文章编号:1671-6833(2023)05-0017-07

基于深度强化学习的大规模敏捷软件项目调度

申晓宁[1,2,3,4], 毛鸣健[1], 沈如一[1], 宋丽妍[5]

(1.南京信息工程大学 自动化学院,江苏 南京 210044;2.南京信息工程大学 江苏省大气环境与装备技术协同创新中心,江苏 南京 210044;3.南京信息工程大学 江苏省大数据分析技术重点实验室,江苏 南京 210044;4.江苏省气象能源利用与控制工程技术研究中心,江苏 南京 210044;5.南方科技大学 广东省类脑智能计算重点实验室,广东深圳 518055)

摘 要:为解决大规模敏捷软件项目调度问题,首先,将其分解为故事选择、故事分配和任务分配 3 个强耦合子问题,并引入用户故事的新增与删除、每个冲刺阶段中员工工作时长的变化等动态事件,考虑团队开发速度、任务时长和技能等约束,以最大化项目所完成用户故事总价值为目标建立大规模敏捷软件项目调度数学模型;其次,根据问题特征设计了马尔可夫决策过程,采用 10 个状态特征描述每个冲刺阶段开始时的敏捷调度环境,12 个复合调度规则作为智能体的候选动作,并按照调度模型的目标函数定义奖励;最后,提出一种基于复合调度规则的优先经验回放双重深度 Q 网络算法来求解所建模型,引入双重深度 Q 网络(DDQN)策略和优先经验回放策略,避免深度 Q 网络的过估计问题,并提高经验回放池中轨迹信息的利用效率。为了验证所提算法的有效性,在 6 个大规模敏捷软件项目调度算例中进行了实验,分析了所提算法的收敛性。根据算法性能测度,与已有代表性算法 DQN、双重深度 Q 网络以及仅使用单一复合调度规则的方法进行对比。结果表明:所提算法在 6 个不同算例中均获得了最高的平均累计奖励值。

关键词:强化学习; 大规模; 敏捷软件项目调度; 深度 Q 网络; 复合调度规则; 优先经验回放; 强耦合

中图分类号:TP311.5;TP301.6 **文献标志码**:A **doi**:10.13705/j.issn.1671-6833.2023.05.003

近年来,信息技术产业飞速发展,不断变化的用户需求和日益严格的项目要求给软件开发企业带来了巨大的考验。不同于传统的瀑布式开发方法,敏捷软件开发(agile software development, ASD)以用户的需求进化为核心,将开发过程划分为若干个冲刺阶段,每个冲刺结束时交付可使用的软件版本供客户体验,然后根据客户的满意度反馈不断改进,因此受到了很多软件企业的青睐[1]。受小型敏捷公司成功案例的启发,大型软件组织开始尝试大规模敏捷软件项目开发[2]。

大规模敏捷软件项目调度问题(large scale agile software project scheduling problem, LSASPSP)是大规模 ASD 中的关键环节,对于提高软件项目的综合效益和减少项目成本有着至关重要的作用。用户故事是 ASD 从客户视角对功能的简要描述[3],LSASPSP 包括 3 个强耦合的子问题。①故事选择,从待开发故事列表中选择用户故事进入冲刺;②故事分配,将选择进入冲刺的用户故事分配给各个团队;③任务分配,将每个团队分配到的故事分解成任务,按照任务所需技能和任务时长分配给团队中的员工。LSASPSP 的目标就是在满足用户故事和团队员工之间各项约束下合理分配每个冲刺阶段的用户故事,最大化开发项目的价值。在每个冲刺阶段结束时,开发团队所交付软件的质量依赖于对用户故事任务和团队员工的有效调度。到目前为止,学者对瀑布式软件项目调度(waterfall software project scheduling problem, WSPSP)关注较多,很少研究敏捷软件项目调度。Zapotecas-martínez 等[4]考虑了新增/删除用户故事、员工离职/加入等破坏性事件的发生,但未将用户故事划分为具体的任务。Roque 等[5]关注任务分配与员工技能之间的联系,但忽略了用户需求的变更。此外,这 2 个模型仅在小规模

收稿日期:2023-01-12;**修订日期**:2023-04-05

基金项目:国家自然科学基金资助项目(61502239, 62002148);广东省重点实验室项目(2020B121201001);江苏省自然科学基金资助项目(BK20150924)

作者简介:申晓宁(1981—),女,江苏南通人,南京信息工程大学教授,博士,主要从事计算智能、多目标优化、调度技术等研究,E-mail:sxnystsyt@sina.com。

引用本文:申晓宁,毛鸣健,沈如一,等. 基于深度强化学习的大规模敏捷软件项目调度[J]. 郑州大学学报(工学版),2023,44(5):17-23. (SHEN X N, MAO M J, SHEN R Y, et al. Large-scale agile software project scheduling based on deep reinforcement learning[J]. Journal of Zhengzhou University(Engineering Science),2023,44(5):17-23.)

