning Based Algorithm

1: Initialization: set $t = 0$, randomly set $v(s)\ \forall s$, $v(\hat{s})\ \forall \hat{s}$, $v(\tilde{s})\ \forall \tilde{s}$, $r_a(s)\ \forall (s, a)$
2: **while** TRUE **do**
3: $i = i + 1$
4: Observe current state s_i
5: Determine the optimal action for s_i

$$a^* = \arg max_{a \in \mathbb{A}} \{ r_a(s_i) + \lambda v^*(\hat{s}) \}$$

6: Determine the PDS $\hat{s}_i$ based on a^* and the rule described in section IV
7: Observe IDS $\tilde{s}_i$
8: If any EV finishes charging, the system updates the related $r_a(s)$
9: Update $v(s_i)$ according to (6)
10: Update $v(\tilde{s}_{i-1})$ according to (9)
11: Update $v(\hat{s}_{i-1})$ according to (8)
12: **end while**

最后，系统应该分别使用（3）、（9）和（8）更新$\boldsymbol{s}$，$\hat{\boldsymbol{s}}$和$\tilde{\boldsymbol{s}}$的值函数。注意，在迭代$\boldsymbol{i}$中，我们得到$\boldsymbol{v(s_i)}$（步骤9）。该值用于更新的$\boldsymbol{s_i}$前一个IDS $\boldsymbol{\tilde{s}_{i-1}}$（步骤10），并且类似于PDS $\boldsymbol{\tilde{s}_{i-1}}$（步骤11）。在迭代$\boldsymbol{i}$中确定的IDS和PDS的状态将在迭代$\boldsymbol{i+1}$中更新。

如前所述，用原有的MDP模型很难获得状态转移概率。特别是对于我们的事件触发MDP模型，转移概率可以是连续的。然而，通过分解状态转移，可以使用在线学习算法来评估状态值函数，而无需计算转移概率。

有两种方法来实施拟议的系统。一种是利用历史数据或模拟数据对系统进行训练，然后在现场部署。另一种是直接部署在线算法。但是，可以使用一些更有效的策略来代替一开始的随机推荐，例如，当系统还没有学习到任何东西时，可以向电动汽车推荐最近的CS（或最欠载的CS）。