软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(3):824−838 [doi: 10.13328/j.cnki.jos.005453]

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

基于树分解的空间众包最优任务分配算法*

李 洋, 贾梦迪, 杨文彦, 赵 艳, 郑 凯

(苏州大学 计算机科学与技术学院,江苏 苏州 215006)

通讯作者: 郑凯, E-mail: zhengkai@suda.edu.cn

摘 要: 随着配备高保真传感器的移动设备的普及以及无线网络资费的快速下降,空间众包作为一种问题解决框架被用于解决将位置相关的任务(如路况报告、食品配送)分配给工人(配备智能设备并愿意完成任务的人)的问题.研究空间众包中最优任务分配问题,关键在于设计出将每个任务分配给最合适的工人的任务分配策略,以使得完成的总任务数目最大化,而所有的工人可以在完成所分配的任务后,在预期最晚工作时间之前返回起点.找到全局最优分配是一个棘手的问题,因为该问题不等于单个工人的最佳分配的简单累加.注意到,仅有部分工人存在任务依赖,因此利用树分解技术将工人分割成独立的集合,并提出一种带启发式的深度优先搜索算法,该算法可以快速地更新启发函数界限,从而高效地对不可能成为最优解的分配方案尽早地进行剪枝.实验结果表明:所提出的方法是非常有效的,可以很好地解决最优任务分配问题.

关键词: 空间众包;任务分配;任务依赖;树分解;最优解算法

中图法分类号: TP311

中文引用格式: 李洋,贾梦迪,杨文彦,赵艳,郑凯.基于树分解的空间众包最优任务分配算法.软件学报,2018,29(3):824−838. http://www.jos.org.cn/1000-9825/5453.htm

英文引用格式: Li Y, Jia MD, Yang WY, Zhao Y, Zheng K. Optimal task assignment algorithm based on tree-decouple in spatial crowdsourcing. Ruan Jian Xue Bao/Journal of Software, 2018,29(3):824−838 (in Chinese). http://www.jos.org.cn/1000-9825/5453.htm

Optimal Task Assignment Algorithm Based on Tree-Decouple in Spatial Crowdsourcing

LI Yang, JIA Meng-Di, YANG Wen-Yan, ZHAO Yan, ZHENG Kai

(School of Computer Science and Technology Soochow University, Suzhou 215006, China)

Abstract: The ubiquity of mobile devices with high-fidelity sensors and the sharp decreases in the cost of ultra-broadband wireless network flourish the market of spatial crowdsourcing, which has been proposed as a new framework to assign location-aware tasks (e.g., reporting road traffic, delivering food) to workers (i.e., persons equipped with smart device and willing to perform tasks). This paper studies the task assignment problem that concerns the optimal strategy of assigning each task to proper worker such that the total number of completed tasks can be maximized while all workers can go back to their starting point before expected deadlines after performing assigned tasks. It is an intractable problem since optimal assignment for individual worker does not necessarily lead to global optimal results. Observing that the task assignment dependency only exists amongst subsets of workers, this study utilizes tree-decomposition technique to separate workers into independent clusters and develops an efficient depth-first search algorithm with progressive bounds to prune non-promising assignments. Extended experiments demonstrate the effectiveness and efficiency of the proposed solution.

Key words: spatial crowdsourcing; task assignment; task dependency; tree-decomposition; optimal solution

随着智能设备迅速普及,移动资费的快速下降,人们随时随地携带各种传感器,参与多种多样的和位置相关的活动,如景点照片采集、路况监测等.在这样的大背景下,空间众包[1]的概念应运而生.空间众包要求参与者(通

本文由基于图结构的大数据分析与管理技术专刊特约编辑林学民教授、杜小勇教授、李翠平教授推荐.

收稿时间: 2017-08-01; 修改时间: 2017-09-05; 采用时间: 2017-11-07; jos 在线出版时间: 2017-12-05

CNKI 网络优先出版: 2017-12-06 15:37:04, http://kns.cnki.net/kcms/detail/11.2560.TP.20171206.1536.016.html

常称为工人)实际行驶到指定位置才能完成指定的任务(如拍摄某个景点的照片等).

空间众包领域的研究热点之一是,如何将任务高效地分配给工人.现有的研究目标包括最大化一次性任务匹配数量[2],实现单个工人最优任务调度方案[3],追求任务完成的质量和多样性[4],迭代进行任务分配和任务调度来最大化全局任务分配数量[5].现有工作大多基于共同的假设,即,所有工人都有一个固定大小的工作范围(比如以工人自身位置为圆心的固定半径的圆形区域)且工人最多只能完成固定数量的任务或者可以无限制地接受任务.这种假设在很多应用场景中是现实的,但本文也发现:在一些场景中,每个工人愿意完成任务的工作范围并不一样.比如在外卖配送工作中,有些送餐人员愿意接受更大的配送区域范围,而也有部分送餐人员只期望在所在小区附近接单.同时,本文注意到:工人通常不会给自己预先设定一个完成任务数量的上界,但是也不会无限制地接受任务.最常见的情况是,工人完成任务的情况依赖于工作时间的约束.以送餐人员为例,工人从家出发开始接单配送,但是通常会选择一个最晚回家的时间,比如 3 点前回家接孩子放学,或者 7 点前回家吃晚饭.再比如按天算钱的传单派发人员,从指定地点集合之后开始发单,在约定好的下班时间之前需要返回出发点集合结账.在该场景中,工人可以接受任务区域范围受预期最晚工作时间的限制,可接受的任务数量由区域内的任务数量和预期最晚工作时间决定.

本文研究基于以上场景的空间众包中的最优任务分配方案.具体来说,给定每个工人的位置和该工人预期的最晚工作时间,给定每个任务的位置和过期时间,寻找使得全局任务分配数量最大化的最优任务分配方案.

相比于现有工作,本问题的主要困难在于:一旦需要考虑工人行驶到任务的时间成本和任务的过期时间,局部最优并不一定能够产生全局最优解.本文的第 2 个挑战在于,工人可达的任务范围高度依赖于工人起始位置和工人的预期最晚工作时间.这使得无法通过设定一个固定的工作区域范围或者最多可以接受任务的数量来排除不可达任务[2,3,5].

针对该问题,本文提出一种精确解算法,用于寻找最大化全局任务分配数量的最优分配方案.本文的主要思想是:根据任务依赖关系(如两个工人均可以完成某个任务,则两者之间存在任务依赖),采用树分解[6]方式将工人分割到相互独立的工人集合中;然后,通过将工人集合作为节点索引成搜索树结构;最后,采用带启发式的深度优先遍历算法搜索最优解.

同时,本文所提算法结合多种优化策略,可以快速收缩搜索过程的上下界,从而进行高效地剪枝.与基于迭代[5]的方法相比,本文所提算法可以在搜索结束时获得最优解.

本文的主要贡献有 3 点.

1) 首次研究带有最晚工作时间约束的空间众包中任务分配问题,该模式中,工人具有不同的工作时间约束,同时需要考虑任务的过期时间限制,工人可以接受的任务数量不再是固定的;
2) 本文提出一种精确解算法,可以有效地寻找最优化的任务分配方案.该算法采用树分解技术分割不存在任务依赖的工人,同时采用有效的剪枝算法提高搜索效率;
3) 通过实验分析关键参数对本方案效率的影响.

1 问题定义

本节介绍服务器端指派任务模式(server assigned task,简称 SAT)下基于自愿形式的无冗余任务分配模式[2],表 1 列出了本文主要的符号说明.

