软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(12):3836−3851 [doi: 10.13328/j.cnki.jos.005969]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

新型时空众包平台中的在线三维稳定匹配问题*

李博扬[1], 成雨蓉[2], 王国仁[2], 袁 野[2], 孙永佼[1]

[1](东北大学 计算机科学与工程学院,辽宁 沈阳 110819)

[2](北京理工大学 计算机学院,北京 100081)

通讯作者: 成雨蓉, E-mail: yrcheng@bit.edu.cn

摘 要: 近年来,时空众包平台正逐步走入人们的生活,并受到研究者的广泛关注.在时空众包平台中,任务分配是一个核心问题,即在满足时间和空间的条件约束下,如何为不同用户分配合适的工人来进行服务.现有的工作往往将最大化任务匹配个数或效用值之和作为研究目标,这些方法关注全局的解决方案,但是没有考虑用户和工人的偏好来提高他们对于分配的满意程度.此外,现有工作大多只考虑用户和工人两种角色,即工人移动到用户当前位置进行服务.但是,新型时空众包平台的中往往包含用户、工人和工作点三种角色,即为用户和工人分配一个工作点来进行服务.基于以上不足,三维时空稳定分配问题被提出.但是,此问题只关注了静态场景,而时空众包平台往往是在线的,即工人和用户发出的任务都是实时出现的.因此,提出了面向新型时空众包平台的三维在线稳定匹配问题和一种基础算法.通过分析基础算法的不足,结合人工智能的方法提出一种改进算法来解决这个问题.采用大量的真实数据和合成数据集来验证算法的高效性和有效性.

关键词: 时空数据;众包;稳定匹配;在线算法;预测

中图法分类号: TP311

中文引用格式: 李博扬,成雨蓉,王国仁,袁野,孙永佼.新型时空众包平台中的在线三维稳定匹配问题.软件学报,2020,31(12):3836−3851. http://www.jos.org.cn/1000-9825/5969.htm

英文引用格式: Li BY, Cheng YR, Wang GR, Yuan Y, Sun YJ. 3D-online stable matching problem for new spatial crowdsourcing platforms. Ruan Jian Xue Bao/Journal of Software, 2020,31(12):3836−3851 (in Chinese). http://www.jos.org.cn/1000-9825/5969.htm

3D-online Stable Matching Problem for New Spatial Crowdsourcing Platforms

LI Bo-Yang[1], CHENG Yu-Rong[2], WANG Guo-Ren[2], YUAN Ye[2], SUN Yong-Jiao[1]

[1](School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China)

[2](School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

Abstract: In recent years, spatial crowdsourcing platforms attract more and more attention. One of the core issues is to assign proper workers to users to finish their tasks under the temporal and spatial constraints. Most existing works aim to maximize the number of tasks that are finished or the sum of utility score. These approaches ignore the preference of users and workers. Moreover, existing works usually only focus on two roles, workers and users. Workers travel to the location of users to finish the tasks. However, new spatial crowdsourcing platforms contain three types of roles, workers, users, and workplaces. The platforms assign workplaces for workers and

* 基金项目: 国家重点研发计划(2016YFC1401900); 国家自然科学基金(U1811262, 61902023, 61932004, 61572119, 61622202, 61672145, 61732003, 61572121, 61972077); 中央高校基础科研业务费(N181605012, N171604007); 中国博士后科学基金(2018M631358)

Foundation item: National Key Research and Development Program of China (2016YFC1401900); National Natural Science Foundation of China (U1811262, 61902023, 61932004, 61572119, 61622202, 61672145, 61732003, 61572121, 61972077); Fundamental Research Funds for the Central Universities (N181605012, N171604007); China Postdoctoral Science General Program Foundation (2018M631358)

收稿时间: 2015-04-22; 修改时间: 2015-06-11, 2015-08-11; 采用时间: 2015-09-10

users to finish the tasks. Thus, the stable matching problem in the three-dimensional platforms is proposed to solve the static scenarios. However, most spatial crowdsourcing platforms are online scenarios. Workers and tasks issued by the users appear in real time. Therefore, a three-dimensional online stable matching problem is formalized in new spatial crowdsourcing platforms. A baseline algorithm and an improved algorithm are proposed which benefit from the advantages of artificial intelligence to solve this problem. Finally, extensive experiments are conducted on real datasets and synthetic datasets to verify the efficiency and effectiveness of the proposed algorithms.

Key words: spatial database; crowdsourcing; stable matching; online algorithm; prediction

近年来,随着移动网络技术和 O2O 商业模式的发展,时空众包平台正逐步走入人们的生活[1].常见的时空众包平台包括出行服务平台(例如滴滴(https://www.didiglobal.com/))、外卖派送平台(例如饿了么(https://www.ele.me/))等等.在这些平台中,用户通过互联网在线上发布自己的任务或需求,平台为用户安排合适的众包工人在线下完成任务.区别于传统的众包平台,时空众包平台同时受线上因素和线下的时空因素的约束.以滴滴平台为例,用户发布用车服务的订单,平台将订单派给司机到指定地点接送乘客.用户所在区域的交通状况和供需情况,会很大程度上决定用户的等待时间.

在时空众包平台中,一个经典的问题是:在满足时空约束的前提下,为用户安排合适的工人来完成其发布的任务.现有的研究工作[2−4]往往只关注用户和工人两种角色,他们将用户发布任务的位置视为工人完成任务的地点.在这种模型下,工人从其当前的位置移动到任务的地点来服务用户.以外卖配送平台为例,配送人员需要将外卖送到用户给定的地址.这些工作的求解目标是最大化匹配个数或效用值值之和(用户满意度、平台收入等),或者最小化分配成本(距离成本等).

例 1:图 1 描述了一个二维任务分配策略的情景.在地图中存在 5 个用户和 5 个工人,用户和工人的地理位置分别标注在地图上.任意两个位置之间的距离采用欧式距离计算.当系统为用户分配工人后,工人从当前位置移动到用户的任务位置.本例中,以最小化工人移动距离为求解目标,红线表示了任务的分配结果和工人的移动距离.

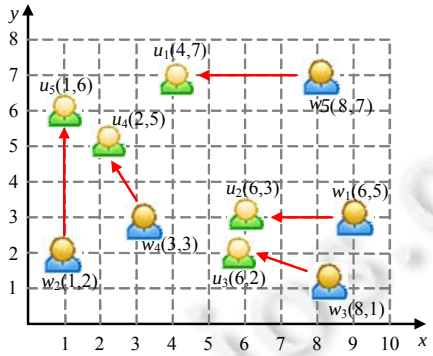


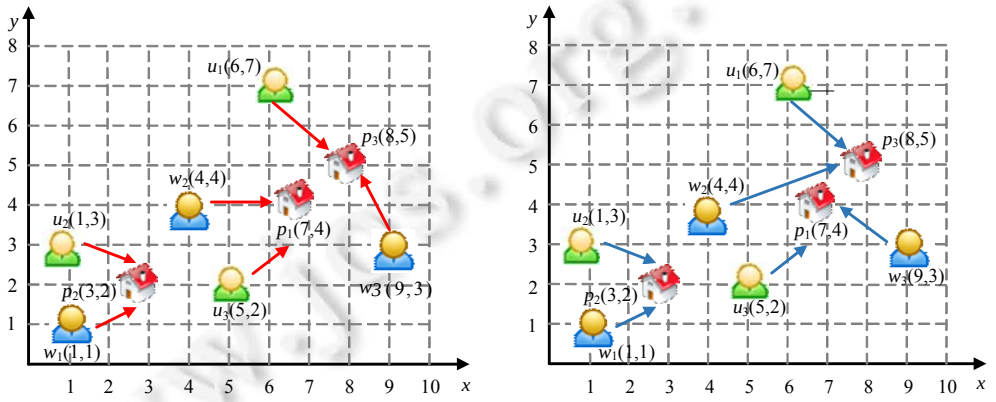
Fig.1 Two-dimensional work assignment strategy

图 1 二维任务分配策略

然而,一些新的众包平台的出现也带来了新的挑战.这些新平台不再局限于两种用户和工人两种角色.在这些平台中,用户不再原地等待工人上门服务,而是与工人约定一个地点来完成任务,例如南瓜车(http://www.nanguache.com/).这些平台为用户和工人提供专业的场地,所以用户和工人需要从当前位置移动到平台安排的场地来完成任务.这样,平台中的角色由用户和工人两种角色转变为用户、工人、工作点这 3 种角色.由于角色的增加,原有的研究无法直接扩展到新的平台中.针对新的挑战,Song 等人[5]提出了面向三维的任务分配问题,根据用户、工人、地点的不同组合方式,会有不同的效用值,研究如何将这个效用值最大化.