实例上进行了实验验证。针对上述不足,本文建立了 LSASPSP 的数学模型,将故事细化成具体的小任务,并为这些任务设置相应的技能和完成时长属性。对于动态事件,考虑了每个冲刺用户需求和员工工作时长的动态变化。

WSPSP 已被证明是一个 NP-hard 问题,传统的精确算法无法在多项式时间内找到该类问题的有效解,多数学者采用元启发式算法。Xiao 等[6]将 WSPSP 转化为基于图的搜索问题并采用蚁群算法求解;Shen 等[7]提出了一种协同进化的多目标遗传算法求解多目标 WSPSP。此外,强化学习算法也逐渐用于求解 WSPSP,Padberg 等[8]基于实际案例的调度搭建了环境转移模型,使用强化学习 Sarsa 算法优化调度策略,但是,该算法需要大量数据,且无法处理复杂问题。

强化学习通过不断与未知环境交互,学会在动态环境中做出最优决策。DQN(deep Q network)算法[9]利用神经网络提取轨迹的高维特征,拟合动作价值函数[10],引入经验回放池以节约存储空间。LSASPSP 的环境涉及用户故事任务的完成进度、团队员工的时间利用情况等复杂因素,导致状态空间庞大,适合于用 DQN 算法来求解。为有效解决该问题,本文提出基于复合调度规则的优先经验回放双重深度 Q 网络(composite scheduling rule-based priority experience replay double deep Q network, CPD-DQN)算法。该算法采用双重网络分离 DQN 的 Q 值预测和动作选择,避免 Q 值过估计;按优先级采样经验回放池中的轨迹进行训练,提高样本的利用效率。针对 LSASPSP 模型特点,设计了 10 个状态特征来描述调度对环境状态变化产生的影响,以及 12 个复合调度规则作为动作供算法选择,加速算法收敛。最后,算法以各冲刺完成故事总价值为奖励函数,指导算法优化。

1 大规模敏捷软件项目调度模型

本节描述了大规模敏捷软件项目开发调度的项目结构和开发团队的各项属性,并规定各项约束关系,建立了一个贴合实际敏捷开发的 LSASPSP 数学模型。

1.1 问题描述

假设一个待开发的大规模敏捷项目 P,将客户的需求分解成 N 个用户故事 $us_i(i=1,2,\cdots,N)$,用户故事 us_i 可以分解成 tn_i 个具体任务 $task_{ij}(j=1,2,\cdots,tn_i)$,完成这些任务需要用到 S 项技能。敏捷开发公司共有 R 个团队 $Team_r(r=1,2,\cdots,R)$,每个团队中有 M_r 名员工 $e_{rk}(k=1,2,\cdots,M_r)$。

LSASPSP 3 个强耦合子问题的分配方案分别由决策变量 x_i^l、y_{ir}^l 和 z_{ijrk}^l 表示,取值均为 $\{0,1\}$。$x_i^l=1$ 时,选择用户故事 us_i 进入第 l 个冲刺;$y_{ir}^l=1$ 时,将第 l 个冲刺选择的故事 us_i 分配给团队 $Team_r$;$z_{ijrk}^l=1$ 时,将第 l 个冲刺中分配给团队 $Team_r$ 的故事任务 $task_{ij}$ 分配给员工 e_{rk}。LSASPSP 的目标就是在满足团队速度、任务和技能等约束时进行调度,最大化 D 个冲刺内完成用户故事的总价值。

1.2 问题的数学表达

围绕 3 个强耦合子问题,建立 LSASPSP 问题的约束单目标优化模型:

$$\max\ vp=\max\sum_{l=1}^{D}\sum_{us_i\in OUT^l}vp_i; \tag{1}$$

$$\text{s.t.}\quad \sum_{us_i\in U^l}x_i^l\cdot sp_i\leqslant sv^l; \tag{2}$$

$$\sum_{us_i\in U^l}x_i^l\cdot sp_i\cdot y_{ir}^l\leqslant v_r^l; \tag{3}$$

$$x_i^l\cdot y_{ir}^l\cdot z_{ijrk}^l\cdot tsk_{ij}\in esk_{rk}; \tag{4}$$

$$\sum_{r=1}^{R}\sum_{k=1}^{Mr}x_i^l\cdot y_{ir}^l\cdot z_{ijrk}^l=1; \tag{5}$$

$$\sum_{us_i\in U^l}\sum_{j=1}^{tn_i}x_i^l\cdot th_{ij}\cdot y_{ir}^l\cdot z_{ijrk}^l\leqslant eh_{rk}^l。 \tag{6}$$

