计算机科学
COMPUTER SCIENCE

基于距离泛化的二分图(α,β)-core高效分解算法

张毅豪, 华征宇, 袁龙, 张帆, 王凯, 陈紫

引用本文

张毅豪, 华征宇, 袁龙, 张帆, 王凯, 陈紫. 基于距离泛化的二分图(α,β)-core高效分解算法[J]. 计算机科学, 2024, 51(11): 95-102.

ZHANG Yihao, HUA Zhengyu, YUAN Long, ZHANG Fan, WANG Kai, CHEN Zi. Distance-generalized Based (α,β)-core Decomposition on Bipartite Graphs[J]. Computer Science, 2024, 51(11): 95-102.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.231000130

基于距离泛化的二分图(α,β)-core高效分解算法

张毅豪[1]　华征宇[1]　袁　龙[1]　张　帆[2]　王　凯[3]　陈　紫[4]
1 南京理工大学计算机科学与工程学院　南京 210094
2 广州大学网络空间安全学院　广州 510555
3 上海交通大学安泰经济与管理学院　上海 200030
4 南京航空航天大学计算机科学与技术学院　南京 211106
(yhzhang@njust.edu.cn)

摘　要　(α,β)-core分解作为图数据管理与分析研究中的热点问题,已经被广泛应用于电商欺诈检测和兴趣群组推荐等实际场景中。然而现有(α,β)-core模型在构建时仅考虑顶点距离为1的邻居,难以刻画出二部图社区中的细粒度信息。针对此问题,提出了基于距离泛化的(α,β,h)-core模型,即由二部图中两个不相交的顶点集构成一个最大子图,满足一个集合中的任何一个顶点至少有α个与它的距离不大于h的邻居顶点,另一个集合中的任何一个顶点至少有β个与它的距离不大于h的邻居顶点。通过引入距离为h的邻居,解决了(α,β)-core模型细粒度刻画能力不足的问题。由于新模型需要考虑距离不大于h的邻居,因此(α,β,h)-core分解变得更为困难。为此,提出了基于计算共享的分解策略,据此设计了高效的(α,β,h)-core分解算法,并分析了算法性能。考虑到确定距离不大于h的邻居顶点非常耗时,还提出一种(α,β,h)-core下界以减少重复计算距离不大于h的邻居顶点,进一步提高计算效率。在8个真实图数据上的对比实验结果验证了新模型的有效性和算法的高效性。

关键词:二部图;(α,β,h)-core分解;高效算法
中图分类号　TP301

Distance-generalized Based (α,β)-core Decomposition on Bipartite Graphs

ZHANG Yihao[1],HUA Zhengyu[1],YUAN Long[1],ZHANG Fan[2],WANG Kai[3] and CHEN Zi[4]
1 School of Computer Science and Technology,Nanjing University of Science and Technology,Nanjing 210094,China
2 School of Cybersecurity,Guangzhou University,Guangzhou 510555,China
3 Antai College of Economics & Management,Shanghai Jiao Tong University,Shanghai 200030,China
4 College of Computer Science and Technology,Nanjing University of Aeronautics and Astronautics,Nanjing 211106,China

Abstract　(α,β)-core decomposition is a fundamental problem in graph analysis,and has been widely adopted for e-commerce fraud detection and interest group recommendation. Nevertheless,(α,β)-core model only considers the distance-1 neighborhood,which makes it unable to provide more fine-grained structure information. Motivated by this,(α,β,h)-core model is proposed in this paper,which requires the vertices in one/another part has at least α other vertices at distance not greater than h within the subgraph. Due to the distance-h neighborhoods being considered,the new model can identify more fine-grained structure information as verified in our experiments,which also makes the corresponding (α,β,h)-core decomposition challenging. To address this problem,an efficient algorithm based on computation-sharing strategy is proposed and time complexity is analyzed accordingly. As obtaining neighbors within distance h is time-consuming,a lower bound related to (α,β,h)-core is designed to avoid unnecessary distance-h neighbors computation to further improve the computational efficiency. Experimental results on eight real graphs demonstrate the effectiveness of the proposed model and efficiency of its algorithm.

Keywords　Bipartitegraphs,(α,β,h)-core decomposition,Efficient algorithm

到稿日期:2023-10-20　返修日期:2024-03-11
基金项目:国家重点研发计划(2022YFF0712100);信息系统工程重点实验室项目(WDZC20205250411);国家自然科学基金青年科学基金(NSFC61902184)
This work was supported by the National Key Research and Development Program of China(2022YFF0712100),Key Laboratory Program of Information Systems Engineering (WDZC20205250411) and Young Scientists Fund of the National Natural Science Foundation of China (NSFC61902184).
通信作者:袁龙(longyuan@njust.edu.cn)

1 引言

实际世界中的许多关系都可以被视为二部图，比如客户-产品网络[1]、用户页面网络[2]、基因共表达网络[3]、协作网络[4]等。随着对二部图应用的不断增加，人们开始专注于解决与二部图数据管理和分析相关的基本问题[5-6]。与此同时，(α,β)-core 在多个实际场景中也得到了广泛的应用[7]。

(α,β)-core 是二部图分析中一个基本的问题[8]。对于给定的二部图，(α,β)-core 被定义为满足以下条件的最大子图：一边顶点集合中所有顶点的度数大于或等于 α，而另一边顶点集合中所有顶点的度数大于或等于 β。(α,β)-core 分解能够计算出给定二部图中所有可能 α 和 β 组合的对应 (α,β)-core，类似于 k-core 分解[9-10]在单部图上的操作。现有文献中有大量关于 (α,β)-core 分解[11-12]以及 k-core 分解[13-15]的研究。

例如，对于在线群组推荐，高效计算 (α,β)-core 是实现容错子空间聚类的关键步骤，可提升协同过滤推荐系统的性能[16]；计算 (α,β)-core 还能在解决双分图问题的关键步骤中发挥作用，如双全子图计算和准双全子图计算[17]；此外，在用户页面网络中，(α,β)-core 模型能够用于欺诈者的检测。在社交网络中，例如 Facebook 和 Twitter，用户和页面形成一个用户页面二部图，其中边表示用户对页面的喜好关系。欺诈者常常使用大量虚假账户来夸大其对某些页面的喜爱度，以提升其知名度，(α,β)-core 模型则能够有效地识别此类欺诈行为[18-19]。