定义 1(空间任务). 空间任务可由两元组 $s=\langle s.e,s.l\rangle$定义.其中,$s.l$ 为任务的位置,$s.e$ 为任务的过期时间.$s.l$ 是二维平面空间中的一个点(x,y).空间众包中只有当工人实际到达任务 s 指定位置 $s.l$ 才可以完成该任务.同时,考虑到任务过期时间,工人只有在任务 s 过期时间 $s.e$ 之前到达 $s.l$ 位置才可以完成任务.在本文所考虑的无冗余任务分配模式下,服务器只会将一个任务分配给单个工人.与以往工作[2−5]一样,本文假设工人在完成一个任务后会立刻前往下一个任务,完成任务本身所需要的时间忽略不计.

定义 2(工人). 工人泛指携带移动设备、同时自愿完成空间任务的人.工人可以由两元组 $s=\langle w.l,w.t\rangle$表示,其中,$w.l$ 为工人起始位置,$w.t$ 为工人预期最晚工作时间,工人需要在不晚于最晚工作时间返回起始位置.

工人的工作模式分在线和离线两种,当处于在线模式时,可以接受空间任务.一旦工人处于在线模式,他将会向服务器发起任务请求.请求中包含了工人的位置 $w.l$,工人的最晚预期工作时间 $w.t$.服务器将会同时考虑极短时间间隔内获取的所有工人和任务,然后做全局最优任务分配.最后返回分配给各个工人的任务序列.

Table 1 Summary of Notations

表 1 主要符号说明

符号	定义
s	空间任务
$s.l$	一个空间任务的位置
$s.e$	空间任务 s 的过期时间
w	工人
$w.l$	工人 w 的起始位置
$w.t$	工人 w 的预期最晚工作时间
R	一个任务序列
S_w	工人 w 的一个任务集
$VTS(w)$	工人 w 的有效任务集
$MVTS(w)$	工人 w 的极大有效任务集
$t(l)$	到达位置 l 的到达时间
$c(a,b)$	从 a 到 b 的行驶时间
A	一种空间任务分配

如图 1 所示,服务器获取当前的工人序列 $W=(w_1,w_2,\ldots,w_7)$,当前的任务集合 S(为简化起见,本文用编号表示具体的任务,如 $S=\{1,2,3,\ldots,16\}$).工人信息包含了工人位置信息,如 w_1 的当前位置是(3,4),以及预期最晚工作时间(服务器分配任务时的时间为 0,这里,w_1 的最晚工作时间为 5).每个任务包括任务的位置和任务的过期时间,如任务 s_1 的位置坐标为(3,5),任务的过期时间是 5.

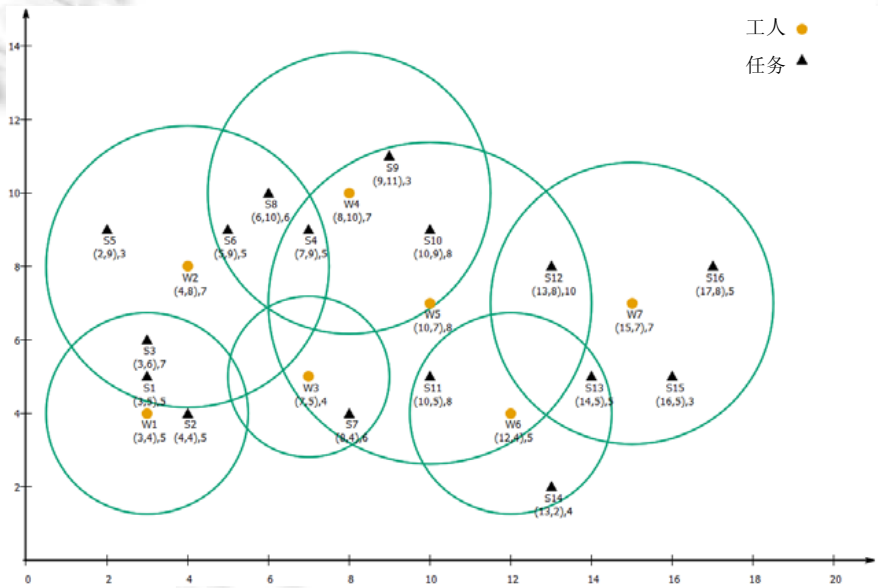


Fig.1 Example distribution of workers and tasks

图 1 工人和任务分布示例

定义 3(任务序列). 给定任意工人 w 及分配给 w 的任务集合 S_w,S_w 的任务序列表示为 $R(S_w)$,代表了工人访问每个任务的时间顺序.工人到达每个任务的时间 $t_{w,R}(s_i.l)$可通过公式(1)定义:

$$t_{w,R}(s_i.l)=\begin{cases}t(s_{i-1}.l)+c(s_{i-1}.l,s_i.l), & \text{if } i\neq 1\\ c(w.l,s_1.l), & \text{if } i=1\end{cases} \tag{1}$$

公式(1)中的 $c(a,b)$表示从任务 a 到达任务 b 的行驶时间,在 w 和 R 不会产生歧义的情况下,本文用 $t(s.l)$来表示工人 w 到达任务 s_i 所在位置 $s.l$ 的时间,用 $t(w.l)$表示工人 w 返回原始位置的时间.

工人在完成任务集合 S_w 中所有任务之后返回到起点的时间定义为

$$t(w.l)=t(s_{|S|}.l)+c(s_{|S|}.l,w.l) \tag{2}$$

因为行驶速度不在本文主要考虑因素范围内,为简化起见,本文假设所有工人的行驶速度相同,因此,工人的行驶时间开销可以用两个位置的欧氏距离定义.但是本文所提方法并不依赖于该假设,并且可以用于工人速度不相同的情况.

定义 4(有效任务集 *VTS*). 当且仅当以下条件成立时,任务集合 S_w 被称为工人 w 的有效任务集合:

1) S_w 中所有任务在其过期时间之前可以被完成,即$\forall s_i\in S_w, t(s_i.l)\leqslant s_i.e$;

2) 工人 w 可以在完成 S_w 中的所有任务后在不晚于最晚工作时间回到起点,即 $t(w.l)\leqslant w.t$.

例如在图 1 中,如果工人 w_1 按照任务序列(1,3),则可以完成两个任务,因为 $t(s_1.l)=1<s_1.e, t(s_3.l)=1+1=2<s_3.e$, $t(w_1.l)=2+2=4<w_1.t$,则{1,3}是 w_1 的其中一个有效任务集.

定义 5(极大有效任务集 *MVTS*). 如果有效任务集合 S_w 的任意超集都不是有效任务集合时,则称 S_w 为极大有效任务集.

如图 1 中,{1},{3}和{1,3}都是工人 w_1 的有效任务集,但是{1}和{3}不是 w_1 的极大有效任务集,因为他们都被有效任务集合{1,3}所包含.由定义可知,极大有效任务集并不唯一.

定义 6(空间任务分配). 给定工人集合 W 和任务集合 S,空间任务分配 A 包含一系列⟨工人,*VTS*⟩元组,形如 $A=\{\langle w_1,VTS(w_1)\rangle,\langle w_2,VTS(w_2)\rangle,\ldots,\langle w_{|S|},VTS(w_{|S|})\rangle\}$.设 $A.S$ 表示分配给所有工人的任务集合,即 $A.S=\bigcup_{w\in W}S_w$,本文所提问题归纳如下.

问题定义. 给定工人集合 W,任务集合 S,最晚工作时间限制下的空间众包任务分配问题,目标是找到一个全局最优任务分配方案 A_{opt},使得$\forall A_i\in A, |A_i.S|\leqslant|A_{opt}.S|$,其中,$A$ 表示所有的空间任务分配方案.

2 算 法

现有工作[2]已经证明,空间众包中的全局最优任务分配方案是 NP 难问题.因此,最简单的方法是使用贪心算法,依次为工人寻找最大有效任务集合,然后对已分配任务数量进行累加.