式(1)为目标函数,表示最大化 D 个冲刺中所完成用户故事的总价值。其中,vp_i 表示价值点,是对用户故事 us_i 市场收益的无单位估算值。OUT^l 为第 l 个冲刺所完成完整用户故事的集合。式(2)~(6)为约束。式(2)表示每个冲刺阶段所选择用户故事的故事点总数不超过冲刺速度 sv^l。其中,U^l 为第 l 个冲刺阶段开始前待开发用户故事的集合;sp_i 表示故事点,是对用户故事 us_i 大小和复杂度的无单位估算值。式(3)表示每个冲刺中团队 $Team_r$ 分配到的故事点总数不超过团队速度 v_r^l。冲刺速度 $sv^l=\sum_{r=1}^{R}v_r^l$,即团队速度 v_r^l 之和。团队速度为各团队前两个冲刺所完成的平均故事点总数,表示为 $v_r^l=\frac{1}{2}(v_r^{l-1}+v_r^{l-2})$,第一个冲刺将团队所有员工的总工作时长按 1 个故事点约为 8 h 的方式估算成初始团队速度,表示为 $v_r^1=\frac{1}{8}\sum_{k=1}^{M_r}eh_{rk}^1$;第二个冲刺的团队速度为第一个冲刺所完成完整用户故事的点数之和,表示为 $v_r^2=\sum_{us_i\in OUT^1}sp_i$。式(4)表示每个冲刺完成任务 $task_{ij}$ 所需要的一项技能 tsk_{ij} 必须在所分配员工 e_{rk} 掌握的技能集合 esk_{rk} 中。式(5)表示每个冲刺的任

务 $task$ 只能由一名员工来完成。式(6)表示每个冲刺过程中分配给员工 e_{rk} 的任务时长 th_{ij} 之和不超过其工作时长 eh_{rk}^{l}。

2　求解 LSASPSP 的 CPDDQN 算法

算法求解框架如图 1 所示。首先,初始化 LSASPSP 环境中的用户故事任务和团队员工信息,计算初始状态特征传递给智能体。智能体依概率 ε 随机选取动作,依概率 $1-\varepsilon$ 用当前网络选择动作。冲刺阶段结束后,环境反馈奖励和新状态特征,并判断调度是否结束。在每个冲刺阶段,将当前/新状态特征、动作、奖励以及调度是否终止的判断存放到回放池中。当有新轨迹加入时,计算其时序差分误差并更新所有轨迹的优先级。之后,轨迹按优先级采样,供给目标网络预测 Q 值,并采用梯度下降训练当前网络参数。当前网络定时将参数复制给目标网络,提高目标网络的预测能力。训练后的当前网络能根据每个动作对应的 Q 值选择 Q 值最大的动作,这样,智能体就能够在调度结束后获得最大的累计奖励值。

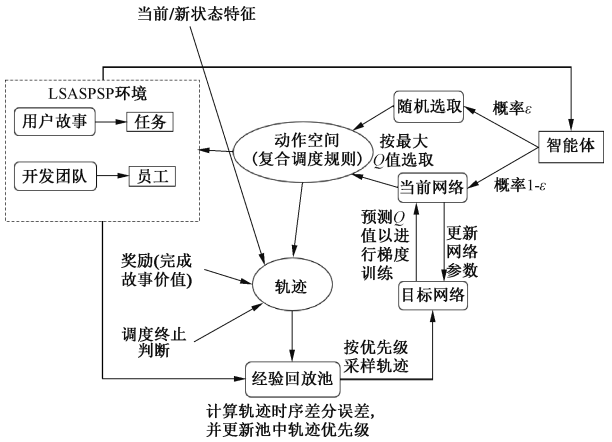

图 1　LSASPSP 的算法求解框架

Figure 1　Framework of CPDDQN for solving LSASPSP

2.1　马尔可夫决策过程

在使用深度强化学习求解 LSASPSP 前,必须先将问题转化为马尔可夫决策过程(Markov decision process, MDP)。MDP 是一个四元组(S,A,R,γ),其中,S 表示状态集合;A 表示动作集合;R 表示奖励集合;γ 表示衰减因子,取值在[0,1],表示后续决策步骤对当前状态执行动作的重要程度。

2.2　状态特征设计

为保证模型的泛化能力,本文设计了 10 个范围在[0,1]的状态特征[11],用于描述每个冲刺阶段开始前用户故事、任务和员工的实际状态。第 l 个冲刺阶段结束后,调度环境的状态特征如式(7)~(16)所示。

所有团队完成完整用户故事的完成率:

$$F_1(l)=\frac{1}{\sum\limits_{us_i\in U^l}x_i^l\cdot sp_i}\sum_{us_i\in OUT^l}x_i^l\cdot sp_i。\tag{7}$$