例 2:图 2(a)中描述了一个三维任务分配策略的情景.在地图中共存在 3 个用户、3 个工人和 3 个工作点.当用户和工人被分配工作点后,他们将移动到工作点的位置来完成任务.采用文献[5]的策略,将最小化用户和工人的移动距离作为求解目标,红线表示了用户和工人的移动距离.

现有方法还存在另一个问题:现有的研究只关注了全局最优的问题,但是并不能同时根据用户和工人的个人情况来考虑分配结果.如例 2 中所示:虽然取得了全局移动距离最短,但是对于用户 u_3 和工人 w_2 来说,p_2 比 p_1 同时距离他们两个更近,没有分配到 p_2 会让他们对让当前分配产生不满的情绪.为解决这个问题,文献[6]提出了面向三维的稳定匹配问题,该问题保证每一个任务分配的结果都处在一个稳定的状态.即:对于一组匹配的工人和用户,不存在一个对于工人和用户都更适合的工作点.如图 2(b)所示,虽然该方案并不是全局移动距离最小的分配,但是当前的每一个分配都处于稳定的状态,降低了用户和工人的不满情绪.但是,该工作面向的是静态匹配问题,现有的平台要具有动态处理问题的能力,用户的任务和工人的状态都是实时变化的,该工作并不能很好地解决在线问题.



(a) 最小化移动距离分配策略 (b) 稳定分配策略

Fig.2 Three-dimensional work assignment strategy

图 2 三维任务分配策略

综上所述,在时空众包平台的任务分配问题中,根据用户和工人的实际情况考虑分配结果更加合理,同时要具有处理在线任务的能力.因此,定义一种在线的三维稳定匹配问题是本文要解决的问题之一.另外,由于现有的研究人员为研究过在线的三维稳定匹配问题,因此,现有的工作能够提供的参考性有限.而在线的三维稳定匹配问题更加复杂,不确定性因素更多,从而更加难以解决.为此,本文提出了时空众包平台中的在线三维稳定匹配问题,既保证任务分配的质量,又具有实时性,弥补了现有工作的不足.为解决该问题,本文首先提出一种基础算法.基础算法采用贪心的思想,当用户到达最长等待时间的时候,根据距离来为用户和工人安排工作点.由于在线问题具有很强的未知性和不确定性,本文提出了一种改进算法,结合人工智能中的预测技术,为在线匹配提供指导.最后,本文在真实数据集和合成数据集上验证了算法的有效性和高效性.

本文第 1 节对时空众包平台中的实时三维稳定匹配问题进行定义.第 2 节对相关工作加以总结.第 3 节介绍一个基础算法,并分析其正确性和复杂度.第 4 节分析基础算法的不足,并提出一种改进的方法.第 5 节通过真实数据集和合成数据集上的实验,验证本文提出算法的高效性和有效性.第 6 节对全文加以总结,并提出对未来工作的设想.

1 问题定义

本节中,我们先介绍时空众包平台中的一些基本概念,然后给出时空众包平台中的在线三维稳定匹配问题正式定义.

一个时空众包平台由用户、工人和工作点组成.平台中的用户会发布一系列的任务,给定由 n 个用户组成的用户集 U,每个用户 $u\in U$,通常被定义为以下形式:$u=(l_u,D_u,S_u,T_u)$,l_u 是用户当前的位置,D_u 是用户对于所有工作点根据距离的升序排序,S_u 是用户发出任务的时刻,T_u 是用户可以等待的最长时间.如果用户发出的任务在 S_u+T_u 时刻之前还没有被安排合适的工人来服务,那么这个任务将会失效.平台中的工人会响应用户发出的任

务,给定由 m 个工人组成的工人集 W,每个工人 $w\in W$ 通常被定义为以下形式:$w=(l_w,D_w,S_w)$,l_w 是工人当前的位置,D_w 是工人对于所有工作点根据距离的升序排序,S_w 是工人开始工作的时间.即:从 S_w 开始,工人可以接受系统中的任务.如果工人已经接收了一个任务,那么他/她将不再接收其他任务.平台中的工作点是供用户和工人完成任务的地点,对于每个工作点 $p\in P$,通常被定义为以下形式:$p=(l_p,DW_p,QU_p)$,l_p 是工作点的位置,DW_p 和 DU_p 是工作点对于工人和用户的距离升序排序.在最开始,所有工作点都是可用的,如果一个工作点被分配了用户和工人,那么它将变为不可用状态.

例 3:图 3 和表 1 描述了一个在线任务分配场景,其中,图 3 描述了工人、用户和工作点的位置信息,表 1 描述了用户和工人在线到达的时间.在 t_0 时刻,所有工作点都处于可用状态,w_1 和 u_1 出现在地图上;随后在 t_1 时刻,w_2 出现在地图上,表示他/她可以平台分配的任务.假设 u_1 的最长等待时间为 2 个时间段,那么平台可以安排 t_2 时刻或之前出现的工人(即 w_1 或 w_2)来完成 u_1 的任务;否则任务在 t_2 时刻到期,并在地图中消失.

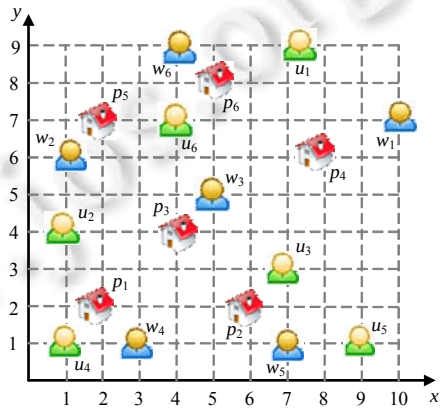


Fig.3 Locations of workers, users and workplaces

图 3 用户、工人和工作点的位置分布

Table 1 Arrival time of workers and users

表 1 工人和用户的出现时间

t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
w_1 u_1	w_2	u_2	w_3	u_3	w_4	u_4 u_5	w_5		w_6 u_6

当平台为用户分配工人和工作点之后,我们将这个分配称为一个元组.在本文研究的问题中,对于元组有特殊的稳定性要求,满足这个要求的元组被称为稳定元组.

定义 1(稳定元组). 一个元组(w,u,p)被称为稳定元组,当且仅当不存在另一个工作点 p',满足 w 和 p'的距离小于 w 和 p 的距离,并且 u 和 p'的距离小于 u 和 p 的距离.

在例 2 中,(w_1,u_3,p_2)是一个不稳定元组,由于 p_3 到 w_1 和 u_3 的距离小于 p_2 到 w_1 和 u_3 的距离,所以(w_1,u_3,p_2)是不稳定的.与之相反,(w_2,u_1,p_1)是稳定元组,原因是不存在一个工作点,同时距离 w_2 和 u_1 更近.同理,(w_3,u_2,p_3)和图 2(b)中的所有元组也都是稳定元组.

基于以上的基本概念,接下来介绍三维在线稳定匹配问题的定义.

定义 2(三维在线稳定匹配问题). 给定用户集 U、工人集 W 和工作点集 P,三维在线问题匹配问题的目标是找到一个满足以下条件的匹配数最多的结果 M(即 $\max|M|$):

(1) 稳定约束:每个匹配的元组$(w,u,p)\in M$ 都是稳定元组;

(2) 时间约束:每个用户都是在$[S_u,S_u+T_u]$时间段内被匹配工人,每个工人都是在 S_w 时刻后被匹配任务;

(3) 唯一性约束:一旦匹配成功,这个匹配中的用户、工人、工作点都不会被其他匹配占用.

在以上的定义中,每个工作点只能匹配一个工人和一个用户.通过将工作点进行复制的方法,可以将这个约束直接扩展到一个地点匹配多个工人和多个用户的场景.本文采用欧几里德距离计算两个点之间的直线距离,

在实际应用中,也可以使用最短路距离等,这些结果是等价的.