该方法的主要问题是:对于那些可以被多个工人完成的任务,如果简单随机指派给其中某一个工人,可能会导致其他的任务无法分配.如图 1 中:如果将任务 s_8 分配给 w_2,则任务 s_5 无法分配给任何人;而如果将任务 s_5 分配给 w_2,任务 s_8 很有可能被其他工人完成.本文提出一种最大化任务分配数量的精确解算法,通过 3 个步骤来解决以上问题:第 1 步,本文采用动态规划方法为每一个工人找到所有极大有效任务集,最优分配方案是每个工人极大有效任务集的某种组合(如果极大有效任务集中某个任务已被分配,则分配时要相应减去该任务);第 2 步,为避免穷举所有极大有效任务集的全部组合情况,本文采用树分割技术将不相关工人分割到不同的工人集合中,将这些工人集合使用搜索树结构进行索引;最后,本文采用带启发式的深度优先算法,搜索第 2 步中构建的搜索树.

2.1 计算有效任务集

2.1.1 寻找可达任务

受工人最晚工作时间限制和任务过期时间的约束,每个工人只能完成小部分任务.因此,首先应该在不违背约束条件的前提下找出每个工人可达的任务集合.工人 w 可达的任务集合,记作 RS_w,应该满足以下两个条件.

1) $\forall s\in RS_w, c(w.l,s.l)\leqslant s.e$;

2) $c(w.l,s.l,w.l)=c(w.l,s.l)+c(s.l,w.l)\leqslant w.t$.

其中,$c(w.l,s.l,w.l)$是工人 w 从 $w.l$ 出发经过 $s.l$ 返回 $w.l$ 的时间.以上两个条件可以保证一个工人在任务过期之前能够从他的起点到达任务所在的位置,并且同时留有足够的时间在工人最晚工作时间前返回他的起点.从计

算的角度来说,可达任务是一个以工人的起点为圆心、以最大行驶距离($w.t/2$ 乘以固定的速度)为半径的圆形.例如在图 1 中,对于工人 w_5,s_{10} 是可达任务,而 s_9 是不可达任务,因为到达 s_9 的时间大于 s_9 的过期时间.

2.1.2 寻找极大有效任务集

在搜索最优解的时候,为了加快搜索的效率,本文希望对某一个工人,可以一次性给其分配多个同时可达的任务,而不是每次分配一个任务,然后再判断新分配的任务和已经分配的任务是否同时可达,因此需要预计算出每个工人的极大有效任务集合.给定每个工人的可达任务集合,可以证明:找出每个工人的极大有效任务集(MVTS)是一个 NP 难的问题,证明过程与文献[5]类似.但是由于每个工人的可达任务集通常都不大,这就意味着实际上这个问题可以由高效的算法解决.而且计算每个工人的 MVTS 是完全相互独立的,因此可以并行计算.

接下来,本文将介绍求极大有效任务集的动态规划算法的状态转移方程.本算法通过逐步增加可达任务集合大小,不断地扩展工人的可达任务集合,并在每次迭代中找出该集合下所有的 MVTS.给定一个工人 w 和一个任务集合 $Q\subseteq RS_w$,本文定义 $opt(Q,s)$为在经过 Q 集合中的任务后,到达任务 $s.l$ 所在位置所能完成的最大任务数量.R 是 Q 集合中的任务调度序列.用 s_i 表示序列 R 中位于 s_j 前面那个任务,而 R'表示 $opt(Q-\{s_j\},s_i)$相应的任务序列.$opt(Q,s_j)$可由公式(3)计算得到:

$$opt(Q,s_j)=\begin{cases}1, & \text{if } |Q|=1\\ \max\limits_{s_i\in Q,s_i\neq s_j} opt(Q-\{s_j\},s_i)+\sigma_{ij}, & \text{otherwise}\end{cases} \tag{3}$$

其中,$\sigma_{ij}=\begin{cases}1, & \text{if } t(s_j.l)\leqslant s_j.e, t(s_j.l)+c(s_j.l,w.l)\leqslant w.t\\ 0, & \text{otherwise}\end{cases}$.

σ_{ij}=1 表示在把任务 s_j 添加到序列 R'的最后,任务 s_j 仍然可以被完成,而且工人可以在其预期的最晚时间前返回起点.

当 Q 只包含一个任务 s_i 时,该问题非常简单,$opt(s_i,s_i)$=1.当$|Q|>1$ 的时候,就需要搜索 Q 来检查有效任务集合的所有可能,并找到 s_i,使得 $opt(Q,s_j)$取得最大值.公式(3)计算 MVTS 集合 Q_w 的时间复杂度为 $O(n^3\cdot 2^n)$,而空间复杂度为 $O(n\cdot 2^n)$.

例如图 1 中,工人 w_7 的可达任务为{12,13,15,16},初始化时,opt({12},12)=1,opt({13},13)=1,opt({15},15)=1,opt({16},16)=1.当集合 Q 的大小从 1 增加到 2 时,本文算法会依次计算 opt 的取值.例如 opt({13,12},12)=1,因为如果按照任务序列(13,12),则工人无法在预期最晚时间之前返回起点,即任务 12 和 13 不能同时完成.而 opt({13,15},15)=2,因为按照任务序列(13,15)的顺序,工人到达任务的时候都小于该任务的过期时间,且完成两个任务之后可以在最晚工作时间之前返回.以此类推,当集合 Q 从 2 增加到 3 时:

$$opt(\{13,15,16\},16)=\max\begin{cases}opt(\{13,15\},15)+\sigma_{15,16}=3\\ opt(\{13,15\},13)+\sigma_{13,16}=2\end{cases}.$$

表 2 展示了所有极大有效任务集,以及每个工人最多能够完成的任务数量 max$R$.

Table 2 Maximal valid task set

表 2 极大有效任务集

工人	MVST	$\lvert\max R\rvert$
w_1	{1,2},{1,3}	2
w_2	{5},{1,3},{4,6},{6,8}	2
w_3	{7}	1
w_4	{9,10},{4,10},{8,9},{4,9},{4,6,8}	3
w_5	{4},{12},{10,11},{7,11}	2
w_6	{11},{13},{14}	1
w_7	{12},{13,15,16}	3

2.2 分割工人集合

寻找最优解主要的挑战在于需要搜索规模庞大的搜索空间,当枚举所有工人所有可能的有效的任务集合时,时间复杂度随着工人数量的增加呈指数增长.

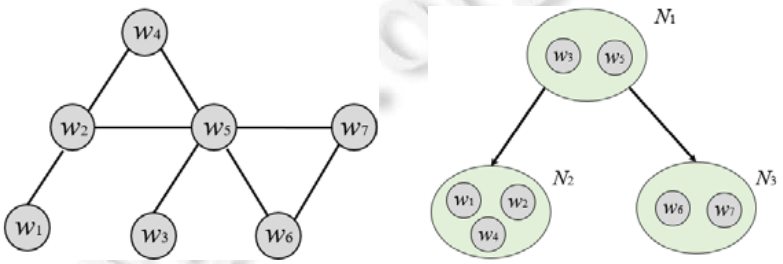
定义 7(任务依赖). 给定两个工人 w_i,w_j 以及各自的可达任务集 RS_{w_i},RS_{w_j},如果 $RS_{w_i}\cap RS_{w_j}=\varnothing$,则它们相互独立;否则,它们之间存在任务依赖.

例如图 1 中,工人 w_4 仅与 w_2 及 w_5 存在任务依赖,而与其他工人互不相关.显而易见:如果所有工人之间都不存在任务依赖,最优任务分配方案只需要把每个工人的最优方案简单相加.因此,为了减少寻找最优解的计算成本,就需要尽可能利用任务依赖,将工人分到互不相关的集合之中,在每个相互独立的集合中寻找该集合中的最优分配方案.