所有团队完成用户故事工作量的完成率:

$$F_2(l)=\frac{1}{\sum\limits_{us_i\in U^l}x_i^l\cdot eff_i}\sum_{r=1}^{R}\sum_{us_i\in U^l}\sum_{j=1}^{tn_i}x_i^l\cdot y_{ir}^l\cdot z_{ijrk}^l\cdot th_{ij}。\tag{8}$$

式中:eff_i 表示完成用户故事 us_i 花费的时间。

所有团队完成用户故事完成率的平均值:

$$F_3(l)=\frac{1}{R}\sum_{r=1}^{R}\sum_{us_i\in U^l}\frac{\sum\limits_{us_i\in OUT^l}x_i^l\cdot y_{ir}^l\cdot sp_i}{\sum\limits_{us_i\in U^l}x_i^l\cdot y_{ir}^l\cdot sp_i}。\tag{9}$$

所有团队完成用户故事完成率的标准差:

$$F_4(l)=\sqrt{\frac{1}{R}\sum_{r=1}^{R}\left(\frac{\sum\limits_{us_i\in OUT^l}x_i^l\cdot y_{ir}^l\cdot sp_i}{\sum\limits_{us_i\in U^l}x_i^l\cdot y_{ir}^l\cdot sp_i}-F_3(l)\right)^2}。\tag{10}$$

所有团队工作量完成率的平均值:

$$F_5(l)=\frac{1}{R}\sum_{r=1}^{R}\frac{\sum\limits_{us_i\in U^l}\sum\limits_{j=1}^{tn_i}x_i^l\cdot y_{ir}^l\cdot th_{ij}}{\sum\limits_{us_i\in U^l}x_i^l\cdot y_{ir}^l\cdot eff_i}。\tag{11}$$

所有团队工作量完成率的标准差:

$$F_6(l)=\sqrt{\frac{1}{R}\sum_{r=1}^{R}\left(\frac{\sum\limits_{us_i\in U^l}\sum\limits_{j=1}^{tn_i}x_i^l\cdot y_{ir}^l\cdot th_{ij}}{\sum\limits_{us_i\in U^l}x_i^l\cdot y_{ir}^l\cdot eff_i}-F_5(l)\right)^2}。\tag{12}$$

所有团队时间利用率的平均值:

$$F_7(l)=\frac{1}{R}\sum_{r=1}^{R}\frac{\sum\limits_{us_i\in U^l}\sum\limits_{j=1}^{tn_i}x_i^l\cdot y_{ir}^l\cdot th_{ij}}{\sum\limits_{k=1}^{M_r}eh_{rk}^l}。\tag{13}$$

所有团队时间利用率的标准差:

$$F_8(l)=\sqrt{\frac{1}{R}\sum_{r=1}^{R}\left(\frac{\sum\limits_{us_i\in U^l}\sum\limits_{j=1}^{tn_i}x_i^l\cdot y_{ir}^l\cdot th_{ij}}{\sum\limits_{k=1}^{M_r}eh_{rk}^l}-F_7(l)\right)^2}。\tag{14}$$

所有团队员工时间利用率的平均值:

$$F_9(l)=\frac{1}{\sum_{r=1}^{R}M_r}\sum_{r=1}^{R}\sum_{k=1}^{M_r}\frac{\sum_{us_i\in U^l}\sum_{j=1}^{tn_i}x_i^l\cdot y_{ir}^l\cdot z_{ijrk}^l\cdot th_{ij}}{eh_{rk}^l}。\tag{15}$$

所有团队员工时间利用率的标准差:

$$F_{10}(l)=\sqrt{\frac{1}{R}\left(\frac{\sum_{us_i\in U^l}\sum_{j=1}^{tn_i}x_i^l\cdot y_{ir}^l\cdot z_{ijrk}^l\cdot th_{ij}}{eh_{rk}^l}-F_9(l)\right)^2}。\tag{16}$$

2.3 动作空间设计

动作空间是智能体在各种状态下所有能采取的动作集合,以调度规则的形式供智能体选择执行[12]。针对 LSASPSP 的 3 个强耦合子问题,本文设计了 7 个通用的单一调度规则,分别用于描述每个冲刺阶段的故事选择、故事分配和任务分配,如表 1 所示。

表 1 单一调度规则

Table 1 Single scheduling rule

决策变量	符号表示	描述
故事选择	SSL	优先选取价值点最大的故事
	SSA	优先选取单位故事点上价值最大的故事
故事分配	STL	故事优先分配给速度最大的团队
	STS	故事优先分配给速度最小的团队
任务分配	TEL	任务优先分配给剩余工作时间最长的候选者
	TES	任务优先分配给剩余工作时间最短的候选者
	TER	任务优先分配给最早等待的候选员工