三维在线稳定匹配问题的离线版本与在线版本问题相同,只是在线问题中平台只知道当前已经到达的用户和工人信息;而在离线问题中,所有用户、工人和工作点的时空信息都是已知的.

定理 1. 三维在线稳定匹配问题的离线版本是 NP-难.

证明:考虑离线问题中的一个特殊情况,假设用户和工人的出现时间都是 0 时刻(即开始时刻全部出现),用户等待时间为无穷大.那么我们可以将文献[6]中的三维稳定匹配(3D-SSM)问题归约到这个特殊情况.所以,三维在线稳定匹配问题的离线版本是 NP-难. □

鉴于问题的困难性以及在线问题具有很强的未知性和不确定性,在后面的章节中,本文先后提出两个近似算法对该问题进行求解.

2 相关工作

本节从 3 个角度总结与本文相关的现有工作:(1) 时空众包;(2) 在线匹配;(3) 离线稳定婚姻匹配.

2.1 时空众包

随着移动互联网和 O2O 商业模式的发展,时空众包正逐渐成为研究领域的热点话题.时空众包平台中的用户指通过互联网发布时空众包任务,并通过众包工人在现实世界中完成该任务.区别于互联网众包平台,时空众包平台同时受线上和线下的时空属性的约束.任务分配是时空众包领域的一个核心问题,即,如何为众包任务安排合适的众包工人.按照问题类别划分,任务分配问题被分为任务匹配和任务规划两种问题.

- 任务匹配问题通常将众包工人和众包任务进行一对一或一对多的匹配.Xing 等人[7]将匹配问题和博弈论相结合,以此来提高用户对匹配的满意程度.Zhang 等人[8]根据工人的可靠程度和雇佣成本,提出了一个可靠任务分配问题;
- 任务规划问题则是为工人规划路径来完成多个任务.Deng 等人[9]研究在离线任务下为工人规划路径来尽可能地完成更多的任务.Tao 等人[10]面向在线场景研究当工人和任务在线到达时如何最大化完成任务的总效用值.

文献[11,12]研究了基于事件的社交网络(EBSN)中用户行程规划问题,考虑了时间冲突、行程预算以及活动人数等约束来为用户安排最满意的行程安排.在此基础上,Cheng 等人[13]也提出了一个基于事件的社交网络中的稳定规划算法,该算法同时考虑了用户和活动举办者的双向偏好.文献[6]首次提出一个面向三维的稳定匹配问题,证明该为题为 NP 难,并提出两种近似算法(DSM 和 Path Swap)进行求解:DSM 算法采用拆分的思想动态的为用户进行匹配;Path Swap 算法构建一个索引,通过此索引为用户匹配任务.虽然这些研究工作都对时空众包平台中的任务分配问题作出了研究,但大部分研究都只考虑用户和工人两种角色,与本研究中关注用户、工人和工作点的 3 种角色的问题模型具有很大的区别.

2.2 在线匹配

相较于离线算法,在线算法中无法事先获得用户和工人出现的时间以及位置,所以具有很强的不确定性.以经典的最大权二分图匹配问题为基础的在线匹配问题已经被广泛研究[14−16].近年来,研究人员也开使用在线算法来解决时空众包平台中的动态匹配问题.在 Hassan 等人[17]研究的问题中,用户提交任务是实时的,平台通过分析工人的历史接单行为和地理信息来为任务安排合适的工人.更进一步,Tong 等人[18]研究了工人和任务同时动态出现的场景,通过分析预测的方法指导工人的接单行为.文献[19]同时考虑距离和订单价值两个因素,提出一个以最大化收益和用户满意度的稳定在线匹配问题.文献[20]研究通过降低用户最长等待时间来提高用户的满意程度.

2.3 离线稳定婚姻匹配问题

在稳定婚姻问题[21]中,假设分别存在 n 个男人和女人,每个人对另一个性别的全体成员具有一个偏好排序.一个稳定的婚姻指不存在任意两对情侣,他们都喜欢对方的伴侣胜过当前的伴侣.研究证明:稳定婚姻问题的解

是永远存在的[22],但是这个算法的解是不公平的.文献[23–25]提出了多种稳定婚姻的衡量指标,以此来求得更公平的解.文献[26]提出算法来研究如何求出稳定婚姻的个数.稳定婚姻问题具有多种扩展版本,在其中一类扩展中,男人和女人之间的偏序可以是不完整的[27]、或者偏序中存在并列[28]、或者既不完整又存在并列[29].虽然这些研究都可以获得稳定匹配的结果,但这些问题都关注与二维的稳定匹配,不能推广到三维匹配问题,且不具备处理在线场景的能力.

3 基础算法

本节中,我们提出一个基于贪心的基础算法.由于不能准确获知工人和用户的到达顺序和地理位置,我们采用延迟匹配的思想,当用户提出的任务到达最长等待时间时,开始对任务进行处理.对于用户来讲,合适的工人可能出现在任务发布之后的某一时刻,所以当任务过期之前的所有时刻,都可能出现合适的工人.因此在延迟匹配的过程中,尽可能为用户扩大候选的工人集合,从而提高匹配的效果.在匹配过程中,根据用户到工作点的距离由近及远地遍历每个闲置的工作点;对于每个工作点,再根据距离遍历每个未匹配的工人.一旦找到满足稳定的元组,那么将该元组作为用户的匹配结果.

延迟匹配算法的伪代码如算法 1 所示.算法的输入是动态到达的用户集 U 和工人集 W 以及静态的工作点集 P,算法的输出是任务的匹配结果集 M,其中的每一个元素都是稳点元组.在初始状态,匹配结果 M 为空(第 1 行),将所有工作点设置为闲置状态(第 2 行).对于时刻 t,取出已经到达最长等待时间的用户集 U_t 和已经到达且未被分配任务的工人集 W_t(第 3 行、第 4 行).对于 U_t 中的每个用户 u,根据 D_u 由近及远地查找每个闲置工作点 p,再根据 DW_p 由近及远地查找 W_t 中的工人 w,一旦(w,u,p)构成一个稳定元组,那么将(w,u,p)作为一个分配结果加入到 M 中,将 w 从 W_t 和 W 中移除,并将 p 被更新为占用状态,也将从 P_t 中移除(第 5 行~第 19 行).每个被处理过的用户,不论是否被匹配,都将从 U 中移除(第 19 行).因为如果这个用户没有被匹配,也到达了最长等待时间,用户的任务也会过期.当循环结束时,当前时刻需要被匹配的用户都已被处理,算法将为下一时刻的用户进行匹配(第 21 行).

算法 1. 延迟匹配算法.

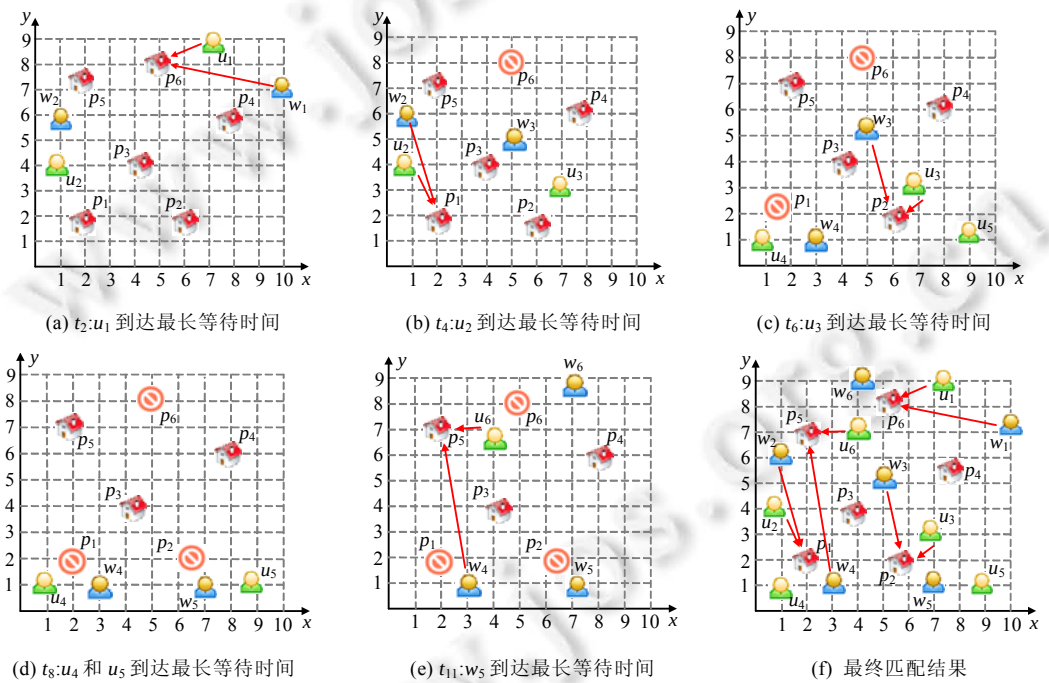
输入:用户集 U,工人集 W,工作点集 P;

输出:全局任务匹配结果集 M.