2.2.1 图分解

定义 8(工人依赖图). 给定工人集 W 和任务集 S,可以构造工人依赖图(WDG)$G(V,E)$,每个节点 $v\in V$ 表示一个工人 $w_v\in W$.如果两个工人 w_u 和 w_v 存在任务依赖,则 u 和 v 之间存在一条边 $e(u,v)\in E$.

如图 2(a)展示了示例中的工人依赖图.



(a) 工人依赖图 (b) 构造的搜索树

Fig.2 Worker dependency graph partition

图 2 分割工人依赖图

图分解的目的是将不相关的工人分割到不同集合中,同时保证所使用的分割集合(下文称割集)能够尽可能平衡地分割该连通图(分解算法见算法 1).做法是:在每一步分解中,将某个点以及与该点相连的点作为割集(第 7 行),尝试使用该割集切割连通图(第 8 行、第 9 行),同时记录该割集的大小(割集中包含多个工人)与图切分之后的最大连通子图中的工人数之和(第 10 行).在每一步中,选择能使得上述最小的割集(第 11 行~第 14 行)对原图进行切分,并对切分之后的连通子图递归调用该算法,直至连通子图中的工人数量小于阈值时停止(第 3 行~第 5 行).

算法 1. 图的树分解算法.

输入:工人依赖图 G,点割集(工人集合)的序列 C,点割集产生的顺序序列 I(初始化为 0,表示第 1 个点割集序号),全局变量 *index*,表示当前正在处理的点割集序号 *cur*;

输出:点割集序列 C 以及每个点割集产生的顺序序列 I.

(1) *TreeDecomposition*(G,C,I,*index*,*cur*);

(2) *Best_h=infinity*, *Best_S=empty* set; /**Best_h* 用来衡量割集切分连通图的能力,*Best_S* 为每一步中所选的最好割集*/

(3) if $|G|$<*threshold* then

(4) C[*cur*]←nodes in G; /*如果当前连通图中的工人数量小于阈值,则将当前连通图中的所有工人作为一个割集*/

(5) end if

(6) for each node $w_i\in G$ do

(7) $Set(V_i)$←node connected to w_i;

(8) $Set(V_i')$ ←node not connected to w_i;

(9) G'←build graph of $Set(V_i')$;

(10) h←|$Set(V_i)$|+max|$SubGraph(G')$|; /*绝对值为工人数量,h 为衡量割集分割能力的衡量标准*/
(11) if h<$Best\_h$ then
(12) $Best\_h$←h;
(13) $Best\_S$←$Set(V_i)$;
(14) end if
(15) end for
(16) $C[cur]$←$Best\_S$;
(17) G''←build graph for node not in $Best\_S$;
(18) for each sub graph $G_s \in G''$ do
(19) Sub_index=++$index$;
(20) $I[cur]$←Sub_index;
(21) $TreeDecomposition(G_s,C,index,Sub\_index)$;
(22) End for

因为最优图分解问题本身是 NP 难问题,所以本方案在树分解的每一次迭代中并没有求解最优的割集,而是依次尝试将依赖图中的每个节点与其周围节点作为割集,同时记录当前已经找到的最好割集(第 7 行~第 15 行).如图 2 中的情况,当使用 w_3 以及与 w_3 相连的 w_5 作为割集时,其余工人被分成相对平均的两组,产生的点割集序列 C 为[{w_3,w_5},{w_1,w_2,w_4},{w_6,w_7}],此时割集大小为 2,w_1,w_2 和 w_4 构成的最大连通子图大小为 3,两者之和为 5.可以发现:其余分割方式中,割集大小和最大连通子图大小之和不会比 5 小,因此,算法 1 会优先使用 w_3 和 w_5 做割集.

2.2.2 构造树

构造树的目的是将工人集合作为节点,组织成搜索树结构,从而使得搜索树中兄弟节点中的工人不存在任务依赖.

在图的树分解过程中,本文使用 I 记录了每个分割集合的序号以及每个子割集的序号,即记录了搜索树的结构信息.割集的序列 C 记录了每个割集中的工人信息,即每个节点所包含的具体工人.使用以上两类信息可以很容易地构造出树分解后的搜索树.例如:使用算法 1 所得的点割集序列 C[{w_3,w_5},{w_1,w_2,w_4},{w_6,w_7}],可以首先将第 1 个找到的割集{3,5}作为根节点,因为该割集能够产生更加平衡的搜索树,以该割集为根,以被该割集分割的连通子图作为子节点,如图 2 中的第 2 层的节点.如果子节点中工人的数量仍然大于阈值(本方案中阈值设置为 3,因为即使暴力穷举 3 个工人的所有可能的任务分配方案,也可以快速得到结果),则需要通过递归方法,依次构造下一级的搜索树.最后的树结构如图 2(b)所示.

通过这样的树结构,可以独立地解决每个兄弟结点的最优分配子问题,然后对子问题的结果进行合并(加法运算).而遍历所有工人所有可能情况的朴素算法需要对所有情况的结果进行乘法运算.例如图 2(b)中,编号为 N_1 的节点中,w_3 有一种分配方案(分配方案数量由表 2 中的极大有效任务集个数确定),w_5 有 4 种分配方案,则节点 N_1 中所有的可能情况为 4 种.同理,节点 N_2 中的可能情况为 40 种,节点 N_3 中的可能情况为 6 种.如果不考虑节点 N_2 和 N_3 中工人的相互独立性,则需要搜索 4×40×6=960 种情况,而如果像本方案一样考虑,则仅需要搜索(40+6)×4=184 种情况.因此,本方案可以极大地减少搜索最优任务分配方案的搜索次数.

2.3 搜 索

本节将展示基于树分解的搜索算法框架,从而解决全局最优任务分配问题.

一旦工人之间的依赖图转换成树结构,最优任务分配方案可以通过深度优先搜索算法求解.首先,本文给出整个搜索过程的框架,见算法 2.

给定工人集合 W 和任务集合 S,首先对每个工人 w_i 求其极大任务子序列 Q_{w_i} (第 2 行、第 3 行),S_i 为可达任务集合,然后建立相应的任务依赖图 G(第 4 行).对任务依赖图中的每个连通子图 $g \in G$,采用树分解算法将工人分到不同的工人集合中(第 6 行),然后,将工人集合的序列构建成搜索树结构(第 7 行).最后,对上一步建立的搜索

树,使用深度优先搜索算法寻找最优任务分配方案.初始化的启发函数值可以通过贪心算法计算.因为 G 的不同连通子图之间互不相关,因此最终任务分配方案只需将不同连通子图的方案累加即可(第 8 行).

算法 2. 搜索框架.

输入:工人集合 W,任务集合 S;

输出:最优任务分配方案 Opt.

(1) $Solve(W,S)$

(2) for each $w_i \in W$ do

(3) $Q_{w_i} = MVTS(w_i, S_i)$;

(4) $G \leftarrow$build WDG; /*建立工人依赖关系图*/

(5) for each connected component $g \in G$ do

(6) $X_g \leftarrow TreeDecomposition()$ of g; /*对连通子图 g 做树分解*/

(7) $N_g \leftarrow$Build search tree; /*将 X_g 使用搜索树结构进行索引*/

(8) $Opt \leftarrow Opt + DFSearch(N_g, S, W_{N_g}, LB(N_g))$;

(9) Return Opt;

下面详细阐述深度优先算法 3.搜索过程的 4 个关键参数如下:(1) 第 N 个子树的根节点;(2) 尚未分配的任务集合 S;(3) 第 N 个子树中尚未搜索过的工人集合 W_N;(4) 用于裁剪搜索空间的启发式函数值 h.启发函数值 h 代表了子树不被裁剪需要分配的最少任务数量.

该算法首先计算第 N 个子树中的所有工人能够完成的任务数量上界 $UB(N)$,然后比较该上界和启发函数值 h 的大小关系.易证:如果 $UB<h$,该子树即可被放心地剪枝.计算 $UB(N)$和 h 的方式会在第 2.3.1 节和第 2.3.2 节中解释.

算法中,搜索树的分支依赖于当前节点中的工人数量.如果节点存在尚未检测过的工人(第 7 行),本算法将会线性地检查这些未检测的工人(第 8 行).通过给工人 w 分配一个极大有效任务子序列中的所有任务(如果该序列中的任务已经被其他工人完成,则需要相应减去已完成任务),然后更新尚未完成的任务集合为 $S–Q$、尚未搜索过的工人集合为 $W_N–w$ 以及启发函数值为 $h–|Q|$,并递归地调用深度优先搜索函数搜索其他工人.启发函数值的更新规则如下:如果子递归返回的分配方案数量加上当前已分配的任务数量大于启发函数,则更新启发函数值并更新当前最优分配方案(第 9 行~第 12 行).如果当前搜索树节点中的工人已经搜索完成,本算法则对当前搜索节点的每一个子节点依次递归调用(第 15 行).因为每个子节点中的工人任务相互独立(由树分解算法决定),寻找最优分配方案的问题可以分解为同时对不同子树分别求解,然后对各自结果汇总求和即可得到全局的最优分配方案(第 16 行).

算法 3. 启发式搜索算法.

输入:当前节点序号 N,尚未分配的任务集合 S,第 N 个子树中尚未搜索过的工人集合 W_N,用于裁剪搜索空间的启发式函数值 h;

输出:最优任务分配方案 Opt.

(1) $DFSearch(N,S,W_N,h)$

(2) $Opt \leftarrow 0$;

(3) $UB(N) \leftarrow$Calculate upper bound of sub tree rooted at N; /*计算以节点 N 为根节点的子树可分配任务数量上界*/

(4) if $UB(N)<h$ then

(5) return 0;

(6) end if

(7) if $W_N \neq \varnothing$ then

(8) for each worker $w_i \in W_N$ do

(9) for each *MVT* set $Q \in MVTS(w_i,S)$ do

(10) $Opt \leftarrow \max\{DFSearch(N,S-Q,W_N-w,h-|Q|)+|Q|,Opt\}$

(11) $h \leftarrow Opt$;

(12) end for

(13) end for

(14) else

(15) for each child node N_i of N do

(16) $Opt+=DFSearch(N_i,S,W_N,h)$;

(17) end for

(18) end if

(19) return *Opt*;

例如,对图 1 中的情况,假设该搜索算法已经遍历过任务分配序列:

$$Q_{w_1}=\{1,3\},Q_{w_2}=\{5\},Q_{w_3}=\{7\},Q_{w_4}=\{4,6,8\},Q_{w_5}=\{10,11\},Q_{w_6}=\{14\},Q_{w_7}=\{13,15,16\}.$$

此时 h=13,则当算法尝试使用另一种任务分配方案 $Q_{w_1}=\{1,2\},Q_{w_2}=\{6,8\},Q_{w_3}=\{7\},Q_{w_4}=\{9,10\},Q_{w_5}=\{4\}$ 时,当算法 3 运行到第 3 行时,$UB(N_3)=|\max R_6|+|\max R_7|=4$,此时:

$$h'=13-Opt(N_2)=13-(|Q_{w_1}|+|Q_{w_2}|+|Q_{w_4}|)=13-6=7.$$

即,N_3 至少需要完成 7 个任务才有可能成为最优解,因为此时 $UB(N_3)<h'$,所以该种分配方案可以被放心裁剪.

算法 3 会递归调用自己,搜索所有的任务分配方案,将不会成为最优解的分配方案排除掉.因此,当算法退出时,该算法可以得到使得全局任务分配数量最多的分配方案.

2.3.1 估算上界

节点 N 可完成任务数量的上界记作 $UB(N)$,代表以节点 N 为根的子树中的所有工人,最多可完成的任务数量.基本的估算上界的方法是将以节点 N 为根的子树中的所有工人的极大有效任务集中,包含工人数量最多的极大有效任务集合进行累加,计算公式如下:

$$UB(N)=\sum_{i=1}^{|W|}(|\max R_{w_i}|) \tag{4}$$

公式中的 W 代表当前子树的所有工人; $\max R_{w_i}$ 表示工人 w_i 的极大有效任务集中,集合元素个数最大的集合.例如,当搜索算法开始搜索图 3 中的 3 号节点,$UB(N_3)=|\max R_6|+|\max R_7|=1+3=4$,$|\max R|$的值可以通过查找表 2 获得.

因为对所有的任务分配方案 A(包括最优任务分配方案),每个工人所能完成的任务数量不会超过$|\max R|$,因此以下不等式成立:

$$|A.S|=|\bigcup_{w\in W}S_w|\leqslant\sum_{w\in W}|S_w|\leqslant\sum_{w\in W}|\max R_w|=UB(N).$$

2.3.2 启发函数

为尽早对没有希望成为最优解的方案进行剪枝,本算法会计算启发式下界 h,且将其作为参数传递到递归函数中.h 表示以节点 N 为根的子树至少需要完成的任务数量.只有不小于 h,才有可能产生一个比当前已搜索的最优情况更有希望的解.易得:当某个子树的上界(最多可分配任务数量)小于该下界时,可以放心地将该种方案排除.

下面描述如何估算以节点 N 为根的子树的启发函数值.假设节点 N 包含 m 个子节点,如 N_1, N_2,…,N_m,深度优先搜索算法将会依次被用于搜索每个子树,目的是找到要给比当前已找到的最优解 *Opt* 更好的解.启发函数值通过以下公式更新:

$$h'=h-\sum_{j=1}^{i-1}Opt(N_j)-\sum_{j=i+1}^{m}UB(N_j) \tag{5}$$

公式(5)中,h 表示节点 N 的所有子节点需要分配的任务数量的最小值.$\sum_{j=1}^{i-1}Opt(N_j)$ 表示已遍历过的子树最多可分配任务数量之和,$\sum_{j=i+1}^{m}Opt(UB(N_j))$ 表示未搜索子树的估计上界之和.即:当前子树 N_i 至少需要完成的任务数量,等于启发函数值 h,减去当前子树之前已经搜索过的子树(N_1–N_{i-1})确定可以完成的任务数量,再减去当前子树之后尚未搜索过的子树(N_{i+1}–N_m)的预估上界 $UB(N)$.

2.3.3 优化策略

这里介绍 3 种优化策略,可以进一步降低搜索代价.

1) $UB(V)$优化:调用深度优先搜索算法之前,从搜索树的叶子节点开始,从下到上,预先通过贪心算法求出以每一个节点 N_i 为根的子树所能分配任务的上界(每个工人只取最大有效任务集合,不考虑任务被分配的情况),可以将启发函数限制得更小;
2) 单工人搜索优化:根据每个工人的极大有效任务集的基数($|MVTS|$)从大到小排序,先搜索基数大的集合.因为若较有可能产生最优解的较大有效任务集合搜索完成后,小的有效任务集合更容易剪枝;
3) 搜索子树顺序优化:对某个节点的所有子节点,按照子节点为根的子树中包含的工人数量从小到大排序.因为小的子树搜索较快,启发式函数值 h 可以快速更新,会变得更加紧密,对大的子树有更好的剪枝效果.