然而，正如本文实验结果（第 4.3 节）所指出，(α,β)-core 模型仅考虑了子图中距离为 1 的邻居顶点，导致该模型在准确捕捉二部图中的细粒度结构信息方面存在一定困难。另一方面，在社会计算、科学复杂网络分析等领域，研究者们广泛使用顶点距离大于 1 的邻居进行社区建模，并取得了丰硕的研究成果[20-21]。鉴于此，本文提出了基于距离泛化的 (α,β,h)-core 模型。不同于传统的 (α,β)-core，(α,β,h)-core 引入了新参数 h，要求子图中一边顶点集合中的任何一个顶点至少与 α 个顶点的距离不大于 h，另一边集合中的任何一个顶点至少与 β 个顶点的距离不大于 h。实验结果表明，(α,β,h)-core 模型能够有效地捕捉二部图中的细粒度结构信息。

由于需要计算距离不大于 h 的邻居，(α,β,h)-core 分解的计算量会增加，因此高效地进行 (α,β,h)-core 分解成为制约该模型的关键问题。为应对这一挑战，本文提出了一种基于共享计算的分解算法，该算法通过共享计算过程的中间值，巧妙地减少了非必要的计算，从而有效提高了分解的效率。同时，由于获取距离不大于 h 的邻居的操作非常耗时，因此本文还创新性地提出一种基于下界的分解方法，以避免重复计算距离不大于 h 的邻居，进一步提高了计算效率。

本文的主要贡献如下：

1）针对 (α,β)-core 模型对二部图中细粒度结构信息刻画能力弱的问题，提出了基于距离泛化的 (α,β,h)-core 新模型；

2）针对 (α,β,h)-core 新模型，提出了基于共享计算的高效分解算法，并设计了适用于 (α,β,h)-core 的下界以提高计算效率；

3）进行了大量的实验，验证了新模型的有效性和所提分解算法的高效性。

2 问题与定义

二部图 $G=(U,V,E)$ 是由两个不相交的顶点集 U 和 V 组成的图，其中 $E\subseteq U\times V$ 且每条边连接 U 中的一个顶点和 V 的一个顶点。使用 $U(G)$ 和 $V(G)$ 表示 G 的两个不相交的顶点集；用 $E(G)$ 表示 G 中边的集合；用 n_u 和 n_v 分别表示 $U(G)$ 和 $V(G)$ 中顶点的数量；用 n 表示图中顶点的总数；用 m 表示图中边的总数；某一顶点 $u\in U(G)\cup V(G)$ 的度数，用 $deg(u,G)$ 表示；同时使用 $d\max_U(G)$（$d\max_V(G)$）来表示在 $U(G)$（$V(G)$）的所有顶点中度数的最大值，即 $d\max_U(G)=\max\{deg(u,G)\mid u\in U(G)\}$。给定一个二部图 G，以及两个顶点集 $U'\subseteq U(G)$ 和 $V'\subseteq V(G)$，由 U' 和 V' 诱导出的二部图是 G 的子图 G'，满足 $U(G')=U'$，$V(G')=V'$，$S=U'\cup V'$，并且 $E(G')=E(G)\cap(U'\cup V')$。给定一个正整数 $h\in\mathbb{N}^+$，对于任意一个在 G' 中的顶点 $v\in S$，将它的 h-近邻定义为 $N_{G'}(v,h)=\{u\in S\mid u\neq v,d_{G'}(u,v)\leqslant h\}$，其中 $d_{G'}(u,v)$ 表示在上顶点 u 和 v 之间的最短距离；同时，将一个顶点的 h-度定义为该顶点 h-近邻中顶点的数量，即 $deg^h_{G'}(v)=|N_{G'}(v,h)|$。

定义 1（(α,β,h)-core） 给定一个二部图 G 和 3 个整数 α,β,h，用 $C_{\alpha,\beta,h}$ 来表示 (α,β,h)-core。$C_{\alpha,\beta,h}$ 由两个集合 $\mathcal{U}\subseteq U(G)$ 和 $\mathcal{V}\subseteq V(G)$ 组成，满足以下条件：由 $\mathcal{U}\cup\mathcal{V}$ 诱导的子图 G' 是 G 最大的子图，其中在 $\mathcal{U}$ 中所有顶点的 h-度不小于 α，在 $\mathcal{V}$ 中所有顶点的 h-度不小于 β，即 $\forall u\in\mathcal{U},deg^h_{G'}(u)\geqslant\alpha\wedge\forall v\in\mathcal{V},deg^h_{G'}(u)\geqslant\beta$。

定义 2（$\beta_{\max,\alpha}(u)$） 给定一个二部图和特定的 α,h，对于每一个顶点 $u\in U(G)\cup V(G)$，$\beta_{\max,\alpha}(u)$ 定义为使得 u 被包含在相应的 (α,β,h)-core 中的最大 β 值。如果不存在这样的 β 值，那么 $\beta_{\max,\alpha}(u)=0$。

定义 3（$\alpha_{\max,\beta}(u)$） 给定一个二部图和特定的 β,h，对于每一个顶点 $u\in U(G)\cup V(G)$，$\alpha_{\max,\beta}(u)$ 定义为使得 u 被包含在相应的 (α,β,h)-core 中的最大 α 值。如果不存在这样的 α 值，那么 $\alpha_{\max,\beta}(u)=0$。

问题定义：本文旨在研究 (α,β,h)-core 分解问题[1)]，即给定一个无向且未加权的二分图 $G=(U,V,E)$ 和一个正整数 $h\in\mathbb{N}^+$，计算出 U 中所有顶点 u 对于任意可能的 α 值的 $\beta_{\max,\alpha}(u)$，以及 V 中所有顶点 v 对于任意可能的 β 值的 $\alpha_{\max,\beta}(v)$。

3 算法

3.1 基础核分解算法

下面考虑对于任意可能的 α 值，如何计算出 U 中任意顶点 u 的 $\beta_{\max,\alpha}(u)$，$\alpha_{\max,\beta}(u)$ 可以类似地计算。对于特定的 α，$C_{\alpha,\beta+1,h}$ 包含在 $C_{\alpha,\beta,h}$ 之中；另外，考虑顶点 $u\in U(G)$ 和一个

1) 通过 (α,β,h)-core 分解结果，可以很容易构建类似文献[12]中的索引来实现最优时间计算给定 α,β,h 对应的 (α,β,h)-core，因此本文仅关注 (α,β,h)-core 分解。

特定的 α，若 $u\in C_{\alpha,\beta,h}\cdot\mathcal{U}$ 且 $u\notin C_{\alpha,\beta+1,h}\cdot\mathcal{U}$，则可得到 $\beta_{\max,\alpha}(u)=\beta$。因此对于特定的 α，可以通过逐步增加 β 来得到顶点 u 的 $\beta_{\max,\alpha}(u)$。据此，本文提出了基础核分解算法，如算法 1 所示。该算法使用了两个序列 BU 和 BV，它们分别按照 h-度递增的次序存储 U 和 V 中的顶点。以 BU 为例，$BU[i]$表示一个包含所有 h-度等于 i 的顶点的列表。这种在计算过程中保持顶点排序的技术被称为存储桶[22]，它可以在 O(1)的时间内更新每个单元。