将这 3 个强耦合决策变量对应的 7 个单一调度规则按 2×2×3 进行排列组合,得到了 12 个复合调度规则 $CR_1\sim CR_{12}$,作为动作空间供智能体进行选择。

2.4 奖励函数设计

奖励函数是环境根据智能体所执行动作反馈的即时回报,用于帮助智能体进行正确的决策,一般按照问题的目标函数进行设计。本文的奖励函数为每个冲刺所完成用户故事的总价值,如式(17)所示:

$$r_t=\sum_{us_i\in OUT^l}vp_i。\tag{17}$$

2.5 双重 Q 网络

Q 值的更新过程如式(18)所示,DQN 选择动作的网络 Q_{t+1} 和预测 Q 值的网络 Q_t 是相同的。更新 Q 值时,使用 max 操作可以快速让 Q 值朝着优化目标靠拢,但是容易导致过度估计。针对该缺点,本文采用双重深度 Q 网络策略(double deep Q network, DDQN)[13]。当前网络观测环境在 t 时刻的状态 s_t,选择 Q 值最大的动作 a_t;目标网络根据 $t+1$ 时刻的状态估计 Q 值,采用梯度下降训练当前网络的参数。每隔固定冲刺阶段,算法将当前网络的参数 θ 复制给目标网络的参数 θ^-,这样,将预测 Q 值和选择动作分离,提高了神经网络预测的准确性。

$$Q_t=Q_t+\alpha(r_{t+1}+\gamma Q_{t+1}(s,\arg\max Q)-Q_t)。\tag{18}$$

2.6 优先经验回放

在深度 Q 网络参数的训练过程中,历史轨迹的采样是均匀随机的。但是,某些相互邻近的轨迹本就相关性强,不同数据对梯度学习的贡献度也不一致,这样容易导致学习效率低和过拟合等情况发生。因此,本文引入优先经验回放策略[14],采用轨迹的时序差分(temporal difference, TD)误差作为评价轨迹重要性指标,计算轨迹间的权重改变量并训练 Q 网络的参数。当经验回放池中有新轨迹加入时,计算该轨迹的 TD 误差并更新所有轨迹的优先级。在进行网络的梯度下降训练时,会优先采样级别较高的一批轨迹。如果轨迹数量超过经验回放池的固定容量,优先级最低的轨迹会被最先剔除出去,以提高轨迹的利用效率。

2.7 算法求解流程

在算法求解过程中,首先初始化用户故事和团队员工的属性值、神经网络以及经验回放池的各项参数。智能体观测到环境状态 S_0,并根据 ε^-贪婪策略 $\pi_\theta(S_0)$ 选择执行动作 A_0。算法共有两个循环:第一个循环是智能体和调度环境交互的过程;第二个循环是采样轨迹训练当前网络参数的过程。在第一个循环中,智能体观测到环境状态特征 S_t,按策略选择执行动作 A_t。调度环境中已分配的用户故事和任务团队被员工完成后,环境的状态特征变为 S_{t+1}。调度是否结束用 *done* 来表示,如果故事全都被完成,则 $done=true$;反之,$done=false$。在第二个循环中,当调度时刻 t 是 K 的整数倍时,将轨迹 $tran_b(b=1,2,\cdots,V)$ 按优先级分布 $m\sim P(m)=p_m^\alpha/\sum_b p_b^\alpha$ 采样 h 条轨迹。然后,根据式(19)和(20)计算所采样 h 条轨迹的重要性权重 ω_m 和 TD 误差 δ_m,更新轨迹的优先级 p_m。最后,使用目标网络预测 $Q(S_{m-1},A_{m-1})$ 值,并按式(21)对累计权重改变量 Δ 进行梯度训练。采样结束后,利用式(22)更新当前网络参数 θ,并重置 $\Delta=0$。每隔步长 K,当前网络将参数 θ 复制给目标网络参数 θ_{target}。当平均累计奖励收敛稳定后,智能体使用更新后的最优策略 $\pi_\theta(S_t)$ 选择动作 A_t。

$$\omega_m = (V \cdot P(m))^{-\beta} / \max \omega_b; \tag{19}$$

$$\delta_m = R_m + \gamma_m Q_{\text{target}}(S_m, \arg\max_a Q(S_m, a)) - Q(S_{m-1}, A_{m-1}); \tag{20}$$

$$\Delta \leftarrow \Delta + \omega_m \cdot \delta_m \cdot \nabla_\theta Q(S_{m-1}, A_{m-1}); \tag{21}$$

$$\theta \leftarrow \theta + \eta \cdot \Delta。 \tag{22}$$

3　仿真测试与对比分析

3.1　实验设置

本文实验基于 Python 编程语言和 Pytorch 框架在 Pycharm 软件上运行算法,计算机配置为 AMD Ryzen 7-5800 H with Radeon Graphics @ 3.20 GHz, RAM 16 GB。

