Densest Multipartite Subgraph Search in Heterogeneous Information Networks

Lu Chen
luchen@swin.edu.au
Swinburne University of Technology

Chengfei Liu
cliu@swin.edu.au
Swinburne University of Technology

Rui Zhou
rzhou@swin.edu.au
Swinburne University of Technology

Kewen Liao
kewen.liao@acu.edu.au
Australian Catholic University

Jiajie Xu
xujj@suda.edu.cn
Soochow University

Jianxin Li
jianxin.li@deakin.edu.au
Deakin University

ABSTRACT

Cohesive multipartite subgraphs (CMS) in heterogeneous information networks (HINs) uncover closely connected vertex groups of multiple types, enhancing real applications like community search and anomaly detection. However, existing works for HINs pay less attention to searching CMS. In this paper, we leverage well-established concepts of meta-path and densest subgraph to propose a novel CMS model called the densest $\mathcal{P}$-partite subgraph. Given a multipartite subgraph of an HIN induced by i=$|\mathcal{P}|$ types of vertices defined in a query meta-path $\mathcal{P}$ (i.e., a $\mathcal{P}$-partite subgraph), we devise a novel density function which is the number of the instances of $\mathcal{P}$ over the geometric mean of the sizes of i different types of vertex sets in the subgraph. A $\mathcal{P}$-partite subgraph with the highest density serves as the optimum result. To find the densest $\mathcal{P}$-partite subgraph in an HIN with n vertices, we first design an exact algorithm with a runtime cost equivalent to solving $\Theta(|\mathbb{M}|)$ instances of the min-cut problem where $|\mathbb{M}|$=$O((\frac{n}{i})^i)$. Then, we attempt a more efficient approximation algorithm that achieves a ratio of $\frac{1}{i}$ but still incurs the cost of solving $\Theta(|\mathbb{M}|)$ instances of our proposed peeling problem. Both approaches struggle with scalability due to $\Theta(|\mathbb{M}|)$. To overcome this bottleneck, we improve the exact algorithm with novel pruning rules that non-trivially reduce the number of min-cut problem instances to solve to $O(|\mathbb{M}|)$. Empirically, 70-90% instances are pruned, making the improved exact algorithm significantly faster than the approximation algorithm. Extensive experiments on real datasets demonstrate the effectiveness of the proposed model and the efficiency of our algorithms.

1 INTRODUCTION

Network data emerged in real-life applications, including but not limited to e-commerce, cybersecurity, online video platforms, health, and social networks, are naturally heterogeneous since multiple types of vertices (objects) and edges (relationships) are involved. Cohesive subgraph search is fundamental and vital to network data analytics as it can naturally exhibit strongly correlated vertices. As a natural consequence, cohesive subgraph search over heterogeneous information networks (HINs) has drawn significant attention lately due to its practicality in revealing communities with multiple types of objects having different relationships. However, existing works pay less attention to cohesive multipartite subgraph-related problems, i.e., they focus more on discovering correlations of the same type of objects [7, 15, 34] or two types of objects [4, 19, 21, 32, 33]. There are few models supporting the search of cohesive multipartite subgraphs. The work of [14] requires many cohesive query parameters, and [13] can only deal with clique-based query patterns. In addition, both works concern search problems that are NP-hard. Despite the complication of HINs, users still prefer efficient searches with the least prior knowledge. Indeed, novel models for HINs with tractable searching problems and fewer query parameters, are desired.

Our problem. In this paper, we propose a novel cohesive multipartite subgraph model for HINs as follows that builds on the densest subgraph concepts [8, 16]. Then, we devise efficient algorithms for the corresponding search problem.

Effective density function. There are two considerations when designing density functions for multipartite subgraphs. Firstly, vertices of different types may not be directly connected, which raises questions about how to effectively bind these vertices together. Secondly, a multipartite subgraph comprises multiple types of vertices representing multiple types of objects, so the size distribution over different types of vertices should be taken into account to make subgraphs with different vertex distributions comparable. Therefore, an effective subgraph density function for HIN should address these concerns. We propose such a multipartite density function with the following numerator and denominator.

Numerator. To address the first concern, we adopt the widely applied concept of meta-paths, which are sequences of different vertex types and edge types between two given vertex types. With the concept of meta-paths, we allow users to specify multipartite subgraphs consisting of types of vertices and edges in the meta-path, denoted as the $\mathcal{P}$-partite subgraph. We then use the number of the meta-path instances as the numerator of the density function, which nicely binds multi-typed vertices via instances of meta-paths.

Denominator. To address the second concern, we incorporate ideas from the geometric mean. Specifically, for a $\mathcal{P}$-partite subgraph that includes i types of vertices $\mathcal{V}$=$\{V_1,...,V_i\}$, the denominator of our proposed density function is the ith root of the product of the sizes of the i types of vertices. Given that the denominator includes multiplication of the sizes of i vertex sets, it reflects the maximum possible number of meta-path instances in a $\mathcal{P}$-partite subgraph.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

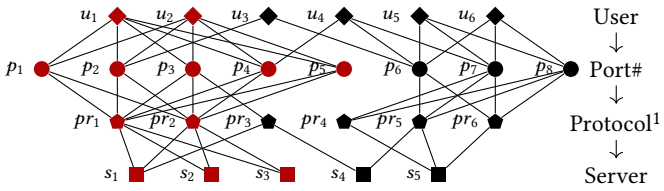


Figure 1: Networking Heterogenous Graph

Thus, together with the meta-path based numerator, our proposed density function is normalized, making $\mathcal{P}$-partite subgraphs with different vertex distributions comparable. Moreover, thanks to the ith root, maximizing such a density function encourages the overall size of the $\mathcal{P}$-partite subgraph to be large.

The density and densest $\mathcal{P}$-partite subgraph. We define the density of the $\mathcal{P}$-partite subgraph as the fraction of the numerator and denominator, denoted as ρ. Given all possible $\mathcal{P}$-partite subgraphs contained in an HIN, finding a $\mathcal{P}$-partite subgraph with the highest ρ serves as the research problem to be addressed in this paper, i.e., the densest $\mathcal{P}$-partite subgraph search problem.

Applications. We choose two typical applications of our search problem and introduce a specific application scenario.

Cybersecurity. In cybersecurity applications, the interactions over software, computers, users etc., can naturally be modelled as a heterogeneous network [10, 23]. $\mathcal{P}$-partite subgraph can discover intensively interacting software, computers and users, identifying and monitoring anomalous behaviours, including but not limited to spreading worms, scanning activities and various DDoS attacks.

E-commerce. Many popular online shopping applications deal with heterogeneous network data [12] that could comprise users, web pages, products and so on, where interactions can capture behaviours such as user-rate-product, webpage-list-product, and user-purchase-product. Densest $\mathcal{P}$-partite subgraphs can reveal users interested in similar webpages or products, which is useful for advertisements, recommendations, etc. Moreover, extensive interactions between users and products, such as abnormally high frequencies of rating and purchasing, can also be captured by densest $\mathcal{P}$-partite subgraphs, indicating possible fraudulent transactions/rates.

Generally speaking, for popular applications like the above two, the $\mathcal{P}$-partite subgraph search problem is superior compared to closely related models in twofold. Firstly, it possesses the capability to identify vertices from multiple specified types that are extensively connected through edges of multiple specified types. Secondly, it needs no cohesive parameters. Next, we introduce a specific application scenario for a more detailed discussion.

Application scenario. Figure 1 shows an HIN for a computer communication network [24] (which is a typical scenario for cybersecurity) consisting of four types of vertices, whose meta-schema is a path, i.e. user→port#→protocol→server. An interesting analysis of Figure 1 is user group discovery [24].

Existing approaches. Existing methods mentioned in [24] or (k,p)-core based approaches [7, 15, 34] share similar ideas. They first create virtual edges over vertices of the user type, which leads to a derived homogeneous graph. Then, cohesive subgraphs in the derived homogeneous graph serves as the discovered user groups, where the cohesiveness could be modularity, minimum degree, etc. As mentioned in [24], the effectiveness of such methods heavily relies on the quality of the derived virtual edges and the discovered group could be better if the quality of the virtual edges is high. The main reason, causing the difficulties of deriving a high quality virtual edge, is that a virtual edge summarises too much structural information that exists in HIN. A typical approach to link users is via instances of meta-paths. For instance, let the meta-path be user→port#→protocol→server→protocol→port#→user, and it is clear that u_1 is adjacent to both u_2 and u_5 in the derived homogeneous graph. However, according to the structure shown in Figure 1, u_1 is extensively connected to u_2 compared to the connection between u_1 and u_5.

Using densest $\mathcal{P}$-partite subgraph. By considering user→port# → protocol→server as a query meta-path $\mathcal{P}$, Figure 1 can be naturally treated as a $\mathcal{P}$-partite graph. Then we can discover a densest $\mathcal{P}$-partite subgraph in Figure 1 directly without the need to derive virtual edges over users. The densest $\mathcal{P}$-partite subgraph is the subgraph induced by the red-coloured vertices. u_1 and u_2 are within the same group. This makes great sense in this application scenario because u_1 and u_2 share similar networking behaviours compared to others, i.e., they communicate to similar servers via similar ports and protocols. For cybersecurity, such users can be treated as potential malicious collaborative attackers since the densest $\mathcal{P}$-partite subgraph reflects abnormally intensive and sophisticated interactions from users to servers via similar software and ports. What is more, the discovered densest $\mathcal{P}$-partite subgraph also reveals vulnerable ports, protocols, and servers.

Challenges. Although algorithms for densest subgraph search in simple graphs, directed graphs, and bipartite graphs have been studied extensively, none of them can be directly applied to searching a densest $\mathcal{P}$-partite subgraph. To the best of our knowledge, the two closest algorithms are [8] and [22], where [8] deals with a density function with a numerator that is closest to ours, while [22] focuses on a density function with a denominator being the square root of the multiplication of two vertex set sizes, which is a special case of our problem. The principal challenges for our densest $\mathcal{P}$-partite subgraph search problem are as follows. First, is it possible to solve the problem exactly in polynomial time? Second, can we approximately solve the problem more efficiently that also comes with an approximation guarantee? Third, are these solutions scalable so that reasonably large datasets can be handled?

Our approaches. We first demonstrate that the densest $\mathcal{P}$-partite subgraph problem can be potentially solved in polynomial time at the framework level. This is achieved by proving that the fractional programming technique [27] can be applied to our problem. Intuitively, to maximize a fractional function $\rho(\cdot)=\frac{f(\cdot)}{h(\cdot)}$, a transformed auxiliary optimization problem, i.e., maximizing $z(\cdot)=f(\cdot)-\rho h(\cdot)$, can be constructed. By solving a set of these auxiliary optimization problems with varying ρ, the optimum solution that maximizes $\rho(\cdot)=\frac{f(\cdot)}{h(\cdot)}$ can be obtained. We use the term 'potentially solved' as we assume that the auxiliary optimization problem related to our studied objective can be solved in polynomial time. However, this is challenging since the denominator $h(\cdot)$ contains the higher-order (ith) root of multiplication over the sizes of i vertex sets.

To cope with the issue, we further transform $h(\cdot)$ to make the ith root of multiplication over the sizes of i vertex sets a linear combination of the sizes of i vertex sets. Here, in the transformed

linear combination, each size of a vertex set is multiplied by a fixed parameter that we propose. We refer to these parameters as the i-root of the multiplication parameter set, denoted as iRM-set, $\mathcal{M}$. Although we later show that for a fixed $\mathcal{M}$ the corresponding auxiliary optimization problem can be exactly solved by our proposed flow network-based algorithm, the entire algorithm remains inefficient. This is because the densest $\mathcal{P}$-partite subgraph can only be derived after testing every possible $\mathcal{M}\in\mathbb{M}$ according to the different possible size distributions of i vertex sets in a $\mathcal{P}$-partite subgraph, where $|\mathbb{M}|$ is bounded by $O((\frac{n}{i})^i)$ in terms of the total number of vertices (n) in an HIN.

To improve efficiency, we first attempt approximation and propose a $\frac{1}{i}$-approximation peeling algorithm. The design and proof are nontrivial. First, the conventional vertex peeling process is more favorable for a density function whose denominator is in terms of the sum of vertice sizes. Second, the existing proof idea in [1] only works when i=2. To address the above concerns, we further utilize the iRM-set to propose a new peeling algorithm for our problem, which makes the peeling process aware of the multiplication of the sizes of i vertex sets. Then we prove that the subgraph with the highest density derived during the peeling process has at least $\frac{1}{i}$-approximation for the optimum result w.r.t. a given iRM-set. By trying every possible iRM-set, a subgraph with the highest density is secured to have $\frac{1}{i}$-approximation w.r.t. the global optimum result. Although expensive flow computations have been replaced by cheaper peelings, $\Theta(|\mathbb{M}|)$ instances of peelings greatly restrict the scalability of the approximation algorithm.

To propose scalable algorithms, we seek opportunities for pruning iRM-sets. We observe that by finding min-cuts using the exact algorithm, two types of density upper bounds can be derived for the use of pruning iRM-sets. Empirically, this can surprisingly prune up to 80% of iRM-sets, which makes the flow-based exact algorithm much faster than the approximation algorithm. Last but not least, we also propose a vertex pruning rule, which takes advantage of the early derived large density from our approximation algorithm. Along with the iRM-set pruning techniques, we propose a practically efficient exact algorithm that achieves the best runtime performance and solves $O(|\mathbb{M}|)$ instances of min-cut problems, which is much smaller than $\Theta(|\mathbb{M}|)$ in practice.

Contributions. Our main contributions are summarized below.

- We propose a novel densest $\mathcal{P}$-partite subgraph model. (**Section 2**)
- We develop an exact algorithm to solve the densest $\mathcal{P}$-partite subgraph problem in polynomial time. (**Sections 3 and 4**)
- We provide an approximate solution and prove an approximation ratio of $\frac{1}{i}$. (**Section 5**)
- We propose novel pruning rules, which significantly accelerates the exact algorithm. (**Section 6**)
- We conduct extensive experiments on real datasets to demonstrate the superior effectiveness and efficiency of our proposed methods. (**Section 7**)

Table 1: Notations

Notation	Definition
$\mathcal{A}$ and $\mathbb{A}$	a vertex types and a set of all vertex types
$\mathcal{P}$=(A_1,...,A_i)	a simplified meta-path consisting of i types of vertices
$\mathcal{V}$={V_1,...,V_i} and $\mathbb{V}$	A $\mathcal{P}$ family and a set of all possible $\mathcal{P}$ families
$G(\mathcal{V})$	a $\mathcal{P}$-partite subgraph induced by $V_1\cup...\cup V_i$
$\mathcal{F}(\mathcal{V})$	all instances of $\mathcal{P}$ contained in $G(\mathcal{V})$
$\mathcal{H}(\mathcal{V})$	$V_1\times...\times V_i$
$\mathcal{V}^*$	$\rho(V^*)$ is the optimum (maximum)
$\mathcal{M}$={m_1,...,m_i}, $\mathcal{M}'$, $\mathcal{M}''$	different iRM-sets
$\mathbb{M}$	The set contains all possible $\mathcal{M}$
$V^*_{\mathcal{M}}$	$\rho(V^*_{\mathcal{M}})$ is the maximum for any $\mathcal{V}$ conforming to $\mathcal{M}$
ρ^* and $\rho^*_{\mathcal{M}}$	abbreviations for $\rho(V^*)$ and $\rho(V^*_{\mathcal{M}})$
γ	a guessed or derived density value

2 PRELIMINARY & PROBLEM FORMULATION

2.1 Preliminary

We first introduce some well-known concepts.

Heterogenous information networks HINs. An HIN is defined as a directed graph G=(V,E) with a vertex type mapping function ϕ:$V\to\mathcal{A}$ and an edge type mapping function ψ:$E\to\mathcal{R}$, in which every $v\in V$ has a unique vertex type $\phi(v)\in\mathcal{A}$ and every $e\in E$ has a unique edge type $\psi(e)\in\mathcal{R}$.

Network schema. The network schema is a meta template for an HIN G=(V,E) with the vertex type mapping ϕ:$V\to\mathcal{A}$ and the edge type mapping function ψ:$E\to\mathcal{R}$, which is a directed graph defined over vertex types $\mathcal{A}$, with edges as relations from $\mathcal{R}$, denoted as T_G=($\mathcal{A}$,$\mathcal{R}$).

Figure 1 shows an HIN consisting of four types of vertices: A_1,A_2,A_3, and A_4 corresponding to User, Port#, Protocol, and Server respectively, and three types of edges: (A_1,A_2), (A_2,A_3), (A_3,A_4) corresponding to ($User$,$Port\#$), ($Port\#$,$Protocol$), and ($Protocol$,$Server$), respectively. Given a vertex u_i for $1\le i\le 6$, $\phi(u_i)$=A_1 corresponds to User-typed vertices and other types of vertices have similar mappings to Port#, Protocol, and Server. Given an edge (u_i,p_j) for $1\le i\le 6$, $1\le j\le 8$, $\psi((u_i,p_j))$ refers to the edge type of ($User$,$Port\#$). Similar edge mappings apply to other edges connecting: 1) Port# and Protocol-typed vertices, and 2) Protocol and Server-typed vertices.

Meta-path. A meta path $\mathcal{P}$ is a path defined on the network schema T_G, and is denoted in the form of $A_1\xrightarrow{R_1}A_2\xrightarrow{R_2}...\xrightarrow{R_{i-1}}A_i$, which defines a composite relation R=$R_1\circ R_2\circ...\circ R_{i-1}$ between type A_1 and A_i, where $\circ$ denotes the composition operator on relations. For simplicity, we denote $\mathcal{P}$ as an ordered set in forms of (A_1,A_2,...,A_i).

Instance of a meta-path. Given a permutation of vertices p=(v_1,v_2,...v_i), p is an instance of a meta path $\mathcal{P}$=(A_1,A_2,...,A_i), if p satisfies that for any two consecutive vertices v_j,v_{j+1} ($1\le j<i$) in p, $\psi((\phi(v_j),\phi(v_{j+1})))$=$R_j$ holds.

For instance, let the meta path $\mathcal{P}$ be ($User$,$Port\#$,$Protocol$) in Figure 1, paths such as (u_1,p_1,$pr1$), (u_1,p_5,pr_2), are instances of $\mathcal{P}$.

Now we discuss frequently used definitions in this paper.

Definition 1. *$\mathcal{P}$ **vertex set family ($\mathcal{P}$-family)**. Given $\mathcal{P}$ = (A_1, ..., A_i), we define a vertex set family $\mathcal{V}$={V_1,...,V_i} as a $\mathcal{P}$ vertex set family if for $1\le j\le i$, $V_j\subseteq V(A_j)$ and $V_j\neq\emptyset$. We also define the union of all possible $\mathcal{P}$ vertex set families of G as $\mathbb{V}$.*

Definition 2. *$\mathcal{P}$-**family $\mathcal{V}$ induced multipartite subgraph ($\mathcal{P}$-partite subgraph)**. Given a $\mathcal{P}$-family $\mathcal{V}$={V_1,...,V_i}, we define the subgraph induced by $V_1\cup...\cup V_i$ as the $\mathcal{P}$-family induced subgraph, denoted as $G(\mathcal{V})$, which is a simplification of $G(V_1\cup...\cup V_i)$.*

Definition 3. ***$\mathcal{P}$-family $\mathcal{V}$ induced $\mathcal{P}$ instances.*** *Given a $\mathcal{P}$-family $\mathcal{V}=\{V_1,...,V_i\}$, we define all $\mathcal{P}$ instances contained in $G(\mathcal{V})$ as the $\mathcal{P}$-family $\mathcal{V}$ induced $\mathcal{P}$ instances, denoted as $\mathcal{F}(\mathcal{V})$.*

For example, in Figure 1, $\mathcal{V}_{exa}=\{\{u_1,u_2\},\{p_1,p_2,p_3,p_4,p_5\},\{pr_1,pr_2\}\}$ is a $\mathcal{P}$-family given $\mathcal{P}=(User,Port\#,Protocol)$. Accordingly, these vertex induced subgraph is a $\mathcal{P}$-partite subgraph. In this $\mathcal{P}$-partite subgraph, $|\mathcal{F}(\mathcal{V}_{exa})|=20$.

Remark. Different from general definitions of HINs, we treat G and T_G as undirected graphs but treat a given meta-path $\mathcal{P}$ as directed. This allows more query meta-paths available, and therefore more multipartite subgraphs can be explored.

2.2 Density of $\mathcal{P}$-partite subgraph

In this section, we formally define the density of a subgraph w.r.t. provided meta-path $\mathcal{P}$ and introduce the research problem studied in this paper.

Definition 4. ***Density*** ρ. *Given a $\mathcal{P}$-family of i vertex sets $\mathcal{V}=\{V_1,...,V_i\}$, and let $\mathcal{H}(\mathcal{V})$ be $V_1\times...\times V_i$, the density of $\mathcal{V}$ induced multipartite subgraph is defined as follows.*

$$\rho(\mathcal{V}) = \frac{|\mathcal{F}(\mathcal{V})|}{|\mathcal{H}(\mathcal{V})|^{\frac{1}{i}}} \tag{1}$$

Problem 1. ***Densest $\mathcal{P}$-partite subgraph problem.*** *Given an HIN $G=(V,E)$ with schema $T_G=(\mathcal{A},\mathcal{R})$, a meta-path $\mathcal{P}=(A_1,...,A_i)$, find a $\mathcal{P}$-family $\mathcal{V}^*=\{V_1^*,...,V_i^*\}$ such that there exists no $\mathcal{P}$-family $\mathcal{V}'$ with $\rho(\mathcal{V}')>\rho(\mathcal{V}^*)$. Alternatively, this statement can be expressed as finding a $\mathcal{P}$-family V^* that solves the argument of the maxima below:*

$$\mathcal{V}^* = \arg\max\{\rho(\mathcal{V})|\mathcal{V}\in\mathbb{V}\}, \tag{2}$$

where $\mathbb{V}$ is the union of all possible $\mathcal{P}$-families.

As in the previous example, given $\mathcal{P}=(User,Port\#,Protocol)$, the $\mathcal{V}_{exa}$ induced $\mathcal{P}$-partite subgraph has a density of $\frac{20}{(2\times5\times2)^{\frac{1}{3}}}\approx7.368$, which is also a densest $\mathcal{P}$-partite subgraph. The runner-up is $V'=\{\{u_4,u_5,u_6\},\{p_6,p_p,p_8\},\{pr_4,pr_5,pr_6\}\}$ induced $\mathcal{P}$-partite subgraph, which has a density of $\frac{22}{(3\times3\times3)^{\frac{1}{3}}}\approx7.333$.

Technical scope. Given G and $\mathcal{P}$, we need some preprocessing to reduce the search space, count and enumerate instances of $\mathcal{P}$, which ensures the efficiency of solving Problem 1 but is not the main focus of Problem 1. To avoid sidetracking, we mainly focus on solving Problem 1 for *a connected multipartite subgraph induced by a $\mathcal{P}$-family of $\mathcal{V}$*.

3 FRACTIONAL PROGRAMMING BASED FRAMEWORK

We propose to use a fractional programming (FP) framework to solve Problem 1 exactly. Although this framework has been successfully adapted to densest subgraph problems with various density functions such as [30] and [3], it is still necessary to verify whether it is feasible to solve Problem 1, where the density (fractional) function has a denominator consisting of higher-order roots.

Unlike existing works that often mix up explanations of fractional programming-based densest subgraph search algorithms with flow network techniques, we aim to concisely explain and prove the correctness of using the fractional programming-based framework to solve Problem 1 without resorting to flow networks.

3.1 Warm-up: the initial transformation

Intuition. A fractional programming based method solves a fractional optimization problem by introducing an auxiliary optimization problem *with a linear formulation*, which can iteratively help verify or improve an intermediate solution for the fractional optimization problem until an optimum solution is derived.

We first introduce the auxiliary optimization problem of Problem 1 that eliminates the fraction.

Problem 2. ***Auxiliary optimization problem (AOP).*** *Sharing the same input as Problem 1, and $\gamma\in\{\rho(\mathcal{V})|\mathcal{V}\in\mathbb{V}\}$, AOP finds a $\mathcal{P}$-family $\mathcal{V}^{*\prime}$ that maximizes the equation below.*

$$\zeta(\mathcal{V},\gamma) = |\mathcal{F}(\mathcal{V})| - \gamma|\mathcal{H}(\mathcal{V})|^{\frac{1}{i}} \tag{3}$$

I.e., AOP solves the arguments of the maxima as follows.

$$\mathcal{V}^{*\prime} = \arg\max\{\zeta(\mathcal{V},\gamma)|\mathcal{V}\in\mathbb{V}\} \tag{4}$$

Intuitively, Equation 3 can help verifying a $\mathcal{P}$-family set that has the optimum density $\rho(\mathcal{V}^*)$ (simplified as ρ^*). Besides, given a guessed density γ, solving Problem 2 can indicate whether there exists a $\mathcal{P}$-partite subgraph having density greater, equal or less than the guessed density. Last but not least, solving Problem 2 can derive $\mathcal{P}$-families that converge to a $\mathcal{P}$-family with optimum density. The above correlations are formally defined and proven as follows.

Correlation 1. When $\gamma=\rho^*$ and $\rho(\mathcal{V}^{*\prime})=\rho(\mathcal{V}^*)$, $\zeta(\mathcal{V}^{*\prime},\rho^*)=0$ must hold. The correctness is clear given the definitions of the two problems and the density equivalence. I.e., $\rho(\mathcal{V}^{*\prime})=\rho(\mathcal{V}^*)$ means $\rho^*\cdot|\mathcal{H}(\mathcal{V}^{*\prime})|^{\frac{1}{i}}=|\mathcal{F}(\mathcal{V}^{*\prime})|$, and thus $\zeta(\mathcal{V}^{*\prime},\rho^*)=0$.

Correlation 2. Let $\gamma>\rho^*$,then $\zeta(\mathcal{V}^{*\prime},\gamma)<0$ must hold. The proof sketch is as follows. Suppose $\zeta(\mathcal{V}^{*\prime},\gamma)\geq0$, this means $|\mathcal{F}(\mathcal{V}^{*\prime})|\geq\gamma|\mathcal{H}(\mathcal{V}^{*\prime})|^{\frac{1}{i}}$, i.e., $\frac{|\mathcal{F}(\mathcal{V}^{*\prime})|}{|\mathcal{H}(\mathcal{V}^{*\prime})|^{\frac{1}{i}}}\geq\gamma$. This contradicts to the fact that $\mathcal{V}^*$ solves Problem1, indicating $\zeta(\mathcal{V}^{*\prime},\gamma)<0$ must hold.

Correlation 3. Let $\gamma<\rho^*$, then, $\zeta(\mathcal{V}^{*\prime},\gamma)\geq\zeta(\mathcal{V}^*,\gamma)>0$ must hold. The proof sketch is as follows. $\zeta(\mathcal{V}^{*\prime},\gamma)\geq\zeta(\mathcal{V}^*,\gamma)$ holds clearly since $\mathcal{V}^{*\prime}$ is an optimum results for Problem 2. On the other hand, $\frac{\gamma}{\rho^*}<1$, then $\zeta(\mathcal{V}^{*\prime},\gamma)>0$ holds, which means $\zeta(\mathcal{V}^{*\prime},\gamma)\geq\zeta(\mathcal{V}^*,\gamma)>0$.

Correlation 4. Let $\gamma_2=\frac{|\mathcal{F}(\mathcal{V}^{*\prime})|}{|\mathcal{H}(\mathcal{V}^{*\prime})|^{\frac{1}{i}}}$, where $\mathcal{V}^{*\prime}$ is $\arg\max_{\mathcal{V}}=\{\zeta(\mathcal{V},\gamma_1)|\mathcal{V}\in\mathbb{V}\}$, then $\gamma_2>\gamma_1$ must hold unless $\gamma_1=\rho^*$. It can be proved by contradiction. Supposing $\gamma_2<\gamma_1$, this means $\zeta(\mathcal{V}^{*\prime},\gamma_1)<0$, which contradicts to the fact that $\mathcal{V}^{*\prime}$ is $\arg\max_{\mathcal{V}}=\{\zeta(\mathcal{V},\gamma_1)|\mathcal{V}\in\mathbb{V}\}$. When $\gamma_2=\gamma_1$, then $\zeta(\mathcal{V}^{*\prime},\gamma_1)=0$ holds, and as shown in *Correlation 1*, this means $\gamma_2=\gamma_1=\rho^*$.

Using *Correlations* 1 to 4, two approaches [9, 31] could be used to solve **AOP** but without the prerequisite of flow network techniques.

Guess & verification based approach. We may guess a value γ via binary search, solve the corresponding Problem 2, and then adjust γ according to *Correlations* 2 and 3. As such, we can eventually approach to ρ^* and $\mathcal{V}^*$ and verify them via *Correlation* 1.

Iterative approach. By *Correlation* 4, given a connected $G(\mathcal{V})$, we can first set γ to be $\frac{|\mathcal{F}(\mathcal{V})|}{|\mathcal{H}(\mathcal{V})|^{\frac{1}{i}}}$ and then derive an improved γ' by using $\mathcal{V}^{*\prime}$ of $\arg\max_{\mathcal{V}}=\{\zeta(\mathcal{V},\gamma)|\mathcal{V}\in\mathbb{V}\}$. We can progressively approach to $\mathcal{V}^*$ by iteratively replacing the current γ with the improved γ' until the conditions in *Correlation* 1 are satisfied, leading to ρ^* and $\mathcal{V}^*$.

By the above analysis, it seems possible to solve Problem 1 as efficiently as the DS problem. However, this is a pitfall since, we have hidden the difficulties of solving Problem 2, i.e., $\mathcal{H}(\cdot)$ has i variables and they are in the form of multiplication. Novel

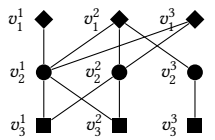


Figure 2: A toy example

techniques are needed to further transform Equation 3 so that the corresponding optimization problem can be solved efficiently.

3.2 The refined transformation

To the best of our knowledge, it is an open problem to solve Problem 2 due to the fact that $\mathcal{H}(\cdot)$ in $\zeta(\cdot)$ has to deal with i sets of vertices. We propose a novel formulation that transforms the multiplication of vertex sizes of i types in $\mathcal{H}(\cdot)^{\frac{1}{i}}$ to the sum of weighted vertex sizes for i types, i.e., from non-linear to linear. After this refined transformation, we will demonstrate that the refined Problem2 maintains similar effects as the discussed correlations and therefore can be used to solve Problem 1. In the next section, we will also show that the refined Problem 2 can be solved efficiently in polynomial time.

First, we introduce the refined ζ that incorporates a new set of parameters $\mathcal{M}$=$\{m_1,...,m_i\}$ in addition to $\mathcal{V}$=$\{V_1,...,V_i\}$ and γ,

$$\zeta(\mathcal{V},\gamma,\mathcal{M}) = |\mathcal{F}(\mathcal{V})| - \frac{\gamma}{i}(m_1|V_1| + \ldots + m_i|V_i|), \tag{5}$$

where $\mathcal{M}$ is defined as follows.

Definition 5. *i-**root of multiplication parameter set (iRM-set)**. Given the set of sizes for $\mathcal{V}$, denoted by X=$\{|V_1|,...,|V_i|\}$, the elements of an iRM-set is defined in the following format for all $1\le j\le i$:*

$$m_j = \frac{(|V_1|\cdot\ldots\cdot|V_i|)^{\frac{1}{i}}}{|V_j|}, \tag{6}$$

i.e., an iRM-set is $\mathcal{M}$=$\{m_1,...,m_i\}$.

Given two sets of sizes for two $\mathcal{P}$-families, it is possible that they lead to the same iRM-set. We formally define such a relationship as follows.

Definition 6. ***iRM-sets conformance**. Given X'=$\{|V_1'|,...,|V_i'|\}$ with $\mathcal{M}'$=$\{m_1',...,m_i'\}$, we define that X' and $\mathcal{M}'$ conform to an existing $\mathcal{M}$=$\{m_1,...,m_i\}$, if and only if for all $1\le j\le i$, m_j'=m_j. Similarly, given $\mathcal{V}$, we define that $\mathcal{V}$ conforms to $\mathcal{M}$ when $\{|V_1|,...,|V_i|\}$ conforms to $\mathcal{M}$.*

For instance, given $\mathcal{V}$=$\{\{v_1^1,v_1^3\},\{v_2^1,v_2^2\},\{v_3^1,v_3^2\}\}$, its X=$\{2,2,2\}$ and its $\mathcal{M}$ is $\{1,1,1\}$. Given $\mathcal{V}'$=$\{\{v_1^1,v_1^2,v_1^3\},\{v_2^1,v_2^2,v_2^3\},\{v_3^1,v_3^2,v_3^3\}\}$, its X' and $\mathcal{M}'$ are $\{3,3,3\}$ and $\{1,1,1\}$ respectively. Therefore, $\mathcal{M}'$ conform to $\mathcal{M}$ and we also say $\mathcal{V}'$ conform to $\mathcal{M}$.

Remark. In Equation 5, $\mathcal{M}$ and $\mathcal{V}$ are treated as two distinct sets of variables. The vertex sizes of i types of vertices in $\mathcal{V}$ may or may not conform to $\mathcal{M}$. Accordingly, we define the refined auxiliary optimization problem as follows.

Problem 3. ***Refined AOP (RAOP)**. Sharing the same input as Problem 2, and a fixed $\mathcal{M}$, RAOP finds a $\mathcal{P}$-family $\mathcal{V}^{*\prime}$ that maximizes Equation 5, i.e., RAOP solves arg max as follows.*

$$\mathcal{V}^{*\prime} = \arg\max\{\zeta(\mathcal{V},\gamma,\mathcal{M})|\mathcal{V}\in\mathbb{V}\}. \tag{7}$$

Next, we show that solving **RAOP** for a given $\mathcal{M}$ helps solving Problem 1. Since we have a new parameter set $\mathcal{M}$, we use $\mathcal{V}_{\mathcal{M}}^*$ to denote a $\mathcal{P}$-family that has corresponding i-type vertex sizes in accordance with $\mathcal{M}$ and has the highest density $\rho_{\mathcal{M}}^*$. $\mathcal{V}^{*\prime}$ is used to denote an optimal $\mathcal{P}$-family that is a solution of **RAOP** with $\mathcal{M}$ as a parameter, which has a density of $\rho(\mathcal{V}^{*\prime})$. Note that the following analysis will consider cases whether $\mathcal{V}^{*\prime}$ conforms to the given $\mathcal{M}$ or not. Therefore, we use $\mathcal{M}^{*\prime}$ to denote the parameter set that the sizes of $\mathcal{V}^{*\prime}$ conform to.

Lemma 1. *When γ=$\rho_{\mathcal{M}}^*$, $\rho(\mathcal{V}^{*\prime})$=$\rho(\mathcal{V}_{\mathcal{M}}^*)$, and $\mathcal{M}^{*\prime}$ conforms to $\mathcal{M}$, then $\zeta(\mathcal{V}^{*\prime},\gamma,\mathcal{M})$=0 must hold.*

Proof. First, Equation 5 can be organized as follows.

$$\zeta(\mathcal{V}^{*\prime},\gamma,\mathcal{M}) = \underbrace{|\mathcal{F}(\mathcal{V}^{*\prime})|}_{1} - \underbrace{\frac{\gamma}{i}(|V_1^{*\prime}|\cdot\ldots\cdot|V_i^{*\prime}|)^{\frac{1}{i}}(\frac{m_1}{m_1^{*\prime}}+\ldots+\frac{m_i}{m_i^{*\prime}})}_{2} \tag{8}$$

Since $\mathcal{M}^{*\prime}$ conforms $\mathcal{M}$, $\frac{m_1}{m_1^{*\prime}}+\ldots+\frac{m_i}{m_i^{*\prime}}=i$ holds. What is more, since $\gamma=\rho(\mathcal{V}^{*\prime})=\rho_{\mathcal{M}}^*$, $\zeta(\mathcal{V}^{*\prime},\gamma,\mathcal{M})=0$ hold.

Lemma 2. *When γ=$\rho_{\mathcal{M}}^*$, $\zeta(\mathcal{V}^{*\prime},\rho^*,\mathcal{M})$=0, but $\mathcal{M}^{*\prime}$ does not conform to $\mathcal{M}$, then $\rho(\mathcal{V}^{*\prime})>\rho_{\mathcal{M}}^*$ must hold.*

Proof. Since $\zeta(\mathcal{V}^{*\prime},\rho^*,\mathcal{M})$=0, the below equation holds.

$$\gamma = \rho_{\mathcal{M}}^* = \frac{i\rho(\mathcal{V}^{*\prime})}{\frac{m_1}{m_1^{*\prime}}+\ldots+\frac{m_i}{m_i^{*\prime}}}. \tag{9}$$

By the inequality of arithmetic and geometric means, the following inequality must hold:

$$\frac{m_1}{m_1^{*\prime}}+\ldots+\frac{m_i}{m_i^{*\prime}} \ge i(\frac{m_1}{m_1^{*\prime}}\cdot\ldots\cdot\frac{m_i}{m_i^{*\prime}})^{\frac{1}{i}}. \tag{10}$$

By the definition of iRM-set, the right part of the inequality, $(\frac{m_1}{m_1^{*\prime}}\cdot\ldots\cdot\frac{m_i}{m_i^{*\prime}})^{\frac{1}{i}}$=1 holds, which means $(\frac{m_1}{m_1^{*\prime}}+\ldots+\frac{m_i}{m_i^{*\prime}})\ge i$. What is more, $(\frac{m_1}{m_1^{*\prime}}+\ldots+\frac{m_i}{m_i^{*\prime}})=i$ if and only if $\frac{m_1}{m_1^{*\prime}}=\ldots=\frac{m_i}{m_i^{*\prime}}$. As such $\rho(\mathcal{V}^{*\prime})>\rho_{\mathcal{M}}^*$ must hold since $\frac{i}{\frac{m_1}{m_1^{*\prime}}+\ldots+\frac{m_i}{m_i^{*\prime}}}<1$.

Lemma 3. *Given fixed γ and $\mathcal{M}$ and when $\zeta(\mathcal{V}^{*\prime},\gamma,\mathcal{M})<0$, then there is no $\mathcal{V}'\subseteq\mathbb{V}$ with $\rho(\mathcal{V}')\ge\gamma$ when the sizes of $\mathcal{V}'$ conform to $\mathcal{M}$.*

Proof. We have the facts that $\mathcal{V}^{*\prime}$ is the optimum solution of **RAOP** and $\zeta(\mathcal{V}^{*\prime},\gamma,\mathcal{M})<0$. Suppose there exists $\mathcal{V}'\subseteq\mathbb{V}$ with $\rho(\mathcal{V}')\ge\gamma$, we have $\zeta(\mathcal{V}',\gamma,\mathcal{M})\ge0$. Then, $\mathcal{V}'$ will be the optimum solution of **RAOP**, which contradicts the facts above.

Lemma 4. *Given fixed γ and $\mathcal{M}$ and when $\zeta(\mathcal{V}^{*\prime},\gamma,\mathcal{M})>0$, then $\gamma<\rho_{\mathcal{M}}^*$ must hold and there exists $\mathcal{V}'$ with $\rho(\mathcal{V}')\ge\gamma$.*

Proof. Since $\zeta(\mathcal{V}^{*\prime},\gamma,\mathcal{M})>0$, we have the fact that $\rho(\mathcal{V}^{*\prime})>\frac{\gamma}{i}(\frac{m_1}{m_1^{*\prime}}+\ldots+\frac{m_i}{m_i^{*\prime}})$. If there is no such $\mathcal{V}'$, then $\forall\ \mathcal{V}'\in\mathbb{V}$, $\rho(\mathcal{V}')<\gamma$. This contradicts the fact above.

Lemma 5. *Let $\gamma_2=\frac{|\mathcal{F}(\mathcal{V}^{*\prime})|}{|\mathcal{H}(\mathcal{V}^{*\prime})|^{\frac{1}{i}}}$, where $\mathcal{V}^{*\prime}$ is $\arg\max_{\mathcal{V}}=\{\zeta(\mathcal{V},\gamma_1,\mathcal{M})|\mathcal{V}\in\mathbb{V}\}$, then $\gamma_2>\gamma_1$ must hold unless $\gamma_1=\rho_{\mathcal{M}}^*$ or $\gamma_1>\rho_{\mathcal{M}}^*$.*

Proof. We have the fact $\gamma_1<\rho_{\mathcal{M}}^*$. Besides, by Lemma 4, we have the fact $\zeta(\mathcal{V}^{*\prime},\gamma_1,\mathcal{M})>0$, where $\mathcal{V}^{*\prime}=\gamma_2$. Suppose $\gamma_2\le\gamma_1$, we have $\zeta(\gamma_2,\gamma_1,\mathcal{M})\le0$ by Lemmas 1 to 3. This contradicts the two facts that are discussed above.

Both the guess & verification and iterative approaches can be adapted, leading to the following theorem.

Theorem 1. *Given a fixed $\mathcal{M}$, we can find $\mathcal{V}^{*\prime}$ that has $\rho(V^{*\prime})$=$\rho(\mathcal{V}_{\mathcal{M}}^*)$, or $\rho(\mathcal{V}^{*\prime})>\rho(\mathcal{V}_{\mathcal{M}}^*)$.*

Proof. If using the guess & verification approach, we can attempt different γ and when γ=$\rho(\mathcal{V}_{\mathcal{M}}^*)$, the search terminates at either Lemma 1 or Lemma 2, which leads to $\mathcal{V}^{*\prime}$ with $\rho(\mathcal{V}^{*\prime})$=$\rho(\mathcal{V}_{\mathcal{M}}^*)$ or $\rho(\mathcal{V}^{*\prime})>\rho(\mathcal{V}_{\mathcal{M}}^*)$.

If using the iterative approach, we can start the search with $\mathcal{V}$, i.e., the entire input vertices. Then, by Lemma 5, it iteratively derives a larger density $\mathcal{P}$-family set until it terminates with one of Lemma 1, Lemma 2, or Lemma 3. As discussed, Lemma 1 or Lemma 2

lead to $\mathcal{V}^{*\prime}$ with $\rho(V^{*\prime})=\rho(\mathcal{V}^*_{\mathcal{M}})$ or $\rho(\mathcal{V}^{*\prime})>\rho(\mathcal{V}^*_{\mathcal{M}})$. For Lemma 3, we can derive $\mathcal{V}^{*\prime}$ with $\rho(\mathcal{V}^{*\prime})>\rho(\mathcal{V}^*_{\mathcal{M}})$.

From the analyses above, it is clear that by exploring every possible $\mathcal{M}$, Problem 1 can be precisely solved. We will provide a detailed algorithm and time complexity analysis after explaining how to solve **RAOP** in polynomial time in the next section.

Discussion. Using fractional programming is one way to solve Problem 1, which essentially involves solving a set of reformulated problems devoid of non-linear components such as fractions and multiplications. There might be other techniques with similar effects, for instance, convex programming. Our proposed $\mathcal{M}$ may also benefit such techniques. We would also like to highlight that the proposed **RAOP** is a generalized auxiliary optimization problem for several densest subgraph problems. When i=1, it can be used to solve the densest subgraph problem maximizing $\frac{|E|}{|V|}$. When i=2, it can be used to solve the densest bipartite subgraph problem maximizing $\frac{|E|}{\sqrt{|V_1||V_2|}}$, or adapted to solve the densest directed subgraph problem. When $i\geq 3$, it can be used to solve our proposed Problem 1.

We also would like to note that although **RAOP** assigns weights to vertex sets to get rid of non-linear components in our proposed density function, the solutions that we have discussed differ from the idea of reducing Problem 1 to a set of vertex-weighed densest subgraph problems [25]. This is because given a set of parameters $\mathcal{M}$, our solutions establish the optimum solution of **RAOP** to $\rho^*_{\mathcal{M}}$ as shown in Theorem 1 (i.e., $\rho(\mathcal{V}^{*\prime})\geq\rho^*_{\mathcal{M}}$), leading to pruning opportunities that will be introduced later. In contrast, the reduction-based approach establishes the optimum density of the vertex-weighted densest subgraph for a given $\mathcal{M}$ as a lower bound of $\rho^*_{\mathcal{M}}$ and when $\mathcal{M}$ is optimum, the two densities coincide.

4 FLOW-BASED EXACT ALGORITHM

In this section, we reduce Problem 3 to the max-flow/min-cut problem with our proposed flow networks. Then we formally show the complete exact algorithm for Problem 1 and analyze its time complexity. First, we will concisely review the flow problem so that our proposed methods can be more easily understood.

Flow network. A flow network is a directed graph $D=(V,E)$ with a set of nodes V and directed edge set E, having a unique source node s, a unique sink node t, and a non-negative capacity $c(u,v)$ for every directed edge (u,v). Note that we prefer using the term *node* when discussing a flow network and *vertex* in the context of an HIN. Following the flow network convention, we extend the capacity function to arbitrary node pairs by defining $c(u,v)=0$, if $(u,v)\notin E(D)$, which implies $f(u,v)=0$ if $(u,v)\notin E(D)$.

Flow and maximum flow. A flow f on N is real-valued function on node pairs satisfying three constraints. First, the capacity constraint, i.e., $f(u,v)\leq c(u,v)$ for every $(u,v)\in V(D)\times V(D)$. Second, the anti-symmetry constraint, i.e., $f(u,v)=-f(v,u)$ for every $(u,v)\in V(D)\times V(D)$. Third, the conservation constraint, i.e., $\sum_{u\in V(D)} f(u,v)=0$ for every $v\in V(D)\setminus\{s,t\}$. The value of the flow f is defined as $\sum_{v\in V(D)} f(v,t)$. A max-flow is a flow with the maximum flow value.

***s*-*t* cut and min-cut**. If S and T are two disjoint node subsets such that $S\cup T=V(D)$ and $S\cap T=\emptyset$, then the capacity or the cut value across the cut (S,T) is $c(S,T)=\sum_{v\in S,u\in T} c(u,v)$. (S,T) is an s-t cut if the source node $s\in S$ and the sink node $t\in T$. A min-cut is a s-t cut with the minimum cut value $c(S,T)$.

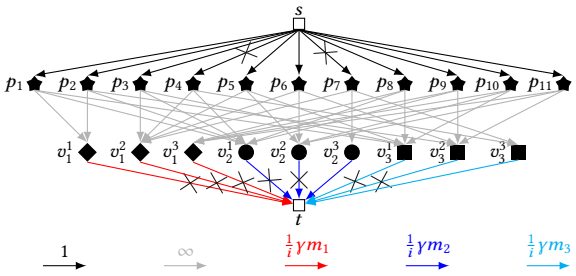


Figure 3: Constructed flow network and edge capacity

Max-flow min-cut theorem. The value of a min-cut is the same as the value of a max-flow.

4.1 Flow network and min-cut expression

Now, we formally define the structure of the flow network D.

Input. The flow network is built using $\mathcal{V}$, $\mathcal{P}$ instances in $G(\mathcal{V})$ denoted as P, along with the given iRM-set $\mathcal{M}$ and density γ.

Node creation. We create the s and t nodes first, followed by creating nodes for each vertex in $\mathcal{V}$, i.e., i sets of nodes $N_1,\ldots,N_i$ for the vertices in $V_1,\ldots,V_i$. Additionally, we create a node n_p for each instance of $\mathcal{P}$.

Directed edge creation. For each instance of $\mathcal{P}$, i.e., $p\in P$, we connect its corresponding node n_p to the nodes representing vertices forming p with an infinite capacity. We also create an edge from the source node s to every n_p with capacity $c(s,n_p)=1$. Next, we set the capacity between a node n_j representing a vertex in V_j and the target node t as $c(n_j,t)=\frac{1}{i}\gamma m_j$, where $1\leq j\leq i$.

For instance, Figure 3 shows the flow network D for the $\mathcal{P}$-partite subgraph in Figure 2. Star nodes represent each path shown in Figure 2, while diamond, circle, and square nodes represent the three types of vertices in Figure 2. As shown in Figure 3, edges are differentiated by colors. Edges with the same color have the same capacity, as indicated by the sample edges with capacities.

Cut expressions. Now we are ready to show three candidate min-cut expressions for D.

Exp 1: only s in S. In this case, only edges from s to each n_p are cut. As such, the cut expression is:

$$C_1(\{s\},T)=\sum_{n_p\in T} C(\{s\},\{n_p\})=|P|, \tag{11}$$

where |P| is the total number of $\mathcal{P}$ instances contained in $G(\mathcal{V})$.

Exp 2: only t in T. In this case, only edges from nodes representing vertices contained in $\mathcal{V}$ are cut. The cut expression is as follows.

$$C_2=\sum_{n_1\in N_i} c(\{n_1\},\{t\})+\ldots+\sum_{n_i\in N_i} c(\{n_i\},\{t\}) \tag{12}$$

$$=\frac{1}{i}|N_1|\gamma m_1+\ldots+\frac{1}{i}|N_i|\gamma m_i \tag{13}$$

$$=\frac{\gamma}{i}(|N_1|\cdot\ldots\cdot|N_i|)^{\frac{1}{i}}\left(\frac{m_1}{m_1'}+\ldots+\frac{m_i}{m_i'}\right). \tag{14}$$

Exp 3: general case. In this case, some edges are cut from s to nodes representing instances of $\mathcal{P}$, and some edges are cut from nodes representing vertices to t. W.l.o.g, let N_{sj} for all $1\leq j\leq i$ denote the vertices of V_j contained in S, N_{sp} denote the nodes representing instances of $\mathcal{P}$ contained in S. Similarly, the same applies for N_{tj} and N_{tp}. We assume that the set of integers $\{|N_{s1}|,\ldots,|N_{si}|\}$ conforms an iRM-set $\mathcal{M}'$, where $\mathcal{M}'$ and $\mathcal{M}$ may be the same or different. Then, the cut can be expressed as:

$$C_3 = \sum_{n_p \in N_{tp}} c(\{s\},\{n_p\}) + \sum_{n_1 \in N_{s1}} c(\{n_1\},\{t\}) + \ldots + \sum_{n_i \in N_{si}} c(\{n_i\},\{t\}) \quad (15)$$

$$= |P| - \sum_{n_p \in N_{sp}} c(\{s\},\{n_p\}) + \frac{1}{i}|N_{s1}|\gamma m_1 + \ldots + \frac{1}{i}|N_{si}|\gamma m_i \quad (16)$$

$$= |P| - (\sum_{n_p \in N_{sp}} c(\{s\},\{n_p\}) - \frac{1}{i}(|N_{s1}|\gamma m_1 + \ldots + |N_{si}|\gamma m_i)) \quad (17)$$

$$= |P| - (\underbrace{|N_{sp}|}_{1} - \underbrace{\frac{\gamma}{i}(|N_{s1}| \cdot \ldots \cdot |N_{si}|)^{\frac{1}{i}}(\frac{m_1}{m'_1} + \ldots + \frac{m_i}{m'_i})}_{2}). \quad (18)$$

For instance, Figure 3 shows a possible Exp3, where edges (s,p_4), (s,p_7), (v_1^1,t), (v_1^2,t), (v_1^3,t), (v_2^1,t), (v_2^2,t), (v_3^1,t), and (v_3^2,t) have been cut.

We would like to mirror Equation 18 to Equation 8, i.e., parts 1 and 2 in both equations are in the same format. This indicates that finding min-cut in D can maximize Equation 8. We are now ready to show finding min-cuts in N is equivalent to solving Problem 3.

4.2 Min-cut and problem equivalence

In this subsection, we show how $\mathcal{V}'$ derived by a min-cut of D can be used to solve Problem 3 w.r.t. $\mathcal{M}$ using the following theorems. We use $\rho^*_{\mathcal{M}}$ denote the optimum density w.r.t. $\mathcal{M}$.

Theorem 2. *When a min-cut of D contains $\{N_{s1},\ldots,N_{si}\}$ representing vertices $\mathcal{V}'=\{V_{s1},\ldots,V_{si}\}$ whose $\{|N_{s1}|,\ldots,|N_{si}|\}$ conforms to $\mathcal{M}$, if $\rho(\mathcal{V}')=\gamma=\rho^*_{\mathcal{M}}$, a min-cut of D with $|S|>1$ must be expressed as C_2 or C_3 with the min-cut value of $|P|$.*

Theorem 3. *When a min-cut of D contains $\{N_{s1},\ldots,N_{si}\}$ representing vertices $\mathcal{V}'=\{V_{s1},\ldots,V_{si}\}$ whose $\{|N_{s1}|,\ldots,|N_{si}|\}$ does not conform to $\mathcal{M}$, and C_3 has the value of $|P|$, $\rho(\mathcal{V}')>\rho^*_{\mathcal{M}}$.*

Theorem 4. *When a min-cut of N contains $\{N_{s1},\ldots,N_{si}\}$ representing vertices $\mathcal{V}'=\{V_{s1},\ldots,V_{si}\}$, if $\gamma>\rho^*_{\mathcal{M}}$, regardless whether $\{|N_{s1}|,\ldots,|N_{si}|\}$ conforms to $\mathcal{M}$ or not, a min-cut of D must be expressed as C_1 with the value of $|P|$, i.e., $\rho(\mathcal{V}')>\gamma$.*

Theorem 5. *When a min-cut of D contains $\{N_{s1},\ldots,N_{si}\}$ representing vertices $\mathcal{V}'=\{V_{s1},\ldots,V_{si}\}$, if $\gamma<\rho^*_{\mathcal{M}}$ and $\{|N_{s1}|,\ldots,|N_{si}|\}$ does not conform to $\mathcal{M}$, a min-cut of N must be expressed as C_3 with a min value less than $|P|$ and $\rho(\mathcal{V}')\geq\gamma$.*

Theorem 6. *Let $\gamma'=\rho(V')$, where $\rho(V')$ is derived by a min-cut in the form of C_3 in N with γ, then $\gamma'>\gamma$ must hold unless $\gamma'\geq\rho^*_{\mathcal{M}}$.*

Theorems 2 to 6 are essentially min-cut based expressions for Lemmas 1 to 5. Since they share similar proof ideas, we omit the proofs for Theorems 2 to 6.

The flow-based algorithm for a fixed $\mathcal{M}$. It is trivial to see that by replacing Problem 3 to the above discussed min-cut problem, the iterative approach can either find an optimum density ($\rho^*_{\mathcal{M}}$) (Theorems 2) or *an upper bound* of $\rho^*_{\mathcal{M}}$. We may also use the guess & verification approach and derive similar results.

For instance, by using the iterative approach, let $\mathcal{M}=\{1,1,1\}$ and initial γ be the density of the entire graph shown in Figure 2, i.e., 3.67, the min-cut of D in Figure 3 is shown by the annotated cut edges. In the case, the S partition contains a subgraph induced by $\mathcal{P}$-family $\mathcal{V}'=\{\{v_1^1,v_1^2,v_1^3\},\{v_2^1,v_2^2\},\{v_3^1,v_3^2\}\}$, which has a higher density of 4.36. Then γ is replaced as 4.36. In this case, the min-cut of D becomes Exp 1. By Theorem 4, $\rho^*_{\mathcal{M}}<4.36$. Therefore, the optimum solution w.r.t. $\mathcal{M}$ cannot beat the subgraph induced by $\mathcal{V}'$.

Algorithm 1: Exact algorithm

Input: $G,\mathcal{V}=\{V_1,\ldots,V_i\}$

$\mathbb{M},X,\mathcal{V}^*\leftarrow\emptyset$, generate $\mathbb{M}$ by calling iRMGenerator(1,X);
foreach $\mathcal{M}\in\mathbb{M}$ **do**
  $\gamma=\rho(\mathcal{V})$, construct flow network D based on γ and $\mathcal{M}$;
  $\mathcal{V}'\leftarrow$ find a min-cut of D;
  while $\gamma<\rho(\mathcal{V}')$ **do**
    $\gamma\leftarrow\rho(\mathcal{V}')$, update capacities of D with γ;
    $\mathcal{V}'\leftarrow$ find a min-cut of D;
  If $\rho(\mathcal{V}^*)<\rho(\mathcal{V}')$ **then** $\mathcal{V}^*\leftarrow\rho(\mathcal{V}')$;
return $\mathcal{V}^*$;
Procedure iRMGenerator(j,X)
  if $j==i+1$ **then** calculate $\mathcal{M}$ based on X, $\mathbb{M}\leftarrow\mathbb{M}\cup\{\mathcal{M}\}$, **return**;
  for $z\leftarrow 1$ to $|V_j|$ **do** iRMGenerator($j+1$, $X\cup\{z\}$);

The complete algorithm. The iterative approach for deriving the exact result is outlined in Algorithm 1. The algorithm's steps are self-explanatory, so we will not discuss them in detail.

For instance, let us go back to the example. We have shown that when $\mathcal{M}=\{1,1,1\}$, a $\mathcal{P}$-partite subgraph with a density of 4.36 can be derived. In fact, for all other possible values of $\mathcal{M}$, Algorithm 1 cannot derive a $\mathcal{P}$-partite subgraph with a higher density than 4.36. Therefore, the $\mathcal{P}$-family $\mathcal{V}=\{\{v_1^1,v_1^2,v_1^3\},\{v_2^1,v_2^2\},\{v_3^1,v_3^2\}\}$ induced subgraph is the global optimum result.

Time complexity. The iterative approach runs in $\Theta(|\mathbb{M}||\mathbb{F}|)$ with the parametric flow network [9]. $|\mathbb{F}|$ is $O(|V(D)|^3)$, $|\mathbb{M}|$ and $|V(D)|$ can be bounded by $O((\frac{n}{i})^i)$, where n is the number of vertices in G.

While it is clear that the algorithm runs in polynomial time when $i\ll n$, the time complexity is high in general. We then attempt an approximation algorithm that may have better scalability.

5 APPROXIMATION ALGORITHM

We have demonstrated that $\mathcal{V}^*$ can be found exactly in polynomial time. Despite the solution optimality, the exact algorithm runs slowly. In this section, we look for a greedy-based approximation algorithm that trades off solution accuracy for speed.

Our proposed algorithm differs significantly from the well-known peeling strategy [17]. Instead, the idea is inspired by [1], in which the technique is for a density function with a denominator in the form of a square root of multiplication of sizes for two vertex sets. The main challenge here is to adapt it to our density function, derive a guaranteed approximation ratio and design an efficient algorithm.

Since the algorithm deals with vertices, we introduce some new notations first. v_j denotes a vertex in V_j for $1\leq j\leq i$ (where $V_j\in\mathcal{V}$). $P(v_j,G(\mathcal{V}))$ denotes the set of $\mathcal{P}$-instances containing v_j in $G(\mathcal{V})$. We use $(\cdot)^*$ and $\widehat{(\cdot)}$ to indicate the optimum and approximation, respectively.

Approximation algorithm. Overall speaking, Algorithm 2 peels the graph according to $\mathcal{M}$ while keeping track of the largest density and the corresponding $\mathcal{P}$-family as output. Specifically, for $G(\mathcal{V})$, Algorithm 2 selects a vertex $v_j\in V_j$ $(1\leq j\leq i)$, such that there is no other vertex $v_{j'}\in V_{j'}$ $(1\leq j'\leq i)$ with $\frac{|P(v_{j'},G(\mathcal{V}))|}{m_{j'}}<\frac{|P(v_j,G(\mathcal{V}))|}{m_j}$, where that j' is allowed to be the same as j. Next, v_j and its incident edges are removed. Algorithm 2 then evaluates ρ of the residual

Algorithm 2: Approximation algorithm w.r.t. $\mathcal{M}$

Input: $G(\mathcal{V}),\mathcal{M}$

1 $\widehat{\gamma}\leftarrow 0$, $\widehat{\mathcal{V}}\leftarrow\{\emptyset_1,\ldots,\emptyset_i\}$;
2 **while** $\forall 1\le j\le i\ V_j\neq\emptyset$ **do**
3 let j' be an integer of $[1,i]$ and $v_{j'}$ be a vertex in $V_{j'}$;
4 **for** j=1 *to* i **do**
  // greediness 1
5 $v_j \leftarrow argmin_v\{|P(v,G(\mathcal{V}))|v\in V_j\}$;
  // greediness 2
6 **if** $\frac{|P(v_j,G(\mathcal{V}))|}{m_j}<\frac{|P(v_{j'},G(\mathcal{V}))|}{m_{j'}}$ **then** $j', v_{j'}\leftarrow j, v_j$;
7 update $\mathcal{V}$ by removing $v_{j'}$ from $V_{j'}\in\mathcal{V}$ and remove edges incident to $v_{j'}$;
8 **if** $\rho(\mathcal{V})\ge\widehat{\gamma}$ **then** $\widehat{\gamma}\leftarrow\rho(\mathcal{V})$, $\widehat{\mathcal{V}}\leftarrow\mathcal{V}$;
9 **return** $\widehat{\mathcal{V}}$ and $\widehat{\gamma}$;

$\mathcal{P}$-partite subgraph and updates $\widehat{\rho}$ and $\widehat{\mathcal{V}}$ if ρ of the residual $\mathcal{P}$-partite subgraph is greater than $\widehat{\rho}$. The above steps are repeated for progressively reducing subgraphs until the residual $\mathcal{V}$ is not a $\mathcal{P}$-family, and then Algorithm 2 returns $\widehat{\rho}$ and $\widehat{\mathcal{V}}$ as the result.

Alternative interpretation of Algorithm 2. Given a $\mathcal{P}$ instance, i.e., $(v_1,\ldots,v_i)$, we can assign this instance to one of the vertices. Thus, $\mathcal{P}$ instances in $G(\mathcal{V})$ can be assigned to every vertex, such that the sum of the assignments of each vertex equals the total number of $\mathcal{P}$ instances in $G(\mathcal{V})$. Algorithm 2 allocates the $\mathcal{P}$ instances following the greedy rule. For each $v\in V(G(\mathcal{V}))$, v is assigned $alloc(v)$ $\mathcal{P}$ instances, and $\sum_{j=1}^{i}\sum_{v\in V_j} alloc(v)=|P|$. Let max_j be $max\{alloc(v)|v\in V_j\}$ for $1\le j\le i$.

Approximation analysis. We are ready to show the approximation guarantee.

Upper bound for $\rho^*_{\mathcal{M}}$. Considering $G(\mathcal{V}^*)$ having the local optimal density $\rho^*_{\mathcal{M}}$ and $|P^*_{\mathcal{M}}|$ instances of $\mathcal{P}$, the following inequality holds:

$$|V_1^*|max_1+\ldots+|V_i^*|max_i\ge|P^*_{\mathcal{M}}|. \tag{19}$$

Then, by dividing $(|V_1^*|\cdot\ldots\cdot|V_i^*|)^{\frac{1}{i}}$ on both side of the inequality, we establish an upper bound of $\gamma^*_{\mathcal{M}}$ as follows:

$$\frac{|V_1^*|max_1}{(|V_1^*|\cdot,\ldots,\cdot|V_i^*|)^{\frac{1}{i}}}+\ldots+\frac{|V_i^*|max_i}{(|V_1^*|\cdot\ldots\cdot|V_i^*|)^{\frac{1}{i}}}=\frac{max_1}{m_1}+\ldots+\frac{max_i}{m_i}$$
$$\ge\frac{|P^*_{\mathcal{M}}|}{(|V_1^*|\cdot\ldots\cdot|V_i^*|)^{\frac{1}{i}}}=\gamma^*_{\mathcal{M}}. \tag{20}$$

Upper bound from Algorithm 2. When a vertex $v_j\in V_j$ $(1\le j\le i)$ is removed from the current $G(\mathcal{V})$, by greediness 1 shown in Algorithm 2, since it is $argmin_{v_j}\{|P(v_j,G(\mathcal{V}))|v_j\in V_j\}$, then $|P(v_j,G(\mathcal{V}))|=alloc(v_j)\le\frac{|P(G(\mathcal{V}))|}{|V_j|}$, where $|P(v_j,G(\mathcal{V}))|$ is the number of $\mathcal{P}$ instances in the current $G(\mathcal{V})$. By greediness 2, the following inequality holds.

$$min\{\frac{|P(v_j,G(\mathcal{V}))|}{m_j}|1\le j\le i\}\le(\frac{|P(v_1,G(\mathcal{V}))|}{m_1}\cdot\ldots\cdot\frac{|P(v_i,G(\mathcal{V}))|}{m_i})^{\frac{1}{i}} \tag{21}$$

Then, since $|P(v_j,G(\mathcal{V}))|\le\frac{|P(G(\mathcal{V}))|}{|V_j|}$ and $m_1\cdot\ldots\cdot m_i=1$, Equation 21 can be written as:

$$min\{\frac{|P(v_j,G(\mathcal{V}))|}{m_j}|1\le j\le i\}\le\frac{|P(G(\mathcal{V}))|}{(|V_1|\cdot\ldots\cdot|V_i|)^{\frac{1}{i}}}\le\widehat{\gamma}, \tag{22}$$

where $\widehat{\gamma}$ is the highest density derived by Algorithm 2. Equation 22 establishes an upper bound for any $\frac{alloc(v_j)}{m_j}$ including for $(1\le j\le i)$ $\frac{max_j}{m_j}$, i.e., $\frac{max_j}{m_j}\le\widehat{\gamma}$.

Approximation ratio of Algorithm 2. Based on the above analysis, putting $\frac{max_j}{m_j}\le\widehat{\gamma}$ into Equation 20, $i\widehat{\gamma}\ge\rho^*_{\mathcal{M}}$ holds. As such the approximation ratio of Algorithm 2 is $\frac{1}{i}$.

Considering the $\mathcal{P}$-partite graph in Figure 2 as an input, and let $\mathcal{M}$={1,1,1}. With such $\mathcal{M}$, since m_1=m_2=m_3=1, Algorithm 2 essentially peels vertices according to the number of $\mathcal{P}$ instances that each vertex involves. One possible peeling process is as follows. It first decides to remove v_2^3 according to lines 5 and 6 in Algorithm 2 and then remove v_3^3, which leads to a subgraph with a density of 4.36. After that, Algorithm 2 continues to remove v_1^1, and the remaining subgraph has a density of 4. Algorithm 2 further peels other vertices but cannot lead to a subgraph with a density greater than 4.36. Therefore, 4.36 is the approximate density w.r.t. $\mathcal{M}$. In fact, the subgraph induced by $\{\{v_1^2,v_1^3\},\{v_2^1,v_2^2\},\{v_3^1,v_3^2\}\}$ is the local optimum result w.r.t. while conforming $\mathcal{M}$, having a density of 4.

$\frac{1}{i}$-approximation regardless $\mathcal{M}$. It is clear that after running Algorithm 2 for every $\mathcal{M}$, the maximum density derived is an $\frac{1}{i}$-approximation w.r.t. the optimum density of Problem 1 by the above discussions.

If we continue the example for $\mathcal{M}$={1,1,1}, we eventually have the approximate density of 4.36. As shown in the example in the exact algorithm section, the approximate density is the same as the global optimum result. Therefore, the actual approximation ratio for this example is 1, which is greater than the lower bound $\frac{1}{3}$.

Time complexity. The amortized time complexity of Algorithm 2 is $O(i|P|+i|V_{max}|log(|V_{max}|))$, where $|P|$ is $\mathcal{P}$ instances in $G(\mathcal{V})$ and $|V_{max}|$ is $\max\{|V_1|,\ldots,|V_i|\}$. The first non-trivial cost for the loop (lines 2 to 10) is visiting vertices in $\mathcal{P}$ instances so that, for any residual vertex v, $|P(v,G(\mathcal{V}))|$ can be correctly updated. Another non-trivial cost is to derive the vertex to be deleted. A vertex $v_j\in V_j$ will be updated up to $|V_{max}|$ times. For an update, we need to adjust the heap, which takes up to $O(|V_{max}|)$ aromatized time using a fibonacci heap. Since each vertex is removed at most once, which takes $O(i|V_{max}|log(|V_{max}|))$ time.

In order to achieve a global $\frac{1}{i}$-approximation guarantee, it is necessary to run $\Theta(\frac{n}{i})^i$ instances of Algorithm 2, which compromises its scalability. This motivates us to study pruning techniques.

6 ADVANCED EXACT ALGORITHM

In this section, we propose various pruning techniques first and then propose an advanced exact algorithm.

Rationale. The first bottleneck in the previous algorithm is that in order to obtain the results, all possible iRM-sets need to be checked. It is natural to ask whether trying every iRM-set is necessary. Secondly, we give every vertex in $\mathcal{V}$ an opportunity. However, if a vertex v has very few $P(v,G(\mathcal{V}))$ compared to most of the other vertices, it should be pruned. What is the pruning condition so that vertices residing in $\mathcal{V}^*$ are safe? We will answer these two questions in this section first.

6.1 Pruning iRM-sets

The two pruning ideas are based on Lemmas 1 and 5 and their analyses discussed in Section 3.2.

Pruning condition 1: w.r.t. $\mathcal{M}$, finding min-cuts converges to $\rho^*_{\mathcal{M}}$. In this case, for $\mathcal{M}$, using the guess & verification approach makes the result converge to $\rho^*_{\mathcal{M}}$, shown in Lemma 2. By Equation 9 in the proof of Lemma 2, it is possible that $\frac{m_1}{m_1'}+\ldots+\frac{m_i}{m_i'}>i$ while $\rho(\mathcal{V}')>\rho^*_{\mathcal{M}}$, as such, the following lemma establishes the pruning basis.

Lemma 6. *Given an optimum $\mathcal{V}'$ w.r.t. $\mathcal{M}$ found by the min-cut, if $\mathcal{M}'$ of V' does not conform to $\mathcal{M}$, i.e., $\rho(\mathcal{V}')>\rho^*_{\mathcal{M}}$, then any $\mathcal{M}''$ satisfying the following condition can be pruned.*

$$i \le \frac{m_1}{m_1''}+\ldots+\frac{m_i}{m_i''} \le \frac{m_1}{m_1'}+\ldots+\frac{m_i}{m_i'}. \tag{23}$$

The correctness of Lemma 6 is clear by Equation 9. This is because, to maintain the equality $\rho^*_{\mathcal{M}}=\frac{i\rho(\mathcal{V}')}{\frac{m_1}{m_1'}+\ldots+\frac{m_i}{m_i'}}$, given $\mathcal{M}''$ satisfying the above condition, $\rho(\mathcal{V}')$ has to be smaller.

Pruning condition 2: w.r.t. $\mathcal{M}$, finding min-cuts does not converge to $\rho^*_{\mathcal{M}}$. In this case, as previously discussed, lines 5 to 7 in Algorithm 1 derive a result with density higher than $\rho^*_{\mathcal{M}}$, denoted by $\overline{\rho^*_{\mathcal{M}}}$, and then terminate based on the condition in Theorem 4.

Let $\mathcal{V}'$ be a $\mathcal{P}$-family making Equation 9 holds, i.e., $\rho^*_{\mathcal{M}}=\frac{i\rho(\mathcal{V}')}{\frac{m_1}{m_1'}+\ldots+\frac{m_i}{m_i'}}$. By Theorem 4 and Exp3, we have the fact that, $\frac{i\rho(\mathcal{V}')}{\frac{m_1}{m_1'}+\ldots+\frac{m_i}{m_i'}}<\overline{\rho^*_{\mathcal{M}}}$. More precisely, let $\delta=\overline{\rho^*_{\mathcal{M}}}-\rho^*_{\mathcal{M}}$, the equation below must hold.

$$\overline{\rho^*_{\mathcal{M}}}-\frac{i\rho(\mathcal{V}')}{\frac{m_1}{m_1'}+\ldots+\frac{m_i}{m_i'}}=\delta \tag{24}$$

Similar to Equation 9, given $\mathcal{M}''$ satisfying conditions in Lemma 6, $\rho(\mathcal{V}')$ has to be smaller, so that the equality can hold. This means, the pruning condition in Lemma 6 can be directly applied in this case. However, such a pruning rule does not fully take the advantage of extra δ. We further propose to using the equality below.

$$\overline{\rho^*_{\mathcal{M}}}=\frac{i\rho(\mathcal{V}')}{\frac{m_1}{m_1'}+\ldots+\frac{m_i}{m_i'}}+\delta=\frac{i\rho(\mathcal{V}')}{B}, \tag{25}$$

where B can be computed accordingly. We are ready to propose the lemma below.

Lemma 7. *Given an optimum $\mathcal{V}'$ w.r.t. $\mathcal{M}$ found by the min-cut, a derived upper bound density, and the derived B, then any $\mathcal{M}''$ satisfying the following condition can be pruned.*

$$i \le \frac{m_1}{m_1''}+\ldots+\frac{m_i}{m_i''} \le B. \tag{26}$$

The correctness is clear by the above discussion.

For instance, for the example shown in Figure 2, when Algorithm 1 dealing with $\mathcal{M}$={1,1,1}, Algorithm 1 runs in pruning condition 2. If the pruning condition in Equation 23 is applied, only a few iRM-sets can be pruned. In contrast, by applying the pruning condition in Equation 26, nearly 80% iRM-sets can be pruned. This is because Algorithm 1 finds the global optimum result when dealing with $\mathcal{M}$={1,1,1}, which is greatly larger than the local optimum density w.r.t. $\mathcal{M}$={1,1,1}, and the pruning condition in Equation 26 takes advantages of such a difference.

Discussion. The opportunities of pruning $\mathcal{M}$ rely on two conditions. First, $\rho^*_{\mathcal{M}}$ has been derived. Second, a subgraph with a density larger than $\rho^*_{\mathcal{M}}$ is derived. When dealing with a Problem 3 w.r.t. $\mathcal{M}$, we cannot apply the pruning if the two conditions cannot be satisfied simultaneously.

Algorithm 3: Advanced exact algorithm

Input: $\mathcal{V}$

1 $\mathcal{V}^*\leftarrow\emptyset$;
2 **foreach** $\mathcal{M}\in\mathbb{M}$ **do**
3 compute $\widehat{\mathcal{V}}$ w.r.t. $\mathcal{M}$ using Algorithm 2;
4 **if** $\rho(\widehat{\mathcal{V}})>\rho(\mathcal{V}^*)$ **then** $\mathcal{V}^*\leftarrow\widehat{\mathcal{V}}$;
5 prune $G(\mathcal{V})$ using Pruning Rule 1 using $\rho(\mathcal{V}^*)$ and $\mathcal{M}$;
6 **foreach** *connected $\mathcal{P}$-subgraph in the pruned $G(\mathcal{V})$* **do**
7 call lines 3 to 7 in Algorithm 1 to derive $\mathcal{V}'$;
8 **if** $\rho(\mathcal{V}')>\rho(\mathcal{V}^*)$ **then** $\mathcal{V}^*\leftarrow\mathcal{V}'$;
9 prune $\mathbb{M}$ using the iRM-sets pruning algorithm if one of the pruning conditions is satisfied;
10 **return** $\mathcal{V}^*$;

6.2 Pruning vertex

Let $\rho^*_{\mathcal{M}}$ be the optimum density w.r.t. $\mathcal{M}$, which can be expressed as follows:

$$\frac{\overline{P_1}}{m_1}+\ldots+\frac{\overline{P_i}}{m_i}=i\rho^*_{\mathcal{M}}, \tag{27}$$

where $\overline{P_1},\ldots,\overline{P_i}$ denote average numbers of $\mathcal{P}$ instances that each vertex involves in $V_1^*,\ldots,V_i^*$ respectively. We have the following lemma.

Lemma 8. *Given any optimum $\mathcal{V}^*=\{V_1^*,\ldots,V_i^*\}$ conforming $\mathcal{M}$, $\frac{\overline{P_j}}{m_j}=\rho^*_{\mathcal{M}}$ must hold for $1\le j\le i$.*

Proof. For any optimum result $\mathcal{V}^*$ conforming $\mathcal{M}$, we have the fact that the set of integer $\{|V_1^*|,\ldots,|V_i^*|\}$ must conform to $\mathcal{M}$. Now, suppose that we have arbitrary i real numbers $\alpha_1,\ldots,\alpha_i$ such that $\alpha_1+\ldots+\alpha_i=i\rho^*_{\mathcal{M}}$ and $\frac{\overline{P_j}}{m_j}=\alpha_j$ for $1\le j\le i$, in order to satisfy such α_j, the sizes of $\{|V_1^*|,\ldots,|V_i^*|\}$ must be $\{\frac{|P|}{\alpha_1 m_1},\ldots,\frac{|P|}{\alpha_i m_i}\}$, where $|P|$ is the total number of $\mathcal{P}$ instances in an optimum result w.r.t. $\mathcal{M}$. $\{\frac{|P|}{\alpha_1 m_1},\ldots,\frac{|P|}{\alpha_i m_i}\}$ cannot confirm $\mathcal{M}$ unless $\alpha_1=\ldots=\alpha_i$. This means, $\alpha_1=,\ldots,=\alpha_i=\rho^*_{\mathcal{M}}$.

Lemma 8 immediately leads to the following lemma.

Lemma 9. *In an optimum result $\mathcal{V}^*=\{V_1^*,\ldots,V_i^*\}$ w.r.t. $\mathcal{M}=\{m_1,\ldots,m_i\}$, a vertex v in $G(\mathcal{V}^*)$ must have $|P(v,G(\mathcal{V}^*))|\ge\frac{\rho^*_{\mathcal{M}}}{\frac{1}{m_1}+\ldots+\frac{1}{m_i}}$.*

Proof. For $v\in V(G(\mathcal{V}^*))$, $|P(v,G(\mathcal{V}^*))|\ge\frac{|P|}{|V_1^*|+\ldots+|V_i^*|}$ and by Lemma 8, $\frac{|P|}{|V_1|+\ldots+|V_i|}=\frac{|P|}{\frac{|P|}{\gamma^* m_1}+\ldots+\frac{|P|}{\gamma^* m_i}}$, which leads to $|P(v,G(\mathcal{V}^*))|\ge\frac{\rho^*_{\mathcal{M}}}{\frac{1}{m_1}+\ldots+\frac{1}{m_i}}$.

The above pruning rule can be further generalized as follows.

Pruning rule 1. *Given $\mathcal{V}$, $\mathcal{M}=\{m_1,\ldots,m_i\}$, a global density lower bound γ, any vertex $v\in V(G(\mathcal{V}))$ with $|P(v_j,G(\mathcal{V}))|<\frac{\gamma}{\frac{1}{m_1}+\ldots+\frac{1}{m_i}}$ can be pruned and the pruning propagates until for all v in the residual $G(\mathcal{V})$ cannot be pruned when solving **RAOP** w.r.t. $\mathcal{M}$.*

Initially, the approximation result w.r.t. $\mathcal{M}$ can serve as a lower bound, and the lower bound can be improved during the process.

6.3 Active pruning based algorithm

Now we are ready to show the advanced exact algorithm incorporating the proposed prunings.

Algorithm 3 shows the main steps. When processing $\mathcal{M}$ that cannot be pruned, it first computes an approximate result with respect to $\mathcal{M}$ and checks whether the best density found so far can be improved (lines 3-4). It then prunes vertices based on Pruning rule 1. Afterwards, the algorithm calls an iterative approach for

Table 2: Datasets

Dataset	$\|A\|$	$\|R\|$	$\|V\|$	$\|E\|$	$max(\|\mathcal{P}\|)$
MovieLens [18]	5	4	2,672	104,747	4
DBLP [18]	5	4	37,795	174,851	4
Douban [18]	6	6	37,597	1,714,941	4
DBpedia [34]	414	673	8,970,120	31,216,862	≥ 9
Freebase [34]	1231	1576	89,934,641	464,233,167	≥ 9
cisco(g22) [24]	4	3	16,177	1,390,120	4
cisco(g21) [24]	4	3	52	1282	4

each connected $\mathcal{P}$-subgraph in the pruned graph $G(\mathcal{V})$ and updates $\mathcal{V}^*$ if the derived $\mathcal{V}'$ has a higher density. The algorithm then checks which iRM-set prunings can be applied and prunes iRM-sets as much as possible. Finally, the algorithm returns $\mathcal{V}^*$ once $\mathcal{M}$ becomes an empty set.

Time complexity. Algorithm 3 has a time complexity of $O((\frac{n}{i})^i|\mathbb{F}|)$. Although there are additional costs associated with pruning iRM-sets, vertices, and computing approximate results, these costs are not dominant compared to the cost of computing maximum flows. Algorithm 3 runs significantly faster than the approximate algorithm since it evaluates far fewer iRM-sets.

7 EXPERIMENTAL STUDY

In this section, extensive experiments are conducted to evaluate the effectiveness and efficiency of the proposed model and algorithms.

Datasets. For the study, we use seven real datasets from different applications, as shown in Table 2. The first 5 datasets have been widely used in evaluating cohesive subgraph models for HINs such as [7, 15, 34]. The last two reflect on the application scenario discussed in the introduction and are benchmark networking datasets [24]. MovieLens contains user-rate-movie related data. Its meta-schema essentially consists of two star schemas centered by users and movies, respectively, connected by the user-rate-movie relationship. DBLP contains user-write-paper related data. Similar to MovieLens, the meta-schema of DBLP consists of two star-schemas modelling users' and papers' information, while the two star-schemas are connected via the author-write-paper relationship. Douban contains user-rate-book related data. Its meta-schema again includes two-star schemas connected via the user-rate-book relationship. DBpedia contains more complicated data extracted from Wikipedia. It has complicated schemas linking star schemas centred by infoboxes, webpages for individuals, objects, organizations, etc. Freebase contains the entities and relations under the domains of Music, Film, and TV, where different domains are connected via multi-talented artists, producers, etc. cisco(g22) and cisco(g21) are two networking datasets having the same meta-schema shown in Figure 1. The critical statistics of the datasets are shown in Table 2.

Query meta-path batches. Following similar approaches adopted in [7, 15, 34], for each dataset, pools of meta-paths are generated. For datasets MovieLens, DBLP, cisco(g22), and cisco(g21), all possible meta-paths with $|\mathcal{P}|$ of 3 and 4 are generated. For DBpediaand Freebase, 20 meta-paths are generated, which are meta-paths leading to the top-20 largest $\mathcal{P}$-partite subgraphs for $|\mathcal{P}|$ from 3 to 9, respectively. If there are less than 20 meta-paths for some $|\mathcal{P}|$, all the available meta-paths are generated. To reduce irrational results, unless explicitly explained, for each experiment, we first randomly get non-repeated 10 meta-paths as queries from the query pools discussed above and average the result. Then, we repeat the randomization 5 times and take the average again, which caters to the reported results.

Table 3: Effectiveness evaluation

Dataset	Method	$des\mathcal{P}3$	$des\mathcal{P}4$	HeterSim3	HeterSim4
MovieLens	DPpS	**0.83**	**0.78**	**0.71**	**0.68**
	VDkpC	0.29	0.27	0.32	0.31
	MAvgP	0.67	0.61	0.48	0.38
	iBF	0.72	0.69	0.58	0.52
	rCom	0.46	0.34	0.39	0.33
DBLP	DPpS	**0.78**	**0.72**	**0.83**	**0.79**
	VDkpC	0.37	0.31	0.27	0.21
	MAvgP	0.49	0.41	0.47	0.41
	iBF	0.59	0.52	0.62	0.59
	rCom	0.41	0.39	0.52	0.51
Douban	DPpS	**0.78**	**0.72**	**0.69**	**0.59**
	VDkpC	0.31	0.26	0.23	0.11
	MAvgP	0.53	0.38	0.48	0.41
	iBF	0.49	0.40	0.56	0.45
	rCom	0.39	0.21	0.31	0.29
DBpedia	DPpS	**0.69**	**0.62**	**0.77**	**0.73**
	VDkpC	0.12	0.11	0.21	0.19
	MAvgP	0.31	0.29	0.38	0.32
	iBF	0.39	0.32	0.67	0.61
	rCom	0.31	0.21	0.41	0.38
Freebase	DPpS	**0.66**	**0.58**	**0.69**	**0.68**
	VDkpC	0.09	0.13	0.15	0.13
	MAvgP	0.29	0.21	0.33	0.29
	iBF	0.28	0.31	0.33	0.39
	rCom	0.28	0.23	0.39	0.32

7.1 Effectiveness evaluations on densest $\mathcal{P}$-partite subgraph model

Models for comparisons. To the best of our knowledge, no work focuses on discovering the densest $\mathcal{P}$-partite subgraphs. Instead, we adapt several recently proposed models that can potentially discover cohesive $\mathcal{P}$-partite subgraphs and use these adapted models as baselines for comparing to our method denoted by DPpS.

Meta-path based models. The following two models are considered.

Vertex-disjoint $(k,\mathcal{P})$-core [7]. As discussed, existing meta-path based models mainly focus on grouping vertices of the same types that are extensively connected via instances of meta-path defined structures. The vertex-disjoint $(k,\mathcal{P})$-core (VDkpC) proposed in [7] is used as a representative. This is because VDkpC considers enlarging both vertices of all types in a query $\mathcal{P}$ and therefore, is closer to our model.

Maximizing average instances of $\mathcal{P}$. Given a $\mathcal{P}$-partite graph induced by $\mathcal{V}=\{V_1,...V_i\}$, we also consider the density function $avgDes(\mathcal{V}) = \frac{\mathcal{F}(\mathcal{V}')}{|V_1|+...+|V_i|}$, where $\mathcal{F}(\mathcal{V}')$ is the same as Definition 3. As such, given an HIN G and a query $\mathcal{P}$, the model finds a $\mathcal{P}$-partite subgraph that maximizes the $avgDes$ density function, denoted as MAvgP.

Models generalizing bipartite cohesiveness. In [6], a butterfly-core model is proposed, in which, cohesiveness between two different types of vertices is measured by butterfly degree while cohesiveness over the same type of vertices is measured by coreness. [6] also considers extending the model to multiple types of vertices, which can be adapted as a baseline as follows. Given a $\mathcal{P}$-partite graph, find

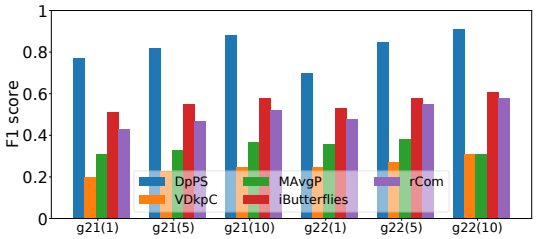


Figure 4: F1 scores for cisco g21 and g22

Table 4: Evaluated variants of the proposed algorithm

Algorithm	Details
ExactGV	For each $\mathcal{M}$, use the guess and verification approach.
ExactIt	For each $\mathcal{M}$, use the iterative appraoch (Algorithm 1).
ExactGVVP	ExactGV plus vertex pruning shown in Lemma 8
ExactItVP	ExactIt plus vertex pruning shown in Lemma 8
AdvExactGV	ExactGV plus Lemma 6 and Lemma 8.
AdvExactGVIt	Algorithm 3 incorporating all the prunings
Approximate	For each $\mathcal{M}$, use Algorithm 2.

a connected $\mathcal{V}$=$\{V_1,...V_i\}$ induced subgraph such that the minimum butterfly degree over every subgraph induced by V_j and V_{j+1} for $1 \le j \le i-1$ is no less than a butterfly degree parameter b. This model is denoted by iBF.

Minimum degree based model. In [14], the relational community model (rCom) is proposed, which is adapted as a baseline as follows. Given a connected $\mathcal{P}$-partite graph induced by $\mathcal{V}$=$\{V_1,...V_i\}$, where $\mathcal{P}$ consists of $i-1$ relationships (edges in the meta-path), the connected $\mathcal{P}$-partite graph is a relational community if, in every bipartite subgraph induced by V_j and V_{j+1} $1 \le j \le i-1$, $\forall v_j \in V_j \wedge v_{j+1} \in V_{j+1}$: $deg(v_j) \ge d_{j(j+1)}$ and $deg(v_{j+1}) \ge d_{(j+1)j}$. Therefore, $2 \cdot (i-1)$ minimum degree constraints are needed.

Parameter configurations. For all methods, meta-paths are generated in the same way as discussed before. Our proposed model and MAvgP are cohesiveness parameter-free, whereas VDkpC, iBF, and rCom require parameters. For VDkpC, iBF, and rCom, we try all possible parameter setups and report the maximum possible value for the corresponding metric.

$\mathcal{P}$-partite cohesiveness evaluation. We report the meta-path density, which is a natural extension of edge density and is defined as follows: Given a $\mathcal{P}$-partite graph induced by $\mathcal{V}$=$\{V_1,...V_i\}$, the meta-path density is $des\mathcal{P}(\mathcal{V}) = \frac{\mathcal{F}(\mathcal{V})}{|V_1| \cdot ... \cdot |V_i|}$. Column $des\mathcal{P}_3$ and $des\mathcal{P}_4$ in Table 3 show the densities for query batches with $|\mathcal{P}|$ of 3 and 4, respectively. For all datasets, our proposed model DPpS clearly has the ability to discover $\mathcal{P}$-partite subgraphs with high meta-path density. This is mainly because DPpS is designed for this purpose, with the suitable searching or approximation objective. It is also worth mentioning that iBF and rCom have the potential to discover high meta-path density $\mathcal{P}$-partite subgraphs as well; however, finding the right parameters is difficult. In comparison, DPpS is cohesive parameters free while preserving the ability to discover high meta-path density $\mathcal{P}$-partite subgraphs. Furthermore, the experimental evaluation also distinguishes DPpS from models aiming to group vertices of the same type, represented by VDkpC. VDkpC has limited ability to discover dense $\mathcal{P}$-partite subgraphs.

Heterogeneous similarity [29] evaluation. We report average heterogeneous similarities (HeteSim) between pairs of vertices connected via instances of $\mathcal{P}$ as well as sub meta-paths of $\mathcal{P}$ in the discovered cohesive $\mathcal{P}$-partite subgraphs. HeteSim is a popular similarity measurement for vertices connected by instances of an asymmetric meta-path. The higher the HeteSim score between two vertices, the more correlated the two vertices are. It is also worth mentioning that PathSim, used in [7, 15, 34], is for measuring vertices of the same type connected via instances of a systematic meta-path. Columns HeteSim3 and HeteSim4 in Table 3 show the HeteSim scores for query batches with $|\mathcal{P}|$ of 3 and 4, respectively. Among all cohesive $\mathcal{P}$-partite subgraphs discovered by various models in the tested datasets, the cohesive $\mathcal{P}$-partite subgraphs discovered by our proposed model, i.e., DPpS, contain more correlated vertices of different types. Especially for DBLP, the average HeteSim scores are as high as 0.83 and 0.79 for $|\mathcal{P}|$ of 3 and 4, respectively. For the schema-enriched datasets DBpedia and Freebase the vertices of different types are not connected as extensively as simple schema datasets such as DBLP. This leads to low scores for all methods. However, DPpS still finds the best $\mathcal{P}$-paritite subgraphs. In contrast, other models are much less effective with reasons similar to before. The experimental results justify the superiority of our proposed model for applications needed to discover highly correlated vertices of different types.

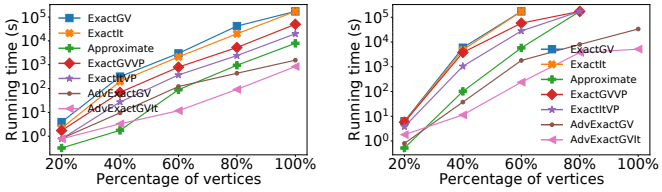
Case study and F1 score evaluation. Previous evaluation demonstrates the advantages of DPpS in discovering strongly correlated and connected vertices of different types. Below we demonstrate that, in real-world application scenarios, DPpS is also effective for grouping vertices of the same type. For the case study, we revisit the cybersecurity scenario discussed in the introduction, in which two networking datasets g21 and g22 [24] were collected with ground truth groups and the same meta-schema shown in Figure 1.

Methodology. The ground truth groups contain non-overlapping members/users, so we borrow ideas from [21] to generate a matched number of groups for evaluation. Every time when the most cohesive $\mathcal{P}$-partite subgraph is generated, induced user-typed vertices are extracted as a resulting group and then removed. This process is repeated until the matched number of groups is reached or no more groups can be generated.

We report the average F1 scores over all groups in Figure 4. For both g21 and g22, we filter the edges in the graph by setting a threshold on the number of packets. For instance, g21(5) denotes that edges in g21 with no less than 5 packets are kept, and so on with other filterings. The first noticeable result is that our proposed DpPS has the highest F1 score among all setups. This is because, in this application scenario, users in the same group communicate with similar servers via similar protocols and port numbers. As such, the networking data within groups exhibit densely connected structures over vertices with types of User, Port Number, Protocol, and Server, which can be captured by our proposed DpPS model. Another interesting result is that by increasing the threshold of the packet number, the F1 scores for all methods increase steadily. This indicates that weights on edges could affect the quality of results in cyber networks. This is because packages could be transferred unintentionally by some unknown accidents such as software bugs, etc. By filtering edges with low weights (low number of packages), the fake interactions over vertices can be eliminated. Therefore, the effectiveness can be improved.

Table 5: Running time (in $\times 100$ seconds) and approximation ratio

Dataset	ExactGV		ExactIt		Approximate		ExactGVVP		ExactItVP		AdvExactGV		AdvExactGVIt	
	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4
MovieLens	-	-	-	-	54 (0.63)	191 (0.58)	324	-	123	-	9.9	27	2.9	6
DBLP	913	-	721	-	41 (0.81)	169 (0.67)	34	-	11	109	6.2	1.1	2.3	5.9
Douban	-	-	-	-	121 (0.49)	984 (0.41)	-	-	-	-	34	89	7.7	16
DBpedia	-	-	-	-	-	-	-	-	-	-	196	453	24	88
Freebase	-	-	-	-	-	-	-	-	-	-	183	517	21	96
cisco(g22)	-	-	-	-	107 (0.76)	789 (0.71)	-	-	-	-	24	91	6.2	22



(a) MovieLens **(b) DBpedia**

Figure 5: Scalability

Table 6: Efficiency of compared models (in $\times 100$ seconds)

Dataset	VDkpC		MAvgP		iBF		rCom	
	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4	$\|\mathcal{P}\|$=3	$\|\mathcal{P}\|$=4
MovieLens	4.3	5.1	1.2	1.6	3.2	4.4	2.1	2.7
DBLP	2.3	2.7	0.7	1.2	2.8	3.2	1.9	2.2
Douban	7.9	9.8	1.5	2.35	4.4	5.3	2.3	2.9
DBpedia	8.3	13	2.6	3.97	12	18	2.4	3.1
Freebase	8.8	17	3.2	4.8	21	31	2.9	3.8

7.2 Evaluations on algorithms and techniques

Implementations. To comprehensively evaluate the performance of the proposed methods, we implemented several variants (i.e., the guess & verification and iterative approaches) of the exact algorithms (Algorithms 1 and 3 with all the pruning) and the approximation-based Algorithm 2. The algorithm names and the adopted techniques are detailed in Table 4, where the first six variations find exact results, the last one finds an approximate result, and AdvExactGVIt serves as DPpS in the effectiveness evaluation. All algorithms are implemented in C++. We also implemented every best online search algorithm of the compared models proposed in the paper in C++. For VDkpC,rCom, and MAvgP, the implemented algorithms find the exact results according to the corresponding model. For iBF, it finds approximate results since only approximate algorithms have been proposed.

Measures. We measure the algorithm runtime with the total CPU time (in seconds), excluding the I/O cost of loading the graph from disk to main memory. A timeout of 48 hours is set, denoted as '-'. Experiments are conducted on a PC with an AMD 3900x CPU, 32GB DDR4 3600Hz memory, and Windows 11 (build 22621.1555). Each algorithm is run no less than 10 times if the running time is less than 24 hours (3 times otherwise). The average results are reported.

Scalability w.r.t. data size. For each dataset, we select fixed percentages of vertices in each type of vertices and compute their induced subgraphs. For each fixed percentage other than 100% in Figure 5, 10 different sets of $\mathcal{V}$ are randomly selected for generating 10 subgraphs on which the algorithms work. The test meta-paths consist of three types of vertices. The average results for MovieLens and DBpedia are reported, and the other datasets have similar results. Since both ExactGV and ExactIt have to try every $\mathcal{M}$ on unpruned subgraphs, they have limited scalability even on MovieLens, i.e., they cannot finish within the cut-off time when there are 100% of vertices. Other algorithms on MovieLens can finish within the cut-off time; however, the runtime difference is dramatic. The two algorithms, AdvExactGV and AdvExactGVIt, scale much better even compared to Approximate. On the large dataset DBpedia, the scalability differences are more dramatic. Only AdvExactGV and AdvExactGVIt can scale to the complete dataset.

Scalability w.r.t. $|\mathcal{P}|$. Table 7 shows the stability of AdvExactGV and AdvExactGVIt when varying $\mathcal{P}$ from 5 to 9. The results are dramatic since as $|\mathcal{P}|$ increases, the running time of each method goes down steadily rather than increases greatly for the two datasets. This is different from the cases when $|\mathcal{P}|$ increases from 3 to 4. The reason is that as $|\mathcal{P}|$ increases, the corresponding connected $\mathcal{P}$-partite subgraphs become smaller since distant types of vertices have weaker relationships. In [7], they obverse similar trends and therefore just consider short $\mathcal{P}$ to conduct the experiments.

Running Time. We report the running times of all algorithms on each dataset for meta-paths comprising 3 and 4 types of vertices. We present and analyze the results via the following comparisons.

Guess & verification vs iterative approach. In Table 5, columns ExactGV and ExactIt show the running times of the exact algorithms that use our proposed flow network but apply different strategies when processing a given $\mathcal{M}$. However, they have limited ability to deal with large datasets since both of them have to try every possible $\mathcal{M}$. Nevertheless, ExactIt runs slightly better than ExactGV on DBLP. The reason is that for most $\mathcal{M}$, ExactIt only runs two or just one min-cut algorithm. That is, for a given $\mathcal{M}$, ExactIt generates a subgraph with near-global optimum density and then it ends $\mathcal{M}$ via the condition in Theorem 4. After that, for other $\mathcal{M}'$, since a near-global optimum density has been generated, ExactIt just ends $\mathcal{M}'$ via the condition in Theorem 4. In contrast, for each $\mathcal{M}$, ExactGV has to attempt many possible γ values via a binary approach so that it ends $\mathcal{M}$ with either Theorem 2 or Theorem 3. The above discussion explains why ExactIt runs faster than ExactGV. Additionally, as shown in columns ExactGVVP and ExactItVP in Table 5, the performance difference between the two algorithms under the vