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.

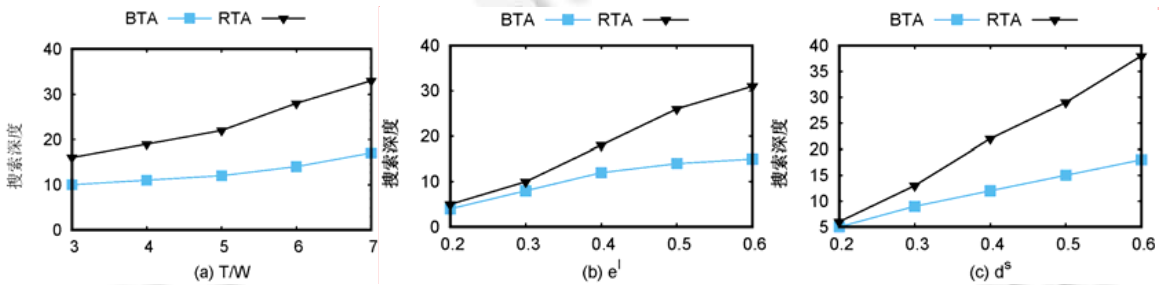

Fig.3 Effect of T/W, e^l and d^s on the depth of constructed search trees

图 3 参数 T/W,e^l,d^s 对构造出的搜索树深度的影响

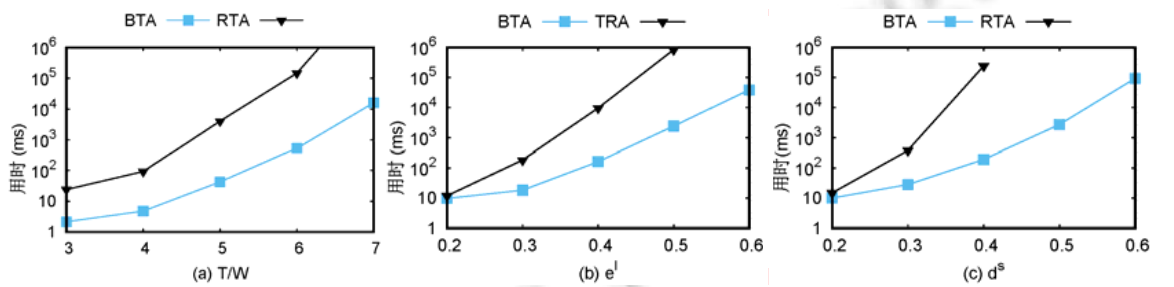

Fig.4 Effect of T/W, e^l and d^s on the Search Time

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.

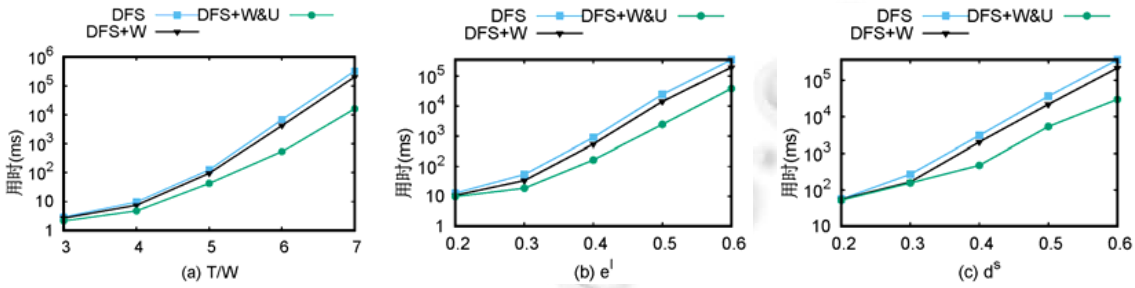

Fig.5 Effect of T/W, e^l and d^s on different search strategies

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.