该算法迭代所有可能的 α 值，然后计算在每个 α 值下任意顶点 $u\in U(G)$的 $\beta_{\max,\alpha}(u)$。计算过程可分为两个步骤：首先删去 U 中 h-度小于 α 的顶点，使得剩余顶点所在子图满足 $C_{\alpha,\beta,h}$，其中 β 表示 V 中顶点的最小 h-度；然后，对于上一步得到的$C_{\alpha,\beta,h}$，删除 V 中 h-度等于 β 的顶点，这一操作可能导致部分顶点的 h-度发生改变，一旦发现存在顶点 $u\in U(G)$的 h-度小于 α，按照上面的论述则可得到 $\beta_{\max,\alpha}(u)=\beta$。按照这种方式，可以通过自底向上的方式迭代所有可能的 α 来计算所有 $u\in U(G)$的 $\beta_{\max,\alpha}(u)$，具体算法操作如算法 1 所示。

算法 1 BasicDecom(G,h)

```
1. for each v∈V(G)do
2.    计算deg_G^h(v);
3.    BV[deg_G^h(v)]←BV[deg_G^h(v)]∪{v};
4. for each u∈U(G) do
5.    计算deg_G^h(u);
6.    BU[deg_G^h(u)]←BU[deg_G^h(u)]∪{u};
7. G* =G,α=-1,BU* =BU,BV* =BV;
8. while U[G* ]≠∅ do
9.    α=α+1;
10.   while BU* [α]≠∅ do
11.      从BU* [α]中选取顶点 u 并将 u 移除;
12.      计算N_G* (u,h);
13.      从G* 中移除 u 和它对应的边;
14.      for each w∈N_G* (u,h) do
15.         计算deg_G^h* (w);
16.         if w∈U(G* ) then
17.            move w to BU* [max(deg_G^h* (w),α)];
18.         else
19.            move w to BV* [deg_G^h* (w)];
20. compute β_max(G* ,α+1,h,BV* ).
```

首先初始化 BU 和 BV(第 1－6 行)，分别记录输入图中 U 和 V 中顶点的 h-近邻。然后从 $\alpha=0$ 开始遍历所有可能的 α(第 8－9 行)。对每个 α，删去 U 中 h-度小于或等于 α 的顶点，并且同步更新受影响顶点的 h-近邻，这对应第一个步骤(第 10－19 行)。具体来说，从列表中取出当前 h-度等于 α 的顶点 u，并且计算出顶点 u 的 h-近邻，接着删除顶点 u 及其连接的边(第 11－13 行)；对于顶点 u 的 h-近邻中的每一个顶点 w，计算出它当前的 h-度并根据其所属顶点集进行更新(第 14－19 行)。要注意的是，若 $w\in U(G^*)$，算法只能将 w 移动到$BU^*[\alpha]$之后的位置，即使 w 此时的 h-度小于 α，这是因为 $BU^*[\alpha]$之前的顶点已经被移除(第 17 行)。至此，U 中剩余顶点的 h-度至少为 $\alpha+1$。接下来计算 $\beta_{\max}$(第 20 行)。

计算 $\beta_{\max}$的算法如算法 2 所示。首先初始化两个集合 DU 和 DV，用于存储需要处理的顶点(第 2 行)。从 $\beta=1$ 开始逐步增加 β 值，对于 V 中所有 h-度等于 β 的顶点 v，计算该顶点的 h-近邻，并移除该顶点及其连接的边(第 3－8 行)。由于顶点 v 的 h-近邻中每个顶点的 h-度均发生了改变，因此算法将这些顶点按照其所属顶点集加入到 DU 或 DV 中(第 9－12 行)。接着对集合 DU 进行处理，取出一个顶点 du，若其 h-度小于 α，则说明该顶点 $du\in C_{\alpha,\beta,h}\cdot\mathcal{U}$ 且 $du\notin C_{\alpha,\beta+1,h}\cdot\mathcal{U}$，由此可知 $\beta_{\max,\alpha}(du)=\beta$，然后将受其影响的 h-近邻加入 DU 和 DV，并删除 du 及其连接的边(第 13－22 行)。再对集合 DV 进行处理，从 DV 中选取顶点 dv，计算 h-度并将 dv 及其连接的边移除。最后将顶点 dv 移动到 $BV[\max(deg^h_{G'}(dv),\beta)]$(第 23－26 行)，这里 dv 只能被移动到 $BV[\beta]$及之后的理由与算法 1 第 17 行相同。

算法 2 compute $\beta_{\max}(G,\alpha,h,BV)$

```
1. G′=G;
2. DU=∅,DV=∅,β=0;
3. while V(G′)≠∅ do
4.    β←β+1;
5.    while BV[β]≠∅ do
6.       从 BV[β]中选取顶点 v 并将 v 移除;
7.       计算N_G′(v,h);
8.       从 G′中移除 v 和它对应的边;
9.       for each w∈N_G′(v,h) do
10.         if w∈U(G′) then
11.            DU=DU∪{w};
12.      else DV=DV∪{w};
13.   while DU≠∅ do
14.      从 DU 中选取顶点 du 并将 du 移除;
15.      计算N_G′(du,h);
16.      if deg_G′^h(du)<α then
17.         for each dw∈N_G′(du,h) do
18.            if d w∈U(G′)then
19.               DU=DU∪{dw};
20.            else DV=DV∪{dw};
21.         β_max,α(du)=β;
22.         从 G′中移除 du 和它对应的边;
23. while DV≠∅ do
24.    从 DV 中选取顶点 dv 并将 dv 移除;
25.    计算deg_G′^h(dv);
26.    move d v to BV[max(deg_G′^h(dv),β)]。
```

定理 1 给定一个二部图 G，算法 1 的时间复杂度为 $O((\max_\alpha+\max_\beta)|U\cup V|D(D+E^{\sim}))$。其中，$\max_\alpha$和$\max_\beta$分别指算法 1 计算 U 中顶点的$\beta_{\max,\alpha}(du)$时所需考虑的 α 的最大值(第 9 行)，以及计算 V 中顶点$\alpha_{\max,\beta}(v)$时所需考虑的 β 的最大值；D 和$E^{\sim}$分别表示一个顶点的 h-近邻引起的子图中所包含的最多顶点数和边数。

证明：算法在$(\max_\alpha+\max_\beta)$次循环中遍历所有顶点，在对每个顶点进行处理时，需要重新计算其 h-近邻内所有顶点的 h-度。该操作需花费$(D+E^{\sim})$的时间，因此该时间复杂度的量级为 $O((\max_\alpha+\max_\beta)|U\cup V|D(D+E^{\sim}))$。

3.2 改进的核分解算法

算法 1 分别对 $U(G)$和 $V(G)$中的顶点进行独立处理，

需要进行 $O(\max_\alpha+\max_\beta)$ 次迭代才能完成核分解。然而，$(\max_\alpha+\max_\beta)$ 在实际图中可能非常大，这使得算法 1 在真实场景中难以被有效应用。本节在处理 $U(G)$ 和 $V(G)$ 顶点的期间，采用计算共享的方式，将迭代次数成功减少到 2δ，其中 δ 是满足 $C_{\delta,\delta,h}$ 不为空的最大值。同时，还提出了核分解的下界，并采用优先队列来存储顶点，改进了对 h-度的计算方法，最终提出了一种改进的核分解算法。

定理 2 给定一个二部图 G，对于图中所有顶点 $v\in V(G)$ 和任意给定 $\beta>\delta$，都有 $\alpha_{\max,\beta}(v)\leqslant\delta$。同样地，对于图中所有顶点 $u\in U(G)$ 和任意给定 $\alpha>\delta$，都有 $\beta_{\max,\alpha}(v)\leqslant\delta$。

证明：假设存在一些顶点 $v\in V(G)$ 并且 $\beta>\delta$，使得 $\alpha_{\max,\beta}(v)>\delta$。由 $\alpha_{\max,\beta}(v)$ 的定义可知，$C_{\delta+1,\delta+1}$ 必为非空，这与 δ 的定义相矛盾。因此对于所有的 $\beta>\delta$ 且 $v\in V(G)$，都有 $\alpha_{\max,\beta}(v)\leqslant\delta$。$\beta_{\max,\alpha}(v)\leqslant\delta$ 的证明同理可得。

由定理 2 可知，算法的迭代次数可减少到 2δ。具体来说，在计算 $\beta_{\max}$ 的过程中仅需将 α 由 1 迭代到 δ；同样地，在计算 $\alpha_{\max}$ 的过程中仅需将 β 由 1 迭代到 δ。这是由于在将 α 由 1 迭代到 δ 的过程中，不仅能计算出对应 α 的 $\beta_{\max}$，还能同时计算出 $\beta>\delta$ 时对应的 $\alpha_{\max}$。β 由 1 迭代到 δ 的过程同理。

下面讨论适用于核分解的下界。算法 1 在大型和密集的网络上是低效的，因为对于每个删除的顶点，它需要重新计算其 h-近邻中每个顶点的 h-度。对此，本文对算法 1 进行了改进，提出了两个下界。

定理 3 给定一个二部图 G 和 h，如果 $\beta'\leqslant\beta,\alpha'\leqslant\alpha$，则 $C_{\alpha,\beta,h}$ 一定包含在 $C_{\alpha',\beta',h'}$ 之中。

证明：该定理可以直接由定义 1 推导而得。

定理 4 对于一个二部图 G，给定 $\alpha_0,h,u_0\in U(G)$，其中 $\beta_{\max,\alpha_0}(u_0)=\beta_0$，则对于任意 α_1，满足 $\alpha_1>\alpha_0$，一定有 $\beta_{\max,\alpha_1}(u_0)=\beta_1\leqslant\beta_0$。故给定 h，对于任意 $u\in U(G)$ 和 α，若存在 $\beta_{\max,\alpha+1}(u)$，则 $\beta_{\max,\alpha}(u)$ 的下界为 $\beta_{\max,\alpha+1}(u)$。

证明：对于任意顶点 $u\in U(G)$ 和给定的 α_0 和 h，其中 $\beta_{\max,\alpha_0}(u)=\beta_0$，假设存在 α_1，满足 $\alpha_1>\alpha_0$，并且其对应的 $\beta_{\max,\alpha_1}(u)=\beta_1>\beta_0$，则说明 u 属于 (α_1,β_1,h)-core。根据定理 3，u 也属于 (α_0,β_1,h)-core，这与 $\beta_{\max,\alpha_0}(u)=\beta_0$ 冲突，故 $\beta_1\leqslant\beta_0$。

观察 1 对于一个二部图 G，给定 α 和 h，算法 2 需要迭代 β 并且删除 h-度小于等于 β 的顶点（算法 2 第 4－8 行）。对于某一顶点 $v\in V(G)$，当 $\alpha=\alpha_0$ 时，将顶点 v 被删除时的 β 值记为 β_0；而当 $\alpha=\alpha_0+1$ 时，将顶点 v 被删除时的 β 值记为 β_1。观察发现，必然满足 $\beta_1\leqslant\beta_0$。换而言之，当 $\alpha=\alpha_0$ 时，顶点 v 被删除时的 β 值的下界是在 $\alpha=\alpha_0+1$ 时该顶点被删除时的 β 值。

此外，对计算 h-度的方法也进行了改进。在基础算法中，当删除某个顶点以后，需对其 h-近邻中的所有顶点重新计算 h-度；而在优化后的算法 3 中，若被删除顶点的 h-近邻中的某个顶点与被删除顶点的距离为 h，那么只需对该顶点原本的 h-度减 1 即可完成更新，无需再重新计算。

改进的算法如算法 3 所示。与算法 1 不同，两个下界的使用要求 α 从大到小进行迭代，因此需要维护两个集合 DG 和 DBV，分别记录图 G^* 和 BV^* 的变化。此外，本文定义了两个数组 LB 和 $setLB$，LB 表示 V 中顶点被删除时 β 值的下界，$setLB$ 为 true 表示该顶点存在这样的下界。

算法 3 ABHCoreDecom(G,h)

1. $\delta\leftarrow$满足 $C_{\delta,\delta,h}$ 不为空的最大 δ 值
2. 同算法 1 第 1－6 行；
3. $G^*=G$，$BU^*=BU$，$BV^*=BV$；
4. $DG=\emptyset$，$DBV=\emptyset$；
5. for $\alpha=0$ to $\delta-1$ do
6. while $BU^*[\alpha]\neq\emptyset$ do
7. 从 $BU^*[\alpha]$ 中选取顶点 u 并将 u 移除；
8. $DG[\alpha]\leftarrow DG[\alpha]\cup$u 连接的边；
9. 同算法 1 第 12－19 行；
10. $DBV[\alpha]\leftarrow BV^*$ 在当前 α 中的变化；
11. $BV_{\delta-1}=BV^*$，$G_{\delta-1}=G^*$；
12. for each $v\in V(G)$ do
13. $LB(v)\leftarrow0$；
14. for $\alpha=\delta-1$ to 0 do
15. $BV_\alpha^*=BV_\alpha$；
16. for each $v\in V(G_\alpha)$ do
17. if $LB(v)\neq0$ then
18. move v to $BV_\alpha^*[LB(v)]$；
19. set $LB(v)\leftarrow$true；
20. else $setLB(v)\leftarrow$false；
21. compute $\beta_{\max}^+(G_\alpha,\alpha+1,h,BV_\alpha^*,LB,setLB)$；
22. $BV_{\alpha-1}\leftarrow$用 $DBV[\alpha]$ 和 BV_α 恢复 $BV_{\alpha-1}$；
23. $G_{\alpha-1}\leftarrow$用 $DG[\alpha]$ 和 G_α 恢复 $G_{\alpha-1}$。

算法 3 首先将 α 由 0 迭代至 δ-1，在这一过程中删除 U 中 h-度小于或等于 α 的顶点，并记录每个 α 下 G^* 和 BV^* 的变化（第 5－10 行）；然后再由大到小反向迭代 α，用 DG 和 DBV 恢复当前 α 下的 G^* 和 BV^*。检查 $V(G_\alpha)$ 中的每个顶点 v，如果该顶点存在被删除时 β 值的下界，则该顶点在 BV^* 中的位置移动到其下界处，表示 β 至少迭代到其下标时顶点 v 才会被删除（第 14－23 行）。

本文同样对 $\beta_{\max}$ 的计算进行了改进，如算法 4 所示。与算法 2 整体结构相同，算法 4 首先删除 V 中 h-度小于或等于 β 的顶点，然后依次更新受影响的顶点。但在实现细节上有所不同，具体如下。与算法 2 中每删除一个 $BV[\beta]$ 中的顶点就更新受其影响的顶点不同，算法 4 使用了双层 while 循环，这样的设计能够一次性处理 $BV[\beta]$ 中现有的所有顶点，再考虑受其影响的顶点。这种方式能够使得受影响的顶点更快达到最终状态，从而减少迭代次数。此外，算法 4 在处理 $BV[\beta]$ 中的顶点 v 时，对其是否存在被删除时的下界进行判断。如果存在且之前未访问过，则将其放入 BV 适当的位置中，并标记为已访问状态（第 7－9 行），这是因为此时 β 一定小于它的下界，所以该顶点不需要被移除，从而避免了对其 h-近邻的 h-度进行更新。若不存在这样的下界，则从 G' 中移除 v 和它连接的边。接着根据观察 1，将 $LB(v)$ 赋值为 β。然后根据定理 2 更新 $\alpha_{\max,i}(v)$，其中 $i=1,2,\cdots,\beta$，若 $\alpha_{\max,i}(v)<\alpha$，则将 $\alpha_{\max,i}(v)$ 赋值为 α（第 10－15 行），这是因为当顶点 v 被删除时，顶点 v 在 $C_{\alpha,\beta,h}$ 中，因此 $\alpha_{\max,i}(v)$ 至少是 α。接着考虑将顶点 v 的 h-近邻中受影响的顶点加入 DU 或 DV。与算法 2 不同的是，算法 4 加入了下界判断。如果满足以下条件，则无需进行更新：基于定理 4，若当前顶点 u 满足 $\beta<\beta_{\max,\alpha+1}(u)$（第 17 行），则该顶点

在这一轮的处理中不会出现 h-度小于 α 的情况，故无需重新计算其 h-度；基于观察 1，若顶点 v 存在被删除时的下界且未被访问，则该顶点不会在当前这一轮处理中被删除。然后算法对 DU 进行处理，由于 DU 使用的是优先队列，在队头的点是最有可能满足 h-度小于 α 的顶点，因此优先处理这样的顶点能够更新更多顶点的 h-度，使未达到 h-度小于 α 条件的顶点更快达到该条件，从而减少迭代次数，达到减少计算的目的。对于 DU 中的元素处理，只需在算法 2 的基础上同样添加第 17 行和第 19 行的下界判断(第 21－30 行)。DV 的处理过程与算法 2 一致(第 31 行)。

算法 4 compute $\beta_{\max}{}^{+}(G,\alpha,h,BV,LB,setLB)$

```
1. 同算法 2 第 1－2 行；
2. while V(G′)≠∅ do
3.     β←β+1；
4.     while BV[β]≠∅ do
5.         while BV[β]≠∅ do
6.             同算法 2 第 6－7 行；
7.             if setLB(v) then
8.                 move v to BV[max(deg^h_G′(v),β)]；
9.                 set LB(v)←false；
10.            else
11.                从 G′中移除 v 和它对应的边；
12.                LB(v)←β；
13.                for i=1 to β do
14.                    if α_max,i(v)<α then
15.                        α_max,i(v)←α；
16.                for each w∈N_G′(v,h) do
17.                    if w∈U(G′)且 β≥β_max,α+1(w) then
18.                        DU=DU∪{w}；
19.                    else if w∈V(G′)且¬setLB(W) then
20.                        DV=DV∪{w}；
21.    while DU≠∅ do
22.        同算法 2 第 14－15 行；
23.        if deg^h_G′(du)<α then
24.            for each dw∈N_G′(du,h) do
25.                if dw∈U(G′)且 β≥β_max,α+1(dw) then
26.                DU=DU∪{dw}；
27.            else if dw∈V(G′)且¬setLB(w) then
28.                DV=DV∪{dw}；
29.            β_max,α(du)←β；
30.            从 G′中移除 du 和它连接的边；
31.    同算法 2 第 23－26 行。
```

定理 5 给定一个二部图 G，算法 3 的时间复杂度为 $O(\delta|U\cup V|D(D+E^{\sim}))$。其中 δ 指计算 $\beta_{\max,\alpha}(u)$ 和 $\alpha_{\max,\beta}(v)$ 所需循环的次数，且 $2\delta\leqslant max_{\alpha}+\max_{\beta}$。

证明：与基础算法相同，该算法在每次循环中都需要遍历所有的顶点，因此在每次循环中具有相同的时间复杂度。然而，正如第 4 章中的结果所示，由于计算 h-度的次数减少，ABHCoreD-ecom 算法每次循环通常比 BasicDecom 算法快。

4 实验结果

本文在 Linux 操作系统下进行了所有实验，并使用了一台 Intel Xeon 2.60GHz CPU 和 256GB RAM 的设备进行测试。

4.1 实验设置

本文评估了算法在 8 个真实的公开数据集上的表现。这些数据集均为未加权的无向图，其中 ML(ml-100k)来自 grouplens[1)]；PR(dbpedia-producer)，CS(komarix-citeseer)，AM(actor-movie)和 PA(dblp-author)来自 konect[2)]；DM(rec-amz-Digital-Music)，PLG(rec-amz-Patio-Lawn-Garden)，AMT(rec-amz-Automotive)来自 networkrepository[3)]。表 1 列出了这些数据集的摘要信息，其中 $|U|$ 和 $|V|$ 表示两个顶点层的顶点数，$|E|$ 表示边数。实验对所有数据集进行了预处理，删除了重复的边和孤立的顶点，并呈现了处理后的结果。实验实现并比较了以下算法：BasicDecom，即第 3 章中提到的基础算法(算法 1)，以下简称为 Basic；ABHCoreDecom，即本文提出的基于距离泛化的(α,β)-core 分解算法(算法 3)，以下简称为 ABH。所有算法均使用 C＋＋语言进行实现。实验将每个测试的最大运行时间设置为 4 天。若一个测试在规定时间内未停止，则将相应的处理时间表示为 INF。需要注意的是，每个数据集都进行了 5 次重复实验，以减小误差并避免结果的随机性。

表 1 真实的网络数据集

Table 1 Real network dataset

数据集	$\|U\|$	$\|V\|$	$\|E\|$	平均度数	最大度数
ML	943	1 682	100 000	76.19	737
PR	48 833	138 839	207 268	2.21	512
CS	105 353	181 395	512 267	3.57	385
DM	478 235	266 414	836 006	2.25	1 953
PLG	714 791	105 984	993 490	2.42	3 180
AMT	851 418	320 112	1 373 768	2.35	2 738
AM	127 823	383 640	1 470 404	5.75	646
PA	1 953 085	5 624 219	12 282 059	3.24	1 386

4.2 评估算法效率

实验 1 效率测试。实验设置了 $h=1,2,3$，分别评估了 Basic 和 ABH 算法在 8 个数据集上的运行时间和 h-度计算时间，结果如表 2 所列。可以发现，在每个数据集上，ABH 算法在运行时间和 h-度计算时间上均优于 Basic 算法，这主要是由于 ABH 算法在运行过程中避免了大量不必要的 h-度计算，这与第 3 章中的分析一致。此外，也可以观察到运行时间并不是简单随着数据集规模的增大而赠加，它还与数据集的结构有关。具体来说，ABH 算法中的 δ 以及 Basic 算法中 α 和 β 的迭代次数都会对运行时间产生影响。当 $h=3$ 时，每个顶点的 h-度急剧增加，导致 h-度计算需要花费更多的时间，从而使得 ABH 算法的剪枝效果更加明显，但是这也导致部分数据集未在规定时间内停止。

1) https://grouplens.org/datasets/movielens/

2) http://konect.cc/

3) https://networkrepository.com/

表 2 在 8 个数据集上 2 种算法的运行时间

Table 2 Running time of 2 algorithms on 8 datasets

(s)

算法	运行时间						h-度计算时间 K					
	$h=1$		$h=2$		$h=3$		$h=1$		$h=2$		$h=3$	
	Basic	ABH	Basic	ABH	Basic	ABH	Basic	ABH	Basic	ABH	Basic	ABH
ML	23.2	2.0	22653.9	476.1	INF	10217.4	13.5	0.2	22247.0	133.2	INF	9853.1
PR	30.0	2.6	281.2	63.2	117287.2	484.5	7.9	0.8	165.6	12.3	112162.6	182.0
CS	71.6	7.4	310.6	89.0	19229.9	694.6	13.6	2.3	158.2	22.1	16421.3	235.9
DM	979.6	29.0	86143.6	2706.7	INF	153847.4	293.3	17.6	79428.5	617.5	INF	40118.6
PLG	1462.2	30.4	21441.3	4197.3	INF	53930.4	528.5	20.1	16712.4	806.7	INF	13700.1
AMT	1716.9	42.1	25765.7	4018.7	INF	83375.6	567.4	29.2	20169.6	953.9	INF	21031.3
AM	340.8	37.5	70658.8	1605.0	INF	INF	141.1	17.0	63058.4	685.6	INF	INF
PA	19588.7	2328.0	INF	26350.5	INF	INF	14770.1	2084.2	INF	20119.2	INF	INF

实验 2 可扩展性测试。实验评估了算法的可伸缩性。为此，实验将顶点随机抽样数量从 20%增加到 100%，并将得到顶点的诱导子图作为输入图，以测试算法的可伸缩性。实验结果如图 1 和图 2 所示。可以发现，随着顶点数量的持续增加，Basic 和 ABH 算法的运行时间稳定增加。ABH 算法在所有情况下都比 Basic 算法具有更好的性能，且 Basic 算法的运行时间增长幅度更大。例如，在 PR 上，当 $h=3$，顶点数从 20% 增加到 100%时，ABH 算法的运行时间从 2.91 s 增加到 484.57 s，而 Basic 的运行时间从 3.77 s 增加到 117287.39 s。当 $h=3$ 时，在任意数据集上，ABH 算法比 Basic 算法快约两个数量级。随着数据规模的增大，Baisc 算法和 ABH 算法均出现了未能在规定时间内停止的情况。其中，Basic 算法在 6 个数据集中出现了这种情况，并且该情况最早出现在考虑 20%的顶点时；而 ABH 算法仅在两个数据集中出现了这种情况，并且在考虑全部顶点时才出现该情况。以上结果表明了 ABH 算法在实践中具有良好的可扩展性。

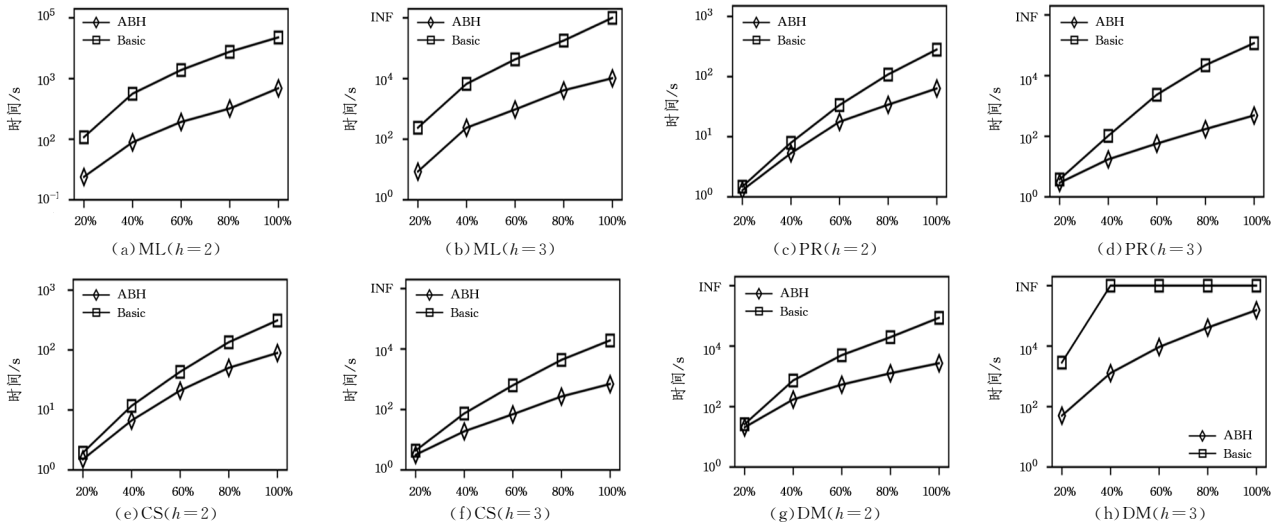

图 1 在前 4 个数据集上不同顶点比例对应的两种算法的运行时间

Fig. 1 Running time of 2 algorithms corresponding to different vertex proportions on the first 4 datasets

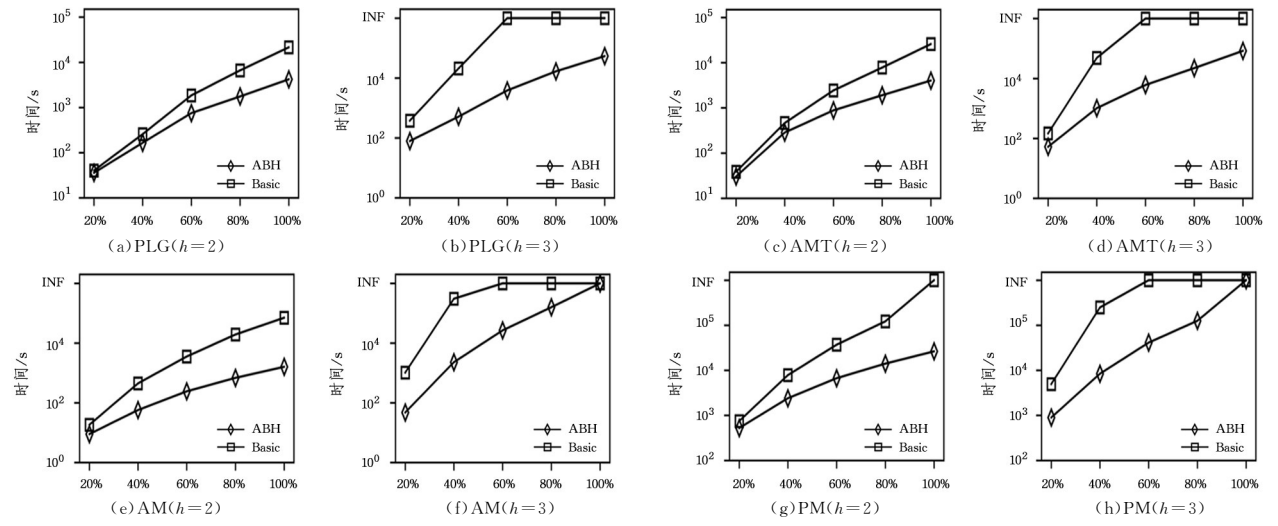

图 2 在后 4 个数据集上不同顶点比例对应的两种算法的运行时间

Fig. 2 Running time of 2 algorithms corresponding to different vertex proportions on the last 4 datasets

4.3 案例分析

实验在 dblp[1] 上节选出部分作者-文章二部图，如图 3 所示。其中，U 顶点集合表示作者（u_1 至 u_5 分别为 Yixiang Fang，Chenji Huang，Xuemin Lin，Long Yuan，Lu Qin），V 顶点集合表示文章。当某篇文章由图中作者创作时，作者顶点就会与该文章顶点建立一条连接边。图 3 展示了(α，β)-core 模型，对应文章中 $h=1$ 的情况。图 4 是(α，β，2)-core 分解的示意图。对于这个案例，现有的(α，β)-core 模型将整个图划分为(2，2)-core，无法进一步分解。但是，若使用(α，β，2)-core 分解，不仅能得到同样划分的(3，4，2)-core，还能获得更为细致的(2，5，2)-core 和(4，4，2)-core。通过观察可以发现，u_1，u_2 以及 u_3 和它们邻接的顶点之间形成了更加结构化的区域。当 $h=2$ 时，允许作者与作者之间建立直接联系，而不需要文章作为中介，这扩大了同一顶点集内的联系，进而得到更加细致的分析，这有效地刻画了二部图中细粒度结构信息。

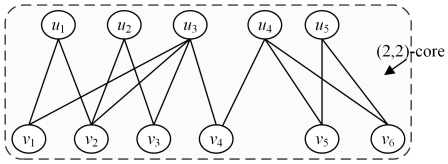

图 3 (α，β)-core 分解示意图

Fig. 3 Schematic diagram of (α，β)-core decomposition

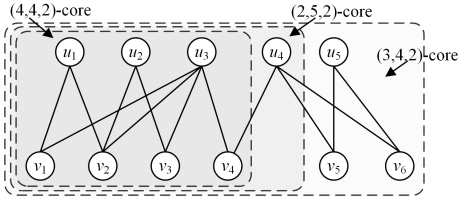

图 4 (α，β，2)-core 分解示意图

Fig. 4 Schematic diagram of (α，β，2)-core decomposition

结束语 本文研究了基于距离泛化的(α，β)-core 分解算法，提出了一种基于共享计算和下界的分解算法，该算法通过共享计算过程的中间值来减少非必要计算，提高分解效率。在此基础上，还提出一种(α，β，h)-core 下界以减少重复计算，进一步提高了计算效率。实验结果证明了该算法的有效性和高效性。然而当 $h=3$ 并且图较大时，仍然存在算法时间过长的问题，未来可以考虑使用并行方式来解决此问题。与此同时，动态变化已成为大规模二部图数据的重要特性，因此，在未来的工作中可考虑进一步研究针对动态二部图的高效(α，β，h)-core 分解算法。

参 考 文 献

[1] LISMONT J，RAM S，VANTHIENEN J，et al. Predicting interpurch-ase time in a retail environment using customer-product networks：An empirical study and evaluation[J]. Expert Systems with Applications，2018，104：22-32.

[2] KIM J，HASTAK M. Social network analysis：Characteristics of online social networks after a disaster[J]. International Journal of Information Management，2018，38(1)：86-96.

[3] YANG J，LI Y，LIU Q，et al. Brief introduction of medical database and data mining technology in big data era[J]. Journal of Evidence-Based Medicine，2020，13(1)：57-69.

[4] ZHOU F，YIN M M，JIAO C N，et al. Bipartite graph-based collaborative matrix factorization method for predicting miRNA-disease associations[J]. BMC Bioinformatics，2021，22(1)：1-16.

[5] XIA W，GAO Q，WANG Q，et al. Tensorized bipartite graph learning for multi-view clustering[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence，2022，45(4)：5187-5202.

[6] WANG K，LIN X，QIN L，et al. Accelerated butterfly counting with vertex priority on bipartite graphs[J]. The VLDB Journal，2023，32(2)：257-281.

[7] CHEN C，ZHU Q，WU Y，et al. Efficient critical relationships identification in bipartite networks[J]. World Wide Web，2022，25(2)：741-761.

[8] LIU B，YUAN L，LIN X，et al. Efficient(α，β)-core computation in bipartite graphs[J]. The VLDB Journal，2020，29(5)：1075-1099.

[9] BATAGELJ V，ZAVERSNIK M. An o(m) algorithm for cores decomposition of networks[J]. arXiv preprint cs/0310049，2003.

[10] KONG Y X，SHI G Y，WU R J，et al. k-core：Theories and applications[J]. Physics Reports，2019，832：1-32.

[11] WANG K，ZHANG W，LIN X，et al. Efficient and effective community search on large-scale bipartite graphs[C]// 2021 IEEE 37th International Conference on Data Engineering (ICDE). IEEE，2021：85-96.

[12] LIU B，YUAN L，LIN X，et al. Efficient(α，β)-core computation：An index-based approach[C]// The World Wide Web Conference. 2019：1130-1141.

[13] MALLIAROS F D，GIATSIDIS C，PAPADOPOULOS A N，et al. The core decomposition of networks：Theory，algorithms and applications[J]. The VLDB Journal，2020，29：61-92.

[14] CHAN T H H，SOZIO M，SUN B. Distributed approximate k-core decomposition and min-max edge orientation：Breaking the diameter barrier[J]. Journal of Parallel and Distributed Computing，2021，147：87-99.

[15] LIU J，XU C，YIN C，et al. K-core based temporal graph convolutional network for dynamic graphs[J]. IEEE Transactions on Knowledge and Data Engineering，2020，34(8)：3841-3853.

[16] DING D，LI H，HUANG Z，et al. Efficient fault-tolerant group recommendation using alpha-beta-core[C]// Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 2017：2047-2050.

[17] ZHANG Y，PHILLIPS C A，ROGERS G L，et al. On finding bicliques in bipartite graphs：a novel algorithm and its application to the integration of diverse biological data types[J]. BMC Bioinformatics，2014，15：1-18.

[18] ALLAHBAKHSH M，IGNJATOVIC A，BENATALLAH B，et al. Collusion detection in online rating systems[C]// Web Technologies and Applications：15th Asia-Pacific Web Confe-

[1] https://dblp.org/

rence, APWeb 2013, Sydney Proceedings 15. Springer Berlin Heidelberg, 2013: 196-207.
[19] BEUTEL A, XU W, GURUSWAMI V, et al. Copycatch: stopping group attacks by spotting lockstep behavior in social networks[C]// Proceedings of the 22nd International Conference on World Wide Web. 2013: 119-130.
[20] BONCHI F, KHAN A, SEVERINI L. Distance-generalized core decomposition[C]// Proceedings of the 2019 International Conference on Management of Data. 2019: 1006-1023.
[21] TATTI N. Fast computation of distance-generalized cores using sampling[J]. Knowledge and Information Systems, 2023, 65(6): 2429-2453.
[22] CORMEN T H, LEISERSON C E, RIVEST R L, et al. Introduction to algorithms[M]. MIT Press, 2022.

ZHANG Yihao, born in 1999, postgraduate. His main research interests include graph data management and analysis, and so on.

YUAN Long, born in 1988, Ph. D, professor, is a senior member of CCF(No. B0267S). His main research interests include graph data management and analysis, and so on.

(责任编辑:柯颖)

2024 CCF-深信服“远望”科研基金(第二期)启动申报

CCF-深信服“远望”科研基金重点面向国内高校、科研机构的研究人员和团队,旨在以小微课题的方式支持科研人员的研究与创新,推动科研成果转化,促进外部科研机构优秀研发能力与公司内部产品价值深度融合,构建互动合作与创新发展的生态圈,为深信服的产品与解决方案创新赋能。

基金申报对象:

1)申请人须是国内本科高校、科研院所在职的全职教师或研究人员;

2)申请人须具有高级专业技术职务(职称)或具有博士学位,拥有一定数量的相关领域研究成果,能作为项目的负责人并担负实质性研究工作;

3)申请人同一研究项目只能申报一个同类科研基金,不能重复申报。

4)申请人须是 CCF 服务计算专委委员。

资助方式:

2024 年计划资助不少于 20 项课题,项目实施期为 1 年,单项资助额度≤20 万元。

项目申请和评审:

1)符合条件的研究人员在项目申报规定时间内点击阅读原文填写《2024 年 CCF-深信服“远望”基金项目申报表》并发送至 yuanwang@sangfor.com.cn,每位申请人仅限提交一份申请。

2)申请人在申报前须确认:所在高校/科研院所可以作为项目依托单位签署科研合作协议,申请人本人可以作为项目负责人签署项目保密协议等相关承诺文件。任何针对项目申报的问题,请联系邮箱:yuanwang@sangfor.com.cn。

3)CCF 和深信服成立联合项目组,共同邀请专家审核申报项目。专家评审时主要考虑:

(1)申报项目的研究意义,包括国内外研究现状以及市场前景;

(2)申报项目的技术基础,包括技术成熟度、自主知识产权积累、与主流技术对标或产品适配验证等情况;

(3)申报项目的主要研究内容,包括技术路线、研究内容、具体技术指标、创新性、支撑条件需求等;

(4)项目实施计划、预算设计的合理性;

(5)预期交付的成果形式及数量;

(6)申请人能力及团队保障条件,包括领军人物、团队学术水平和科研能力。

4)联合项目组依据专家评审意见,结合深信服需求情况,确定资助的研究项目及资助强度等。

基金申报方式:

符合条件的研究人员在 2024 年 11 月 30 日之前填写《2024 年 CCF-深信服“远望”基金项目申报表》并发送至 yuanwang@sangfor.com.cn,每位申请人仅限提交一份申请。

时间规划:

项目指南发布	2024 年 10 月 21 日
项目申请截止	2024 年 11 月 30 日
项目评审结果发布	2024 年 12 月 30 日
签署协议,启动立项	2025 年 01 月 31 日
中期检查,提交报告	2025 年 06 月 30 日
项目结束,提交成果	2025 年 12 月 30 日
终期答辩	2025 年 12 月 31 日

据 CCF 微信公众号