3 实验与结果

3.1 实验设置

因为空间众包领域缺乏基准实验数据,所以本文使用模拟数据集进行实验数据生成规则如下.

- 首先,在一个 100×100 的二维平面上使用均匀分布,随机产生 100 个点坐标,代表工人的位置;然后,变化每个工人的平均任务数量(T/W),在每一个工人周围随机产生代表任务的点,T/W 取值范围是[3,7] (经过实验验证:在更大的规模下实验时,搜索的深度通常会超过 20,搜索的时间会呈指数级增长,超过本实验机器配置允许的范围);
- 其次,给定任意工人及其周围的任务,任务的过期时间定义如下:从工人当前位置出发,采用贪心算法,依次选择距离工人当前位置最近的任务,对贪心算法计算出的任务序列,计算总的行驶时间 t,并将其作为任务过期时间的上界;然后定义一个范围$[e^l,e^u](0<e^l<e^u<1)$,任务的过期时间定义为$[e^l.t,e^u.t]$,本文使用 5 组任务过期范围[0.2,0.3],[0.3,0.4],[0.4,0.5],[0.5,0.6]以及[0.6,0.7];
- 最后,工人的预期最晚工作时间定义如下:将贪心算法计算的工人行驶时间 t 加上最后一个任务与工人起点的距离得 t_w,作为工人预期最晚工作时间的上界;然后定义一个范围$[d^s,d^t](0<d^s<d^t<1)$,工人的预期最晚工作时间定义为$[d^s.t_w,d^t.t_w]$,本文使用 5 组该范围[0.2,0.3],[0.3,0.4],[0.4,0.5],[0.5,0.6],[0.6,0.7].

基本上,任务过期时间范围和工人预期最晚工作时间范围决定工人可以完成的任务的百分比.

对每一个参数变化的结果,本文进行 50 组的实验.汇报的结果为 50 组实验的平均结果.所有实验均是在一台配置酷睿 i5-2400,CPU 3.1G HZ,8GB RAM 的机器上进行.

3.2 实验结果

3.2.1 树分解算法

本文评估了工人划分阶段的性能以及任务划分结果对搜索的影响,对比随机构造树算法 RTA(random tree-construction algorithm)和本文提出的平衡构造树算法 BTA (balanced tree-construction algorithm).RTA 算法随机选择一个工人集群作为搜索树的当前节点,BTA 算法每一次构造当前搜索节点时,总是选择较优的一个工人集群作为当前节点.本节从两个维度进行对比:(1) 搜索深度:使用深度优先遍历从根节点到叶节点搜索所枚举的工人最大数量;(2) 搜索时间:使用构造的搜索树搜索最优分配方案所花费的 CPU 时间.

图 3 展示了工人平均任务数量 T/W、任务过期时间系数 e^l 和工人最晚工作时间系数 d^s 对搜索深度的影响.

图 4 展示了以上参数对搜索时间的影响.如图 3(a)所示:两种构造树的算法中,搜索深度都随着 T/W 的增加而增加,但 BTA 算法可以产生更为平衡的树,这使得其效率比 RTA 更高,如图 4(a)所示.

如图 3(b)所示:当任务过期时间系数较小时,每个工人的可达任务数量也很少,BTA 算法和 RTA 算法在构造搜索树的能力上相差无几;但随着$[e^l,e^u]$的增加,工人可达任务数量也迅速增长,BTA 算法的优势越发明显.如图 4(b)所示:虽然搜索时间都随着任务过期时间系数的增加呈指数增长,但 BTA 算法的性能相比于 RTA 要高出一个数量级.

工人预期最晚工作时间对本方案的影响如图 3(c)所示.与任务过期时间类似,在最晚工作时间较小时,工人可达的任务数量有限,工人之间的任务重叠程度较小,即使存在重叠,重叠区域的可达任务数量也较少.因此 BTA 和 RTA 算法构造出的搜索树的搜索深度都是个位数.然而随着工人最晚工作时间系数的增长,工人可完成的任务数量也迅速增长,两种算法构造的搜索树深度线性增长.从图 4(c)可以看出:当工人最晚工作时间系数 $d\in[0.4,0.5]$时,RTA 算法构造的搜索树的搜索时间已经增长到不能容忍的程度.这是因为 RTA 算法构造的搜索树的搜索深度接近于 30.在这样大的搜索深度情况下,即使每个工人的极大有效任务集的数量为 3,3^{30} 的搜索规模也已经不可解了.这进一步印证了树结构对于搜索代价的重要性,也反映出本文提出 BTA 建树算法的高效性.

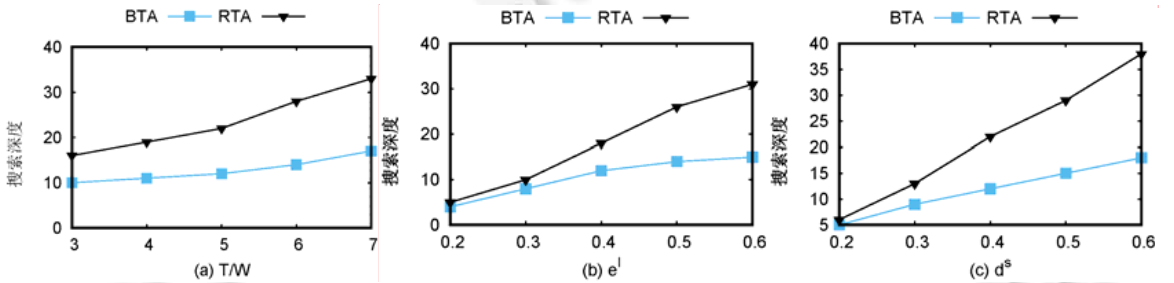


Fig.3 Effect of T/W, e^l and d^s on the depth of constructed search trees

图 3 参数 T/W,e^l,d^s 对构造出的搜索树深度的影响

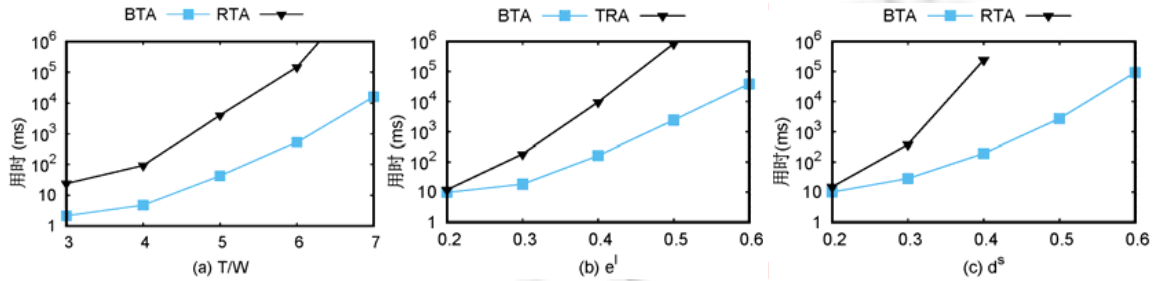


Fig.4 Effect of T/W, e^l and d^s on the Search Time

图 4 参数 T/W,e^l,d^s 对搜索时间的影响

3.2.2 搜索算法性能

本节在 BTA 算法构造的搜索树基础上,比较了 3 种不同搜索算法的性能:(1) 不带任何优化的分支限界算法(depth first search,简称 DFS);(2) 使用基于排序的搜索优化 DFS+W(depth first search+worker sort)(单工人的有效任务集从大到小顺序以及按照子树中工人数量从小到大的顺序);(3) 在(2)的基础上加入预计算搜索上界的算法 DFS+W&U(depth first search+worker sort and upper bound estimate).对于最终任务分配数量,本文比较了本文所提算法和两个基本方法的分配任务数量,这两个方法是贪心算法(greedy algorithm,简称 GA)和迭代贪心算法(iterative greedy algorithm,简称 IGA)[5].GA 依次为每个工人计算未分配任务集中可分配给该工人的最大任务数量,直到所有工人都已经分配完毕或待分配任务集合为空.IGA 迭代地进行任务分配和任务调度,直到

1 000 步以内无法出现更优解为止,最终选择最好的分配作为结果.

图 5 展示了不同搜索算法的效率.由图可知,无论是工人的平均任务数量、任务过期时间系数以及工人最晚工作时间系数较小时,每个工人的可达任务数量都很小,任务优化策略都不会收到显著效果.但是随着以上 3 个参数的增大,工人可达任务数量都会增大,问题变得越加复杂.

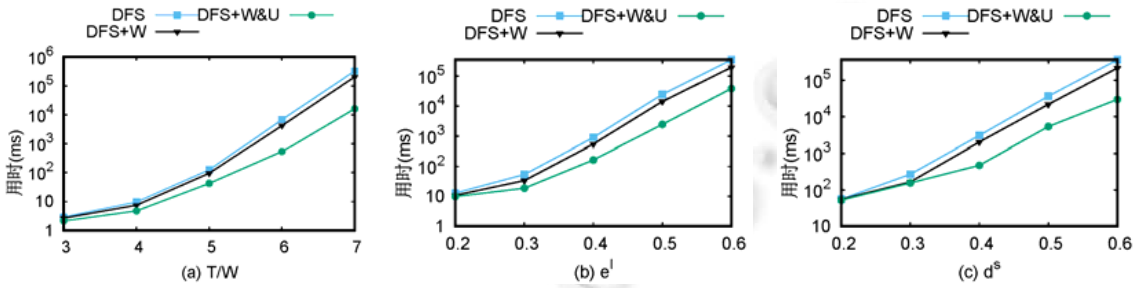


Fig.5 Effect of T/W, e^l and d^s on different search strategies

图 5 参数 T/W,e^l,d^s 对搜索策略的影响

基于排序的搜索算法 DFS+W 先搜索能够很快搜索完成的方案,快速地修正启发函数的值,以便对后面的需要耗费大量搜索时间的方案及早地进行剪枝.由于图 5 中的纵坐标成指数增大,所以得知 DFS+W 算法性能相比于 DFS 算法存在常量系数的提高.而 DFS+W&U 算法因为在预计算的过程中已经为搜索每一个节点提前计算了一个静态的上界,结合在每一步中动态计算的搜索上界,使得启发函数值被限制得更加紧密.从图 5 可以看出:DFS+W&U 实现更加高效的剪枝,且随着问题规模的变大,搜索的性能相比于 DFS 至少存在一个数量级的提高.

表 3 展示了随着工人平局任务数量增长,GA、IGA 和 OPT 搜索算法的搜索结果.OPT 算法是所假设情况下的最优情况,与 GA 相比,数据规模越大的情况下,OPT 可分配的任务数量和 GA 算法差别越大,由表 4 和表 5 可以看出:IGA 算法在 e^l<0.5 以及 d^s<0.5 的情况下,与 OPT 差别小于 20,和 GA 相比优势较为明显.但是在更为复杂的情况下,即使做到局部最优,与全局最优的 OPT 算法仍存在 10%左右的差距.

Table 3 Effect of T/W on theTotal Number of Assigned Tasks, e^l=d^s=0.6

表 3 T/W 对最终任务分配数量的影响,e^l=d^s=0.6

T/W	GA	IGA	OPT
3	97	106	121
4	125	136	167
5	176	189	223
6	222	243	284
7	255	282	327

Table 4 Effect of e^l on the Total Number of Assigned Tasks, T/W=5,d^s=1

表 4 e^l 对最终任务分配数量的影响,T/W=5,d^s=1

e^l	GA	IGA	OPT
0.2	113	120	124
0.3	145	154	168
0.4	174	194	216
0.5	220	243	279
0.6	263	285	328

Table 5 Effect of d^s on the Total Number of Assigned Tasks, T/W=5,e^l=1

表 5 d^s 对最终任务分配数量的影响,T/W=5,e^l=1

d^s	GA	IGA	OPT
0.2	140	143	148
0.3	165	172	183
0.4	218	226	242
0.5	259	273	295
0.6	298	317	342

4 相关工作

近年来,随着移动设备的迅速普及和移动数据资费的下降,空间众包市场蓬勃发展[1,2,7,22].空间众包模式中,任务发布者会将有明确位置要求的任务发布到某个平台上,平台上的用户(一般称为工人),需要实际到达指定的位置才当完成任务.空间众包模式已经在多个领域有着广泛应用,例如交通运输(Uber)、外卖配送(饿了么)、路况监控(OpenStreetMap)等.

4.1 任务发布模式

根据任务发布模式的不同,空间众包模式可以分为服务器端指派任务模式(SAT)和工人自选任务模式(WST)[2].在 SAT 模式中,服务器在收集了所有工人位置的前提下,将任务就近指派给周围的工人,目标是最大化任务分配的数量[2,3,5].2012 年,Kazemi 和 Shahabi 将空间众包问题归纳为工人和任务之间的匹配问题[2].

在工人自选任务模型中,服务器在线发布不同的空间任务,工人在获取满足自身限制条件的任务集合之后,可根据个人偏好选择完成其中的部分任务,而无需再次和服务器协商[3].Deng 等人在文献[3]中将 WST 模式中工人自选任务问题形式化为任务调度问题.2015 年,Ding 等人结合之前的工作[2,3],通过迭代进行任务分配和任务调度,逐步逼近最大化任务分配数量的最优解[5].这也是与本文研究内容最为相似的工作,不同点在于:本文从工人带有最晚工作时间的场景出发,并没有为工人统一设置接受任务的上界,且本文提出的是最优解方案.

4.2 数据质量

除了考虑任务发布模式,数据质量问题也备受关注[4,8,9].比如,Cheng 等人将工人的移动方向和任务的时间约束考虑在内,提出只有当工人到达任务的方向和到达任务的时间差足够大时,才会将任务分配给相应工人[4].文献[8]中,作者对工人完成任务的信誉进行打分,只有当工人的信誉值大于任务预先设定的阈值时,任务才是可以被接受的.Zhang 等人[9]分析了任务时空分布对完成质量的影响,并得到结论:在短时间内,在附近位置上的对同一任务高度重复的答案将会带来低质量;同时,在近期完成任务时的欺诈行为将会极大影响接下来的任务完成质量.

4.3 隐私保护

空间众包模式要求工人报告自身的位置和其他相关信息,其中包含一些敏感信息.保障这些信息不被泄露,可以有效地保护工人的隐私[10−16].比如,WST 模式下的隐私保护框架在文献[10]中被提出,参与者以某种概率收集信息,无需和服务器进行交互.Kazemi 等人[13]研究工人参与收集信息的运动中,以私密的方式将一系列任务分配给工人.文献[15]中,工人的位置收集并保存在可信的第三方,通过差分隐私的方式[11]向工人位置原始数据中插入噪音数据.当收到空间任务时候,SC 服务器需要在任务指定区域中找到足够多的工人来保证即使去除噪音数据的工人集合足以完成该任务.最后,在该区域内的真实工人会接收到任务通知,并决定是否接受该任务分配.空间屏蔽技术被用于混淆工人的实际位置,一种使用局部参与者精确位置优化和全局位置屏蔽的两阶段方案在文献[16]中被提出.

4.4 其他类型

Cheng 等人提出了基于预测的方式,通过预留工人来最优化多个时间片上的任务分配[17].近期,文献[18]研究了超局部的空间众包问题,在该问题中,只有在任务现场的工人才能成为该任务的候选工人.Tong 等人针对动

态场景下的微任务问题提出在线的分配算法[19].任务动态地出现,同时,工人可靠性未知情况下的任务分配问题在文献[20]中被关注.文献[21]中,将事件视为任务,将社交网络用户视为工人,根据时空关系将问题拆分成多个子问题,采用动态规划逐一求解,解决了任务稀疏场景下的分配问题.

5 总 结

空间众包中的任务具有特定的位置要求,只有工人实际行驶到指定位置才可能完成任务.本文研究了工人带有最晚工作时间约束的任务分配问题.本文创新性地提出了树分解方式,将不存在任务依赖的工人分割到相互独立工人集合中,并使用了尽最大努力的搜索树构建算法,构造尽可能平衡的搜索树.最后,本文设计了一种深度优先搜索算法,结合优化策略快速收紧上下界,能够有效地裁剪没有可能成为最优解的方案.实验结果表明:本文所提出的最优分配算法,在复杂的任务分配场景中更加具有优势.

接下来的研究方向是挖掘本算法中的可并行部分,使用并行算法以及分布式算法实现更加高效的分配方案,以便于将本算法推广到数据量更大、更加复杂的实际应用场景中.

References:

[1] Alt F, Shirazi AS, Schmidt A, Kramer U, Nawaz Z. Location-Based crowdsourcing: Extending crowdsourcing to the real world. In: Proc. of the Nordic Conf. on Human-Computer Interaction. Reykjavik: DBLP, 2010. 13–22. [doi: 10.1145/1868914.1868921]

[2] Kazemi L, Shahabi C. GeoCrowd: Enabling query answering with spatial crowdsourcing. In: Proc. of the 20th ACM SIGSPATIAL Int'l Cof. on Advances in Geographic Information Systems. 2012. 189–198. [doi: 10.1145/2424321.2424346]

[3] Deng DX, Shahabi C, Demiryurek U. Maximizing the number of worker's self-selected tasks in spatial crowdsourcing. In: Advances in Geographic Information Systems. 2013. 324–333. [doi: 10.1145/2525314.2525370]

[4] Cheng P, Lian X, Chen Z, Fu R, Chen L, Han JS, Zhao JZ. Reliable diversity-based spatial crowdsourcing by moving workers. Proc. of the VLDB Endowment, 2015,8(10):1022−1033. [doi: 10.14778/2794367.2794372]

[5] Deng DX, Shahabi C, Zhu LH. Task matching and scheduling for multiple workers in spatial crowdsourcing. In: Proc. of the 20th ACM SIGSPATIAL Int'l Cof. on Advances in Geographic Information Systems. 2015. 21. [doi: 10.1145/2820783.2820831]

[6] Khanafer A, Clautiaux F, Talbi EG. Tree-Decomposition based heuristics for the two-dimensional bin packing problem with conflicts. Computers & Operations Research, 2012,39(1):54–63. [doi: 10.1016/j.cor.2010.07.009]

[7] Bulut MF, Yilmaz YS, Demirbas M. Crowdsourcing location-based queries. In: Proc. of the Int'l Conf. on Pervasive Computing and Communications Workshops. 2011. 513–518. [doi: 10.1109/PERCOMW.2011.5766944]

[8] Kazemi L, Shahabi C, Chen L. GeoTruCrowd: Trustworthy query answering with spatial crowdsourcing. In: Proc. of the 20th ACM SIGSPATIAL Int'l Cof. on Advances in Geographic Information Systems. 2013. 314–323. [doi: 10.1145/2525314.2525346]

[9] Zhang G, Chen HP. Quality control for crowdsourcing with spatial and temporal distribution. In: Proc. of the Int'l Conf. on Internet and Distributed Computing Systems. Berlin, Heidelberg: Springer-Verlag, 2013. 169–182. [doi: 10.1007/978-3-642-41428-2_14]

[10] Cornelius C, Kapadia A, Kotz D, Peebles D, Shin M, Triandopoulos N. Anonysense: Privacy-Aware people-centric sensing. In: Proc. of the Int'l Conf. on Mobile Systems, Applications, and Services. 2008. 211–224. [doi: 10.1145/1378600.1378624]

[11] Dwork C. Differential privacy: A survey of results. In: Proc. of the Int'l Conf. on Theory and Applications of MODELS of Computation. Springer-Verlag, 2008. 1–19. [doi: 10.1007/978-3-540-79228-4_1]

[12] Gruteser M, Grunwald D. Anonymous usage of location-based services through spatial and temporal cloaking. In: Proc. of the Int'l Conf. on Mobile Systems, Applications, and Services. 2003. 31–42. [doi: 10.1145/1066116.1189037]

[13] Kazemi L, Shahabi C. Towards preserving privacy in participatory sensing. In: Proc. of the Int'l Conf. on Pervasive Computing and Communications Workshops. 2011. 328–331. [doi: 10.1109/PERCOMW.2011.5766897]

[14] Shen Y, Huang LS, Li L, Lu XR, Wang SW, Yang W. Towards preserving worker location privacy in spatial crowdsourcing. In: Proc. of the IEEE Global Communications Conf. 2015. 1–6. [doi: 10.1109/GLOCOM.2015.7416965]

[15] To H, Ghinita G, Shahabi C. A framework for protecting worker location privacy in spatial crowdsourcing. ACM Trans. on Very Large Databases Endowment, 2014,7(10):919–930. [doi: 10.14778/2732951.2732966]

[16] Pournajaf L, Li X, Sunderam V, Goryczka S. Spatial task assignment for crowd sensing with cloaked locations. In: Proc. of the Int'l Conf. on Mobile Data Management. 2014. 73−82. [doi: 10.1109/MDM.2014.15]

[17] Cheng P, Lian X, Chen L, Shahabi C. Prediction-Based task assignment on spatial crowdsourcing. In: Proc. of the Int'l Conf. on Data Engineering. 2017. 997−1008.

[18] To H, Fan L, Luan T, Shababi C. Real-Time task assignment in hyperlocal spatial crowdsourcing under budget constraints. In: Proc. of the Int'l Conf. on Pervasive Computing and Communications. 2016. 1−8. [doi: 10.1109/PERCOM.2016.7456507]

[19] Tong YX, She JY, Ding BL, Wang LB, Chen L. Online mobile micro-task allocation in spatial crowdsourcing. In: Proc. of the Int'l Conf. on Data Engineering. IEEE, 2016. 49−60. [doi: 10.1109/ICDE.2016.7498228]

[20] Ul Hassan U, Curry E. Efficient task assignment for spatial crowdsourcing. Expert Systems with Applications: An Int'l Journal, 2016,58(C):36−56. [doi: 10.1016/j.eswa.2016.03.022]

[21] She JY, Tong YX, Chen L. Utility-Aware event-participant planning. In: Proc. of the Int'l Conf. on Management of Data. 2015. 1629−1643.

[22] 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/1000-9825/5140. htm [doi: 10.13328/j.cnki.jos.005140]

附中文参考文献:

[22] 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/1000-9825/5140.htm [doi: 10.13328/j.cnki.jos.005140]

李洋(1990－),男,江苏徐州人,硕士,主要研究领域为数据挖掘,空间众包.

贾梦迪(1994－),女,硕士生,CCF 学生会员,主要研究领域为数据挖掘,机器学习.

杨文彦(1994－),男,硕士生,CCF 学生会员,主要研究领域为数据挖掘,机器学习.

赵艳(1988－),女,博士,CCF 专业会员,主要研究领域为数据挖掘,机器学习,轨迹分析,空间众包.

郑凯(1983－),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为轨迹挖掘,空间数据库,不确定数据库,空间众包,数据挖掘.