为了验证本文提出的算法 CPDDQN,本文参考了挪威最大的大规模敏捷开发项目之一 Beta 的各项数据[2]以及敏捷专家对于敏捷软件开发实践的估算与计划[3],生成了 6 个 LSASPSP 算例,以用户故事和团队的数量将其命名为 N_R,分别为 200_3、200_5、300_3、300_5、500_3 和 500_5。项目的目标冲刺数 D 为 16,每个冲刺周期为 2 周。新增和初始用户故事的故事点均从斐波那契数列{2,3,5,8,13}中选择。用户故事按必须故事(基础功能)、线性增加故事(具备持续收益的功能)以及兴奋点故事(具备爆发性收益的功能)划分,价值点分别在{2,3}、{5,8}、{13}中选择。而新增故事不考虑必需故事,价值点只在{5,8,13}选取。故事工作量的估算按一个故事点约为 8 h 进行高斯分布生成,任务的时长设置在[4 h,8 h]内。开发团队有 3~5 个,每个团队的员工数量在[5,9]。团队每个员工掌握的以及任务所需的技能对应于页面开发、数据库管理、美工设计、算法编程和自动化测试等软件开发岗位,用{1,2,3,4,5}表示。考虑每个冲刺客户需求的变化,根据冲刺速度和已结束的冲刺数量估算出项目的预计完成冲刺数。若预计完成冲刺数不超过 D,则从新增用户故事集合中随机选取用户故事优先开发;反之,则删除单位故事点价值最低的用户故事。在新增和删除用户故事的过程中,预计完成冲刺数始终不得超过目标冲刺数 D。考虑每个冲刺阶段员工面临的各种特殊情况,员工在每个冲刺阶段的总工作时长会在[60 h,80 h]动态变化。

CPDDQN 算法的各项参数如下:回放池容量 V 为 512,更新频率 K 为 10,最小采样批次 h 为 64,学习率 η 为 0.002,优先级权重 α 为 0.6,采样权重系数 β 为 0.4,概率 ε 为 0.01,衰减率 γ 为 0.98。

3.2　本文算法的收敛性验证

为验证本文算法的收敛性,与 DQN 和 DDQN 算法在 6 个算例中进行了实验验证。算法迭代 10 000 次,每 200 次迭代将最后 50 个累计奖励值取平均,得到 50 个平均累计奖励。3 种算法在算例 200_5 中的平均累计奖励变化曲线如图 2 所示。

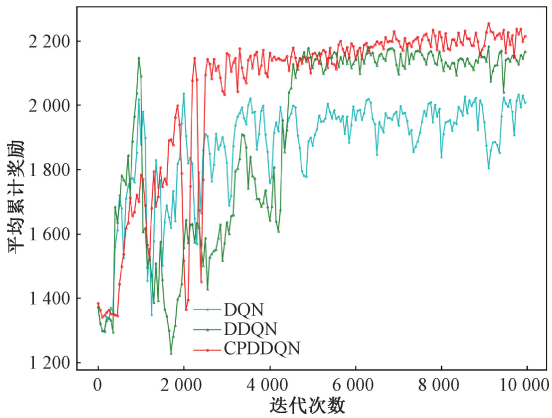

图 2　3 种算法的平均累计奖励变化曲线

Figure 2　Average cumulative reward of three algorithms

由图 2 可知,CPDDQN 的平均累计奖励在迭代 3 000 次后收敛,而 DDQN 则要 4 000 次才趋于稳定。在收敛精度上,CPDDQN 稳定后平均累计奖励上下波动最小。DQN 的轨迹利用效率低,预测 Q 值的能力差,因此稳定性和获得的平均累计奖励值都要低于 CPDDQN。算例 200_5 中新增故事出现的随机性较其他算例大,导致 3 个算法的稳定性都较差。在其他算例中,CPDDQN 的收敛速度和精度也优于其他两种算法。3 种算法的最大预测 Q 值变化如图 3 所示。

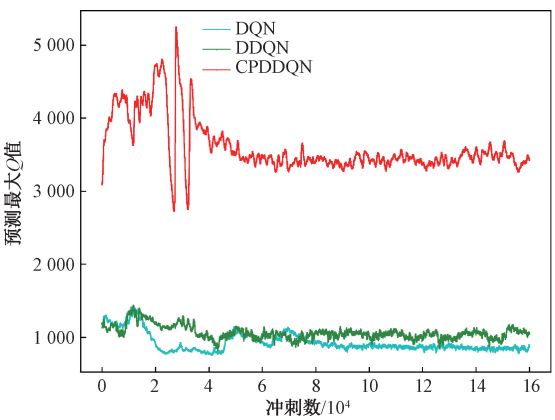

图 3　3 种算法的预测 Q 值变化

Figure 3　Predicted Q value change of three algorithms

CPDDQN 只保留重要的轨迹,并按优先级采样,而 DQN 与 DDQN 均匀随机采样轨迹,因此 CPDDQN 的最大预测 Q 值远高于 DQN 和 DDQN。

3.3　本文算法与其他算法的对比分析

为验证 CPDDQN 的性能,本文将 DQN、DDQN 和 12 个复合调度规则作为对比算法,在 6 个算例中进行实验。为保证复合调度规则不受随机性影响,

每种规则运行10遍,每遍迭代1 000次,记录最后50次迭代的累计奖励值平均值,并以其最大值为运行结果。

表2为CPDDQN与对比算法的实验结果。由表2可知,CPDDQN在绝大多数算例中取得了最高的平均累计奖励。算例200_5中,5个团队很快就能完成200个初始故事,有更多价值点高的新增故事加入,故所获累计奖励要远高于其他算例。为避免陷入局部最优[15],CPDDQN根据概率随机选择动作,获得的平均累计奖励值会略低于CR_7。

表2 CPDDQN算法与对比算法的实验结果

Table 2 Experimental results of CPDDQN algorithm and comparison algorithm

算法	平均累计奖励值					
	200_3	200_5	300_3	300_5	500_3	500_5
CR_1	2 936.090	6 248.969	2 377.059	4 603.31	1 990.219	3 486.678
CR_2	1 898.337	3 648.424	1 859.094	2 688.201	1 310.241	1 831.918
CR_3	1 810.319	3 230.634	1 978.648	2 764.505	1 296.890	1 833.150
CR_4	2 943.257	6 089.245	2 275.692	4 591.882	2 020.924	3 747.559
CR_5	1 909.159	3 755.186	1 926.934	2 667.279	1 381.519	2 028.157
CR_6	1 812.760	3 399.241	1 951.692	2 724.209	1 348.242	2 036.500
CR_7	3 163.510	**7 054.61**	2 392.640	4 778.651	2 091.442	3 245.436
CR_8	1 863.429	3 687.184	1 950.972	2 797.991	1 250.808	2 127.209
CR_9	1 803.817	3 395.357	2 072.315	2 787.691	1 286.153	2 185.997
CR_{10}	3 184.471	6 620.211	2 344.104	4 619.703	2 019.857	4 009.573
CR_{11}	1 930.873	3 774.065	2 071.180	2 924.761	1 354.600	2 261.544
CR_{12}	1 817.064	3 434.261	2 024.796	2 926.283	1 334.956	2 239.926
DQN	3 119.122	6 248.408	2 151.696	2 695.438	2 119.788	3 810.110
DDQN	3 174.268	6 694.306	2 209.686	2 927.962	2 206.132	3 973.166
CPDDQN	**3 201.112**	6 735.686	**2 468.382**	**5 052.078**	**2 254.450**	**4 094.942**

4 结论

本文考虑团队速度、任务时长和技能等约束,引入每个冲刺客户需求和员工工作时长的变化等动态事件,建立了以最大化团队所开发项目总价值为目标的LSASPSP约束单目标优化模型,并用CPDDQN算法求解所建模型。针对敏捷开发特点,采用10个状态特征描述调度环境信息,结合3个强耦合决策变量设计了12个复合调度规则作为智能体的候选动作。引入双重网络和优先经验回放策略,提高Q网络预测能力和轨迹利用效率。在6个LSASPSP算例上的实验结果表明,所提算法的稳定性和所获平均累计奖励值均优于其他深度强化学习算法和12种复合调度规则。

本文所建LSASPSP模型对实际大规模敏捷软件开发中的多种动态事件和不确定因素尚欠考虑,如多团队间员工的调动,员工技能提升对任务时长的影响等。在后续研究过程中,将在模型中引入更多实际因素,并探讨更多深度强化学习算法在LSASPSP中的应用,并与所提算法进行比较分析。

参考文献:

[1] 王映红. 企业大规模敏捷转型探索与实践[J]. 金融科技时代, 2017, 25(11): 84-85.
WANG Y H. Exploration and practice of large-scale agile transformation of enterprises[J]. Financial Technology Time, 2017, 25(11): 84-85.

[2] BIESIALSKA K, FRANCH X, MUNTÉS-MULERO V. Mining dependencies in large-scale agile software development projects: a quantitative industry study[C]//Evaluation and Assessment in Software Engineering. New York: ACM, 2021: 20-29.

[3] COHN M. Agile estimating and planning[M]. Upper Saddle River: Prentice Hall, 2005.

[4] ZAPOTECAS-MARTÍNEZ S, GARCÍA-NÁJERA A, CERVANTES H. Multi-objective optimization in the agile software project scheduling using decomposition[C]// Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion. New York: ACM, 2020: 1495-1502.

[5] ROQUE L, ARAÚJO A A, DANTAS A, et al. Human Resource Allocation in Agile Software Projects Based on Task Similarities[C]//International Symposium on Search Based Software Engineering. Cham: Springer, 2016: 291-297.

[6] XIAO J, AO X T, TANG Y. Solving software project scheduling problems with ant colony optimization[J]. Computers & Operations Research, 2013, 40(1): 33-46.

[7] SHEN X N, GUO Y N, LI A M. Cooperative coevolution with an improved resource allocation for large-scale multi-objective software project scheduling[J]. Applied Soft Computing, 2020, 88: 106059.

[8] PADBERG F, WEISS D. Optimal scheduling of software projects using reinforcement learning[C]//2011 18th Asia-Pacific Software Engineering Conference. Piscataway: IEEE, 2012: 9-16.

[9] LIU W T, SU S, TANG T, et al. A DQN-based intelligent control method for heavy haul trains on long steep downhill section[J]. Transportation Research Part C: Emerging Technologies, 2021, 129: 103249.

[10] HUYNH T N, DO D T T, LEE J. Q-Learning-based parameter control in differential evolution for structural optimization[J]. Applied Soft Computing, 2021, 107: 107464.

[11] LUO S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning[J]. Applied Soft Computing, 2020, 91: 106208.

[12] LI Y X, GU W B, YUAN M H, et al. Real-time data-driven dynamic scheduling for flexible job shop with insufficient transportation resources using hybrid deep Q network[J]. Robotics and Computer-Integrated Manufacturing, 2022, 74: 102283.

[13] Van HASSELT H, GUEZ A, SILVER D. Deep reinforcement learning with double Q-Learning[C]//Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. New York: ACM, 2016: 2094-2100.

[14] SCHAUL T, QUAN J, ANTONOGLOU I, et al. Prioritized experience replay[EB/OL]. (2016-02-25)[2023-02-09]. https://arxiv.org/abs/1511.05952.

[15] 王培崇, 尹欣洁, 李丽荣. 一种具有学习机制的海鸥优化算法[J]. 郑州大学学报(工学版), 2022, 43(6): 8-14.
WANG P C, YIN X J, LI L R. An improved seagull optimization algorithm with learning[J]. Journal of Zhengzhou University (Engineering Science), 2022, 43(6): 8-14.

Large-scale Agile Software Project Scheduling Based on Deep Reinforcement Learning

SHEN Xiaoning[1,2,3,4], MAO Mingjian[1], SHEN Ruyi[1], SONG Liyan[5]

(1. School of Automation, Nanjing University of Information Science and Technology, Nanjing 210044, China; 2. Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology, Nanjing University of Information Science and Technology, Nanjing 210044, China; 3. Jiangsu Key Laboratory of Big Data Analysis Technology, Nanjing University of Information Science and Technology, Nanjing 210044, China; 4. Jiangsu Engineering Research Center on Meteorological Energy Using and Control (C-MEIC), Nanjing 210044, China; 5. Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation, Southern University of Science and Technology, Shenzhen 518055, China)

Abstract: This study aimed to solve the scheduling problem of large-scale agile software project. It was decomposed into three strong-coupled subproblems: story selection, story allocation and task allocation. Dynamic events such as the addition and deletion of user stories, the change of employee's working hours in each sprint, and other constraints such as team development speed, task duration and skills were introduced. To maximize the total value of user stories completed by the project, a large-scale agile software project scheduling mathematical model was established. According to the characteristics of the problem, the Markov decision process was designed. Ten state features were used to describe the agile scheduling environment at the beginning of each sprint; 12 composite scheduling rules were designed as candidate actions of the agent; and rewards were defined according to the objective function of the scheduling model. A priority experience replay double deep Q network algorithm based on composite scheduling rules was proposed to solve the built model. The double Q network strategy and priority experience replay strategy were introduced to avoid the over-estimation problem of deep Q network and improve the utilization efficiency of trajectory information in the experience replay pool. In order to verify the effectiveness of the proposed algorithm, experiments were carried out in six large-scale agile software project scheduling numerical examples, and the convergence of the proposed algorithm was analyzed. According to the performance measurement of the algorithm, it was compared with the existing representative algorithm DQN, double deep Q network and 12 single composite scheduling rules. The results showed that it had the highest average cumulative reward value in 6 different numerical examples.

Keywords: reinforcement learning; large-scale; agile software project scheduling; deep Q network; composite scheduling rules; priority experience replay; strong coupling