1. $M=\varnothing$;
2. 所有工作点处于闲置状态
3. **For** (当前时刻 t)
4. 取出到达最长等待时间的用户集 U_t、当前未分配任务的工人集 W_t、当前闲置的工作点集 P_t
5. **For** (每一个在 U_t 的用户 u)
6. **For** (每一个工作点 $p\in D_u$ 并且 $p\in P_t$)
7. **For** (每一个工人 $w\in DW_p$ 并且 $w\in W_t$)
8. **If** ((w,u,p)是稳定的)
9. 将(w,u,p)加入到 M
10. 将 w 从 W_t 和 W 中移除
11. 更新 p 为占用状态
12. **break**;
13. **End if**
14. **End for**
15. **If** (p 被更新为占用状态)
16. 将 p 从 P_t 中移除
17. **break**;

18. **End if**
19. **End for**
20. 将 u 从中 U 移除
21. **End For**
22. **End for**

例 4:如例 3 中的出现序列和坐标,假设每个用户的等待时间为 2 个时间段,算法 1 的执行过程和匹配结果如图 4 所示.首先,在 t_2 时刻,u_1 到达最长等待时间.此时可服务的工人为 w_1 和 w_2,算法执行的匹配元组是(w_1,u_1,p_6),如图 4(a)所示.在 t_4 时刻,u_2 到达最长等待时间,此时可服务的工人为 w_2 和 w_3,p_6 是被占用状态,算法执行的匹配元组是(w_2,u_2,p_1),如图 4(b)所示.在 t_6 时刻,u_3 到达最长等待时间,此时可服务的工人为 w_3 和 w_4,p_1 和 p_6 是被占用状态,算法执行的匹配元组是(w_3,u_3,p_2),如图 4(c)所示.在 t_8 时刻,u_4 和 u_5 到达最长等待时间,此时可服务的工人为 w_4 和 w_5, p_1、p_2 和 p_6 是被占用状态.由于 p_1 和 p_2 都被占用,无论当前怎么组合,算法都无法找到稳定元组,如图 4(d)所示.在 t_{11} 时刻,u_6 到达最长等待时间,此时可服务的工人为 p,w_5 和 w_6,p_1,p_2 和 p_6 是被占用状态,算法执行的匹配元组是(w_6,u_4,p_5),如图 4(e)所示.图 4(f)描述了该算法最终的匹配结果,成功匹配 4 个稳定元组.任务匹配率为 66.67%.



(a) t_2:u_1 到达最长等待时间　(b) t_4:u_2 到达最长等待时间　(c) t_6:u_3 到达最长等待时间

(d) t_8:u_4 和 u_5 到达最长等待时间　(e) t_{11}:w_5 到达最长等待时间　(f) 最终匹配结果

Fig.4 Execution process of the delay matching algorithm

图 4 延迟匹配算法的执行过程

算法复杂度分析:算法每个时刻需要执行一次,每次需要遍历用户集 U_t、工人集 W_t 和工作点集 P,复杂度是 $O(|U_t|\times|W_t|\times|P_t|)$,最差情况的复杂度为 $O(|U|\times|W|\times|P|)$.综上,延迟匹配算法的复杂度 $O(|U|\times|W|\times|P|)$.算法需要记录 3 种对象的信息,分别记录用户和工作点之间、工人和工作点之间的距离,空间复杂度为 $O(|U|\times|P|+|W|\times|P|)$.

4 改进算法

在上一节中提出了延迟匹配算法.延迟匹配算法以距离为核心,优先查找相近的任务点和工人.虽然该算法可以求出一定数量的稳定匹配,但是由于它无法预知将来时刻工人和用户的分布,仅根据当前到达情况进行匹

配,可能影响后续的匹配效果.比如在例 4 中,p_1 组成了(w_2,u_2,p_1),p_2 组成了(w_3,u_3,p_2),这个分配方式直接导致 u_4 和 u_5 找不到稳定匹配结果.如果将 w_2 和 u_2 分配到 p_5,将 w_3 和 u_3 分配到 p_3,那么所有的任务都可以被匹配,匹配率提高到了 100%.

由于以上的不足,本节提出了基于预测的匹配算法:首先,使用人工智能技术预测出工人和用户的到达时间区间和大致位置;其次,根据预测结果计算最大化匹配个数的结果;最后,根据预测结果和用户、工人的实际到达情况,计算真正的匹配结果.

4.1 离线预测

在预测方面,基于历史记录的预测方法已经被广泛研究[30−32].文献[13]通过实验结果证明,HP-MSI[32]具有较好的预测结果.基于以上的时间和空间划分,本文采用 HP-MSI 作为离线预测工具,根据历史记录信息预测出工人和用户的出现情况.对于每一个区域,它的工人/用户数量可以表示为一个时间序列,序列的长度为时间槽的个数,序列中的值代表数量.熵是表征可预测性程度的有效指标,包括随机熵、香农熵和真实熵.真实熵的公式如下所示:

$$S_{real}^{(i)} = -\sum_{S_n^{(i)} \subset D_n^{(i)}} P(S_n^{(i)}) \log_2[P(S_n^{(i)})].$$

其中,$D_n^{(i)}$ 表示区域 i 在 n 时刻的历史数量序,长度为 n;$S_n^{(i)}$ 是 $D_n^{(i)}$ 的有序子序列,$P(S_n^{(i)})$ 表示在 $D_n^{(i)}$ 中找到子序列 $S_n^{(i)}$ 的概率.HP-MSI 使用真实熵和移动的相关性来衡量区域级别的需求不确定.为预测区域中的数量,HP-SMI 采用马尔可夫预测器.预测器的输入是 $D_n^{(i)}$、时刻 n 和马尔可夫预测矩阵 T.输出是 n+1 时刻的预测数量.马尔可夫预测器可以在较低耗时下取得较高的预测准确率.

我们将时间和空间划分为若干个范围,每个时间段称之为“时间槽”,用户和工人的出现时间属于一个时间槽,等待时间为一个或多个时间槽.整个地图也被划分为多个区域,每个用户和工人都出现在一个区域内,处于区域交界处的用户和工人属于其左下角的区域,坐标按照区域的中心点计算.每个工人被表示为 $w_{i,j}^k$,每个用户被表示为 $u_{i,j}^k$.其中,i 表示区域的编号,j 表示时间槽的编号,k 表示在该区域该时间槽内的编号,从 1 开始.

例 5:对于表 1 中的出现顺序,将整个时间段分为两个时间槽:$[t_0,t_4]$和$[t_5,t_9]$,那么 $u_{1\text{-}3}$ 和 $w_{1\text{-}3}$ 属于时间槽 1,$u_{4\text{-}6}$ 和 $w_{4\text{-}6}$ 属于时间槽 2.整个地图按照 1×1 的方格划分为 90 个区域,从左下角开始编号,即 u_4 所处于的区域编号为 1,w_4 所处于的区域编号为 3.基于以上的划分,可以将预测的工人和用户进行划分,例如 u_4 可以被编号为 $u_{1,2}^1$,1 是区域编号,2 是时间槽编号,1 是该区域该时间槽内的编号.w_4 可以被编号为 $w_{3,2}^1$.

通过预测方法得出的结果,将工人和用户分别进行编号,然后求解全局的稳定匹配.根据证明,求解全局稳定匹配是 NP-hard 问题,可以采用最大团的方法求得准确,但时间复杂度极高.本文采用 DSM 算法进行求解[10].静态匹配采用合并和拆分的思想.首先采用贪心求解出一个初步的匹配结果.对于未被匹配的用户,如果拆掉当前的某个匹配再将拆离的用户递归匹配,如果可以使得匹配总数增加,那么当前的匹配状态将会被调整.由于原算法并不考虑时间属性,调用过程中需要检查工人出现的时间是否可以覆盖用户出现的时间槽,如果工人出现过晚,则不能匹配.

4.2 在线匹配

在线匹配的主要思想是:根据离线匹配的结果,为当前到达的工人或用户进行匹配.如果当前到达用户是预测结果之外的,那么让这个用户延迟到最长等待时间时,到达这个时间点时,根据当前的实际情况,采用延迟匹配算法进行匹配.在线匹配算法分为两个部分,当用户(或工人)到达时,算法首先在离线匹配结果中查找是是否存在相关的匹配,如果存在,则直接进行匹配或等待到查找到了离线匹配;如果离线结果中查找不到结果且用户等待到达最长等待时间,但仍没有被匹配,这种情况是由于预测或离线匹配的误差所导致,对于类似情况的用户,在不影响离线匹配结果的基础上,调用延迟匹配算法进行求解.

预测指导的匹配算法的伪代码如算法 2 所示.算法的输入是动态到达的用户集 U 和工人集 W、静态的工

作点集 P 以及离线预测的结果.算法的输出是任务的匹配结果集 M,其中的每一个元素都是稳点元组.在初始状态,匹配结果 M 为空(第 1 行),将所有工作点设置为闲置状态(第 2 行).对于时刻 t,当一个新的对象 o 出现时,从离线匹配结果 M_{off} 中查找和 o 有关的元组(第 3 行~第 5 行).如果这个元组在实际情况下是稳定的,将(w,u,p)移出 M_{off},作为一个分配结果加入到 M 中,将 u 从 U 中移除,将 w 从 W 中移除,并将 p 被更新为占用状态(第 6 行~第 10 行).如果在 M_{off} 中不存在有关 o 的匹配结果,则原地等待(第 10 行~第 12 行).对于当前到达最长等待时间的用户,则调用延迟匹配算法在同样等待的工人中查找是否具有稳定匹配(第 14 行~第 17 行).由于预测产生的误差,会不可避免地出现一些意料之外的工人和用户,如果忽略这种情况,只按照预测的结果进行分配,会降低任务匹配的数量,所以在预测指导之下,还需要再根据实际情况对分配结果进行调整.最后,算法会更新 M_{off} 并移除那些已经过期的预测结果.

算法 2. 预测指导的匹配算法.

输入:用户集 U,事件集 W,工作点集 P,离线匹配结果 M_{off};

输出:全局任务匹配结果 M.

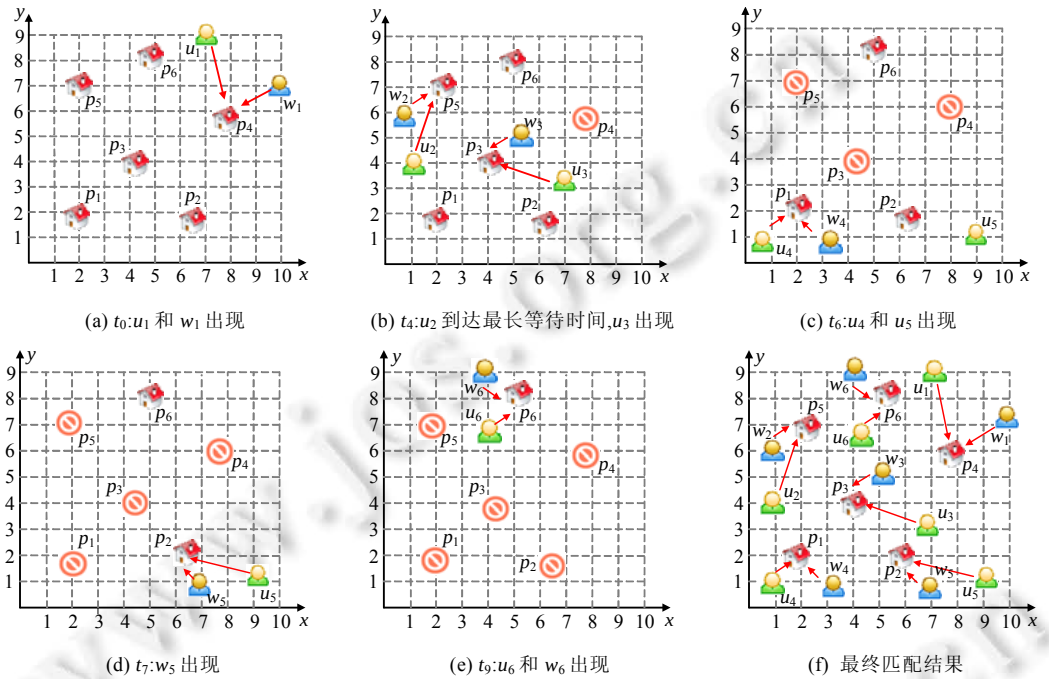
1. $M=\varnothing$
2. 所有工作点处于闲置状态
3. **For** (当前时刻 t)
1. **If** (出现一个新的对象 o) //o 的类型可以是工人也可以是用户
5. 在 M_{off} 中查找与 o 有关元组(w,u,p)
6. **If** ((w,u,p)存在)
7. 将(w,u,p)从 M_{off} 中移除
8. 将(w,u,p)加入 M
9. 将 u 从 U 中移除,w 从 W 中移除,更新 p 为占用状态
10. **Else**
11. o 原地等待
12. **Endif**
13. **Endif**
14. **If** (到达最长等待时间的用户集 $U_t\neq\varnothing$)
15. 取出当前未分配任务且不属于 M_{off} 的工人集 W_t、当前闲置且不属于 M_{off} 的工作点集 P_t
16. 调用延迟匹配算法
17. **Endif**
18. 删除 M_{off} 中的过期结果
19. **Endfor**

例 6:按照例 5 中的划分方法,可以得到 90 个区域和 2 个时间槽,表 2 描述了预测方法所预测出的结果和离线匹配结果,共匹配了 5 个任务.首先,在 t_0 时刻,w_1 和 u_1 出现,此时 w_1 对应 $w^1_{70,1}$,u_1 对应 $u^1_{87,1}$,(w_1,u_1,p_6)可构成一个稳定元组,如图 5(a)所示.在 t_2 时刻,u_2 出现,但 M_{off} 中不存在与 u_2 对应的元组,所以 u_2 需要等待.同理,w_2 出现之后依旧需要等待.在 t_4 时刻,w_3 和 u_3 出现,此时 w_3 对应 $w^1_{25,1}$,u_3 对应 $u^1_{27,1}$,(w_3,u_2,p_3)可构成一个稳定元组.与此同时,u_2 到达最长等待时间,由于 w_3 在 M_{off} 存在有关元组,此时可服务的工人为 w_2,可供选择的工作点是 p_5,(w_2,u_2,p_5)可构成一个稳定元组,如图 5(b)所示.在 t_5 时刻,w_4 出现并开始等待.在 t_6 时刻,u_4 和 u_5 出现,此时 w_4 对应 $w^1_{3,2}$,u_4 对应 $u^1_{1,2}$,(w_4,u_4,p_1)可构成一个稳定元组.u_5 开始等待,如图 5(c)所示.在 t_7 时刻,w_5 出现,w_5 对应 $w^1_{7,2}$,u_5 对应 $u^1_{9,2}$,(w_5,u_5,p_2)可构成一个稳定元组,如图 5(d)所示.在 t_9 时刻, w_6 和 u_6 出现,此时 w_6 对应 $w^1_{84,2}$,u_6 对应 $u^1_{64,2}$,(w_6,u_6,p_6)可构成一个稳定元组.

Table 2 Matching results of the offline guide

表 2 离线匹配结果

$(w^1_{70,1},u^1_{87,1},p_4)$	$(w^1_{25,1},u^1_{27,1},p_3)$	$(w^1_{3,2},u^1_{1,2},p_1)$	$(w^1_{7,2},u^1_{9,2},p_2)$	$(w^1_{84,2},u^1_{64,2},p_6)$



(a) t_0:u_1 和 w_1 出现 (b) t_4:u_2 到达最长等待时间,u_3 出现 (c) t_6:u_4 和 u_5 出现

(d) t_7:w_5 出现 (e) t_9:u_6 和 w_6 出现 (f) 最终匹配结果

Fig.5 Execution process of the prediction-oriented matching algorithm

图 5 预测指导的匹配算法执行结果

算法复杂度分析:算法每个时刻需要执行一次,算法 M_{off} 中使用索引可快速查询相应的匹配结果,复杂度为 $O(1)$,对于所有用户和工人最优的匹配复杂度可以达到 $O(|U|+|W|)$.对到达最长等到时间的用户调用延迟匹配算法的最坏时间复杂度是 $O(|U|\times|W|\times|P|)$.更新 M_{off} 需要对其进行遍历,复杂度是 $O(M_{off})$.综上所述,算法最差复杂度是 $O(|U|\times|W|\times|P|)$.算法需要记录 3 种对象的信息,分别记录用户和工作点之间、工人和工作点之间的距离,还需要记录离线匹配结果,空间复杂度为 $O(|U|\times|P|+|W|\times|P|)$.

5 实验及分析

本节报告文中算法的实验结果,在真实数据集和合成数据集的情况下,分别对算法的效率和效用进行评估.

5.1 实验设置

本文采用滴滴盖亚数据开放计划(https://outreach.didichuxing.com/research/opendata/)作为测试的真实数据集.滴滴出行是国内著名的出行服务平台,盖亚数据开放计划提供了真实脱敏的数据源,以天为单位,记录平台中的出行信息.数据包括订单文件和轨迹文件:订单文件记录订单的起止时间和起止位置,轨迹文件按照时间记录司机的行车位置.本文选取 4 天、相同区域的订单作为真实数据集,根据订单信息和轨迹信息选取工人和用户的时空信息,出现在不同位置的同一司机被视为两个不同的工人,根据订单起始位置选取一些位置作为工作点.时间槽设定为 15 分钟,即:全天被划分为 96 个时间槽,用户的最长等待时间是 15 分钟,区域大小为 0.5km×0.5km.相同区域不同时间的数据被作为预测算法的训练数据.真实数据集的详细信息见表 3.

Table 3 Real dataset

表 3 真实数据集

日期	工人	用户	工作点
11.09	9 425	9 755	9 900
11.16	10 657	11 523	12 000
11.23	9 147	9 062	9 200
11.30	9 871	10 894	11 000

本文还通过一组合成数据集测试算法的可扩展性.首先,通过划定不同大小区域,调整工人集、用户集和工作点集的大小.通过调整时间槽的时间跨度、用户等待时间的长短和区域大小来验证这 3 个参数对算法效率的影响.合成数据集的详细信息见表 4,实验默认设置已经加粗显示.

Table 4 Synthetic dataset

表 4 合成数据集

参数	范围
$\|W\|=\|U\|=\|P\|$	10k,**20k**,30k,50k,100k
等待时间(分钟)	5,10,**15**,20,30
时间槽(分钟)	5,10,**15**,20,30
区域大小(千米)	0.1×0.1,0.2×0.2,**0.5×0.5**,1.0×1.0

实验环境采用的 CPU 为 Intel(R) Core(TM) i5-6500 3.20GHz CPU 以及 32GB 内存,算法通过 C++和 STL 库实现.评价的算法包括延迟匹配算法(DM)和预测指导的匹配算法(POM),此外,将静态的三维稳定匹配算法[10]作为最优解(OPT)进行比较.算法的评价指标包括任务匹配个数、运行时间和占用内存,预测和离线匹配的运行时间和内存不计算在内.

5.2 真实数据集的实验结果

本节报告 3 种算法在真实数据集上的测试结果见表 5.

- 在 4 组不同日期的数据上,OPT 算法耗时最多,并且由于使用索引,导致占用内存最大;
- POM 算法的匹配结果次之,耗费时间最短,但占用内存略高于 DM 算法;
- DM 算法的匹配结果较差,运行时间高于 POM 算法,但是占内存最少.

实验结果说明预测技术对本文的问题具有指导作用,能够在一定程度上预测工人和用户的时空分布,从而提高在线匹配的效果.由于 POM 首先进行离线匹配结果的查找,查找失败再进行 DM 的匹配,算法耗时较少.由于 POM 保存了离线分配结果等,而 DM 算法只保存了基本的信息,导致了 POM 算法内存占用高于 DM 算法.

Table 5 Results over real datasets

表 5 真实数据集实验结果

数据集	算法	匹配率(%)	运行时间(s)	内存(MB)
11.09	OPT	84.54	45.88	204.47
	DM	38.34	12.48	50.67
	POM	62.14	9.67	127.88
11.16	OPT	84.96	57.23	278.53
	DM	55.30	15.79	57.87
	POM	64.99	11.81	159.96
11.23	OPT	81.83	44.12	188.75
	DM	34.40	11.75	32.78
	POM	58.20	7.14	107.67
11.30	OPT	86.86	49.66	218.67
	DM	48.13	13.74	58.95
	POM	67.34	8.47	130.36

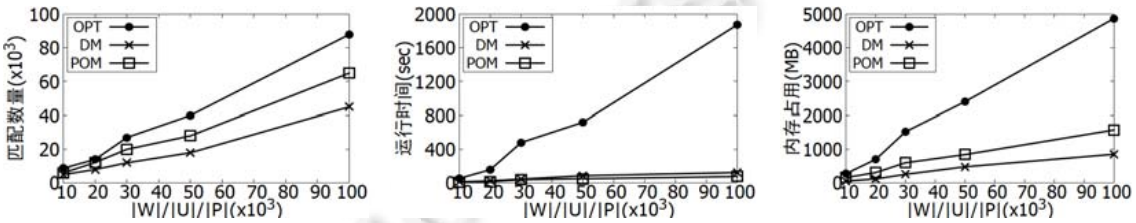
5.3 合成数据集的实验结果

本节在合成数据上研究不同参数对 3 个算法的影响,包括工人数量、用户数量和工作点数量、用户等待时

间、时间槽长度和区域大小.

首先报告工人数量、用户数量和工作点数量变化对算法结果的影响.实验的设置是用户等待时间 15 分钟,时间槽长度 15 分钟,区域大小是 0.5km×0.5km,工人、用户和工作点数量从 10k 扩展到 100k.

- 3 种算法匹配数量的结果如图 6(a)所示:随着工人数量、用户数量和工作点数量的不断增加,3 种算法的匹配数量不断上升,POM 算法的匹配效果一直高于 DM 算法,低于 OPT 算法;
- 算法的运行时间如图 6(b)所示:OPT 算法的运行时间远远高于其他算法,POM 算法的运行时间略低于 DM 算法,且两种算法的运行时间增幅较缓;
- 算法的内存占用如图 6(c)所示:OPT 算法由于索引的使用内存占用迅速增加,POM 算法和 DM 算法的内存占用都有增加,且 POM 算法的内存占用高于 DM 算法.



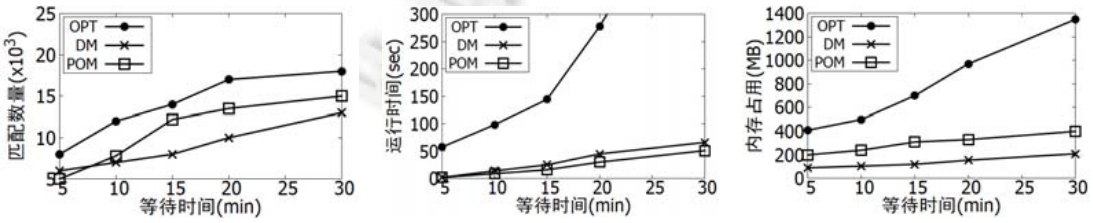
(a) 匹配数量 (b) 运行时间 (c) 内存占用

Fig.6 Results on varying $|W|$, $|U|$ and $|P|$

图 6 工人数量、用户数量和工作点数量变化的实验结果

用户等待时间变化对算法的影响如图 7 所示.算法的默认设置是时间槽长度 15 分钟,区域大小是 0.5km×0.5km,工人、用户和工作点的数量是 20k,用户等待时间从 5 分钟变化到 30 分钟.

- 3 种算法的匹配数量都随着等待之间的延长而多变,如图 7(a)所示:OPT 算法的匹配数量一直最多,POM 算法在等待时间短时的结果差于 DM 算法.随着等待时间边长,POM 效果高于 DM 算法.原因是当等待时间小于时间槽长度时,离线匹配结果可能提前失效,要调用 DM 算法来为这些用户进行在线分配,但是很多角色还在离线匹配结果中,导致可用的工人和工作点很少,匹配结果不理想.随着等待时间逐渐边长,这种提前失效的情况消失,匹配结果变好;
- 算法的运行时间如图 7(b)所示:OPT 算法时间消耗上升较快;在等待时间远低于时间槽长度时,POM 算法的运行时间和 DM 算法差不多,原因和匹配结果一致.随着等待时间变长,POM 算法的运行时间低于 DM 算法;
- 算法的内存占用如图 7(c)所示:由于索引开销加大,OPT 算法的内存消耗持续上涨,POM 算法和 DM 算法的内存消耗小幅上涨,这是在线分配结果和可选工人数量变多的缘故.等待时间的变长不影响离线匹配的结果,所以 POM 算法的内存增长和 DM 算法趋势一致.



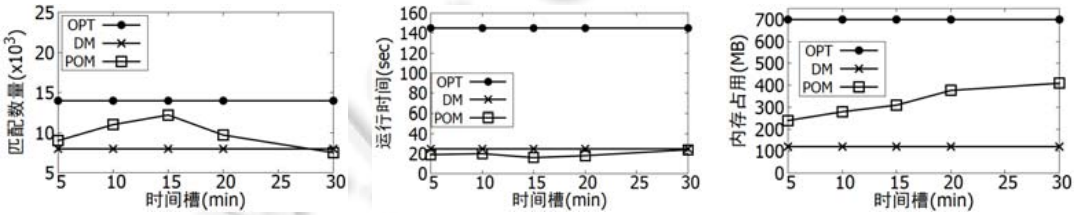
(a) 匹配数量 (b) 运行时间 (c) 内存占用

Fig.7 Results on varying T_u

图 7 用户等待时间变化的实验结果

时间槽长度变化对算法的影响如图 8 所示.算法的默认设置是区域大小是 0.5km×0.5km,工人、用户和工作点的数量是 20k,用户等待时间 15 分钟,时间槽长度从 5 分钟变化到 30 分钟.由于 OPT 算法和 DM 算法的效果与时间槽的长度无关,所以两种算法的曲线都保持直线,时间槽的长度影响预测结果,从而影响 POM 算法,所以 POM 算法的曲线根据参数变化而变化.

- POM 算法的匹配结果如图 8(a)所示:匹配数量随着时间槽的增长呈先上升后下降的趋势.原因是当时间槽过短时,预测的准确性降低,离线匹配结果对在线匹配的指导性不强,在线匹配过程中无法在离线结果中找到合适的匹配结果,算法效果类似于 DM;当时间槽长度超过最长等待时间时,用提前失效现象会发生,即离线匹配中的结果会由于用户超过等待时间而失效,在线匹配只能从当前空闲的资源中查找匹配,导致匹配的数量开始下降;
- 算法运行时间如图 8(b)所示:当时间槽长度和用户等待时间相近时, POM 算法的运行时间最短,其他时间由于多次调用 DM 算法导致时间上升;
- 算法的内存占用如图 8(c)所示:由于时间槽长度增加,离线匹配结果增加,导致算法存储空间增大.

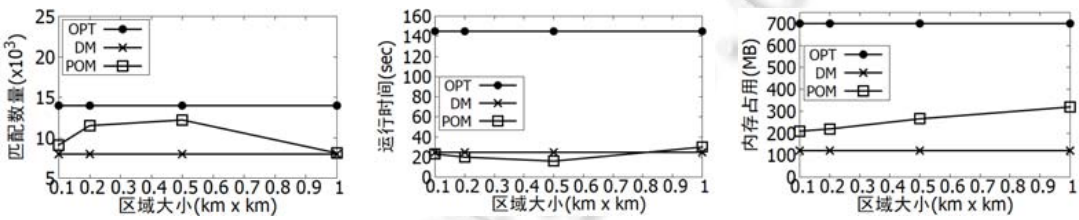


(a) 匹配数量 (b) 运行时间 (c) 内存占用

Fig.8 Results on varying length of slots

图 8 时间槽长度变化的实验结果

区域大小变化对算法的影响如图 9 所示.3 种角色的数量是 20k,用户等待时间 15 分钟,时间槽长度 15 分钟,划分区域大小从 0.1km×0.1km 变化到 1.0km×1.0km.如时间槽长度变化的实验分析中所提到的,OPT 算法和 DM 算法与区域大小划分无关,所以两种算法的曲线都保持直线,划分区域大小影响预测结果,从而影响 POM 算法,所以 POM 算法的曲线根据参数变化而变化.



(a) 匹配数量 (b) 运行时间 (c) 内存占用

Fig.9 Results on varying grid size

图 9 区域大小变化的实验结果

- POM 算法的匹配结果如图 9(a)所示:随着划分区域面积的不断增大,POM 算法的匹配结果呈先上升后下降的趋势.这个趋势和时间槽的划分有相似之处,但是原因不同.当区域过小时,预测准确性降低,此时无法起到指导作用,离线匹配中查找不到合适的结果,所以需要对当前闲置资源调用 DM 算法进行匹配.当区域过大时,预测算法将产生大量预测结果,但是由于区域划分过大,导致在线分配时这些结果不一定都是稳定的,找不到稳定匹配的用户要调用 DM 算法进行处理,此时和用户提前失效的结果相似;

- 算法的运行时间如图 9(b)所示:POM 算法的运行时间呈先减少后上升的趋势.在区域过大或过小时,算法大部分时间调用 DM 算法,导致运行时间升高,和 DM 算法的运行时间相近;
- 算法的内存占用如图 9(c)所示:由于区域划分的增大,离线匹配结果增大,离线匹配结果增加,导致 POM 算法存储空间增大.原因和时间槽增大的情况类似.

5.4 实验总结

在本节中,我们使用真实数据集和合成数据集分别对算法的效用和效率进行了测试.在合理的设置下,POM 算法的实验结果和运行速度都高于 DM 算法,但是算法占用内存高与 DM 算法.在不同参数的测试中,DM 和 POM 算法都受到数据规模和用户等待时间的影响.随着数据规模的增长,两种算法的匹配数量、运行时间和内存消耗都进行增长.随着等待时间的延长,两种算法的匹配数量、运行时间和内存消耗也都进行增长,但是在时间过长或过短时,都会影响 POM 算法的效果.POM 算法还受时间槽长度和划分区域大小所影响.时间槽长度和用户等待时间的差距影响了 POM 算法的效果,当时间槽长度和用户等待时间相近时,POM 的效果较好.区域划分的粒度通过影响预测结果和离线匹配,从而影响 POM 的效果.在本实验中,采用 0.5km×0.5km 的划分时 POM 获得较好的效果.

6 结论及展望

本文研究新型时空众包平台中在线稳定匹配问题.区别于传统的研究工作,本文考虑工人、用户和工作点这 3 种角色,将二维匹配问题扩展到三维问题,同时研究如何求解在线场景下的稳定匹配问题.本文首先给出了三维在线稳定匹配问题的定义,然后提出一种基础算法(延时匹配算法)和一种改进算法(预测指导的匹配算法)对问题进行求解.延时匹配算法当用户到达最长等待时间时,根据当前到达的工人和闲置的工作点进行匹配.预测指导的匹配算法采用 HP-MSI 方法对数据分布进行预测,并生成离线匹配结果,算法根据离线匹配结果对在线匹配进行指导.最后,本文使用真实数据集和合成数据集对算法进行实验.实验结果验证了算法的效果和效率,以及算法的可扩展性和不同参数对算法的影响.实验结果表明:在合理的参数配置下,预测指导的匹配算法的匹配结果和运行时间皆优于延时匹配算法.

未来工作中,将考虑用户和工人的活动距离约束,即用户和工人可以选择可接受的服务点的距离范围,防止出现远距离移动的情况.同时,为用户的任务设置服务时间,工作点在服务中处于占用状态,服务完成后将被释放,供其他用户使用.

References:

[1] Tong YX, Yuan Y, Cheng YR, Chen L, Wang GR. Survey on spatiotemporal crowdsourced data management techniques. Ruan Jian Xue Bao/Journal of Software, 2017,28(1):35−58 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/28/35.htm [doi: 10.13328/j.cnki.jos.005140]

[2] Kazemi L, Shahabi C. GeoCrowd: Enabling query answering with spatial crowdsourcing. In: Proc. of the Int'l Conf. on Advances in Geographic Information Systems. Los Angeles: ACM, 2012. 189−198.

[3] To H, Shahabi C, Kazemi L. A server-assigned spatial crowdsourcing framework. ACM Trans. on Spatial Algorithms and Systems, 2015,1(1):2:1−2:28.

[4] Zheng L, Chen L. Mutual benefit aware task assignment in a bipartite labor market. In: Proc. of the 32nd IEEE Int'l Conf. on Data Engineering. Helsinki: IEEE, 2016. 73−84.

[5] Song TS, Tong YX, Wang LB, She JY, Yao B, Chen L, Xu K. Trichromatic online matching in real-time spatial crowdsourcing. In: Proc. of the 33rd IEEE Int'l Conf. on Data Engineering. San Diego: IEEE, 2017. 1009−1020.

[6] Li BY, Cheng YR, Yuan Y, Wang GR, Chen L. Three-Dimensional stable matching problem for spatial crowdsourcing platforms. In: Proc. of the 25th ACM SIGKDD Conf. on Knowledge Discovery and Data Mining. Anchorage: ACM, 2019. 1643−1653.

[7] Xing YP, Wang LM, Li ZY, Zhan YZ. Multi-Attribute crowdsourcing task assignment with stability and satisfactory. IEEE Access, 2019,7:133351−133361.

[8] Zhang XL, Yang Z, Liu YH, Tang SH. On reliable task assignment for spatial crowdsourcing. IEEE Trans. on Emerging Topics Comput, 2019,7(1):174−186.

[9] Deng D, Shahabi C, Demiryurek U. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In: Proc. of the 21st ACM SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems. Orlando: ACM, 2013. 314−323.

[10] Tao Q, Zeng YX, Zhou ZM, Tong YX, Chen L, Xu K. Multi-Worker-Aware task planning in real-time spatial crowdsourcing. In: Proc. of the 23rd Int'l Conf. on Database Systems for Advanced Applications. Gold Coast: Springer-Verlag, 2018. 301−317.

[11] Cheng YR, Yuan Y, Chen L, Giraud-Carrier CG, Wang GR. Complex event-participant planning and its incremental variant. In: Proc. of the 33rd IEEE Int'l Conf. on Data Engineering. San Diego: IEEE, 2017. 859−870.

[12] She JY, Tong YX, Chen L, Song TS. Feedback-Aware social event-participant arrangement. In: Proc. of the 2017 ACM Int'l Conf. on Management of Data. Chicago: ACM, 2017. 851−865.

[13] Cheng YR, Wang GR, Li BY, Yuan Y. Bilateral preference stable planning over event based social networks. Ruan Jian Xue Bao/ Journal of Software, 2019,30(3):573−588 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/30/573.htm [doi: 10.13328/j.cnki.jos.005682]

[14] Karp RM, Vazirani UV, Vazirani VV. An optimal algorithm for on-line bipartite matching. In: Proc. of the 22nd Annual ACM Symp. on Theory of Computing. Baltimore: ACM, 1990. 352−358.

[15] Mehta A. Online matching and ad allocation. Foundations and Trends in Theoretical Computer Science, 2013,8(4):265−368.

[16] Ting HF, Xiang XZ. Near optimal algorithms for online maximum edge-weighted b-matching and two-sided vertex-weighted b-matching. Theoretical Computer Science, 2015,607:247−256.

[17] ul Hassan U, Curry E. A multi-armed bandit approach to online spatial task assignment. In: Proc. of the 2014 IEEE 11th Int'l Conf. on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Int'l Conf. on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops. Bali: IEEE, 2014. 212−219.

[18] Tong YX, Wang LB, Zhou ZM, Ding BL, Chen L, Ye JP, Xu K. Flexible online task assignment in real-time spatial data. PVLDB, 2017,10(11):1334−1345.

[19] Zhao BM, Xu P, Shi YX, Tong YX, Zhou ZM, Zeng YX. Preference-Aware task assignment in on-demand taxi dispatching: An online stable matching approach. In: Proc. of the 33rd AAAI Conf. on Artificial Intelligence. Honolulu: AAAI, 2019. 2245−2252.

[20] Chen Z, Cheng P, Zeng YX, Chen L. Minimizing maximum delay of task assignment in spatial crowdsourcing. In: Proc. of the 35th IEEE Int'l Conf. on Data Engineering. Macao: IEEE, 2019. 1454−1465.

[21] Gale D, Shapley LS. College admissions and the stability of marriage. The American Mathematical Monthly, 2013,120(5): 386−391.

[22] Roth AE. Deferred acceptance algorithms: History, theory, practice, and open questions. Int'l Journal of Game Theory, 2008, 36(3-4):537−569.

[23] Feder T. A new fixed point approach for stable networks and stable marriages. In: Proc. of the 21st Annual ACM Symp. on Theory of Computing. Seattle: ACM, 1989. 513−522.

[24] Iwama K, Miyazaki S, Yanagisawa H. Approximation algorithms for the sex-equal stable marriage problem. ACM Trans. on Algorithms, 2010,7(1):2:1−2:17.

[25] Irving RW, Leather P, Gusfield D. An efficient algorithm for the "optimal" stable marriage. Journal of ACM, 1987,34(3):532−543.

[26] Thurber EG. Concerning the maximum number of stable matchings in the stable marriage problem. Discrete Mathematics, 2002, 248(1-3):195−219.

[27] Gale D, Sotomayor M. Some remarks on the stable matching problem. Discrete Applied Mathematics, 1985,11(3):223−232.

[28] Irving RW. Stable marriage and indifference. Discrete Applied Mathematics, 1994,48(3):261−272.

[29] Iwama K, Miyazaki S, Yamauchi N. A 1.875-approximation algorithm for the stable marriage problem. In: Proc. of the 18th Annual ACM-SIAM Symp. on Discrete Algorithms. New Orleans: ACM, 2007. 902−914.

[30] Hendawi AM, Mokbel MF. Predictive spatio-temporal queries: A comprehensive survey and future directions. In: Proc. of the 1st ACM SIGSPATIAL Int'l Workshop on Mobile Geographic Information Systems. Redondo Beach: ACM, 2012. 97−104.

[31] Sun JM, Papadias D, Tao YF, Liu B. Querying about the past, the present, and the future in spatio-temporal. In: Proc. of the 20th Int'l Conf. on Data Engineering. Boston: IEEE, 2014. 202−213.

[32] Zhao K, Khryashchev D, Freire J, Silva CT, Vo HT. Predicting taxi demand at high spatial resolution: Approaching the limit of predictability. In: Proc. of the 2016 IEEE Int'l Conf. on Big Data. Washington: IEEE, 2016. 833−842.

附中文参考文献:

[1] 童咏昕,袁野,成雨蓉,陈雷,王国仁.时空众包数据管理技术研究综述.软件学报,2017,28(1):35−58. http://www.jos.org.cn/1000-9825/28/35.htm [doi: 10.13328/j.cnki.jos.005140]

[13] 成雨蓉,王国仁,李博扬,袁野.基于事件的社交网络上的双边偏好稳态规划.软件学报,2019,30(3):573−588. http://www.jos.org.cn/1000-9825/30/573.htm [doi: 10.13328/j.cnki.jos.005682]

李博扬(1992－),男,博士生,CCF 学生会员,主要研究领域为社交网络数据分析,时空数据分析,机器学习.

成雨蓉(1989－),女,博士后,CCF 专业会员,主要研究领域为图数据查询处理与分析,时空众包数据分析,社交网络数据分析.

王国仁(1966－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为不确定数据管理,数据密集型计算,可视媒体数据管理与分析,非结构化数据管理,分布式查询处理与优化技术(主要包括传感器网络和 P2P 对等计算),生物信息学.

袁野(1981－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为云计算,大数据管理(包括图数据管理,不确定数据管理,数据隐私保护),P2P 计算.

孙永佼(1983－),男,博士,副教授,CCF 专业会员,主要研究领域为云计算,大数据管理,数据分析,机器学习,不确定数据管理,分布式数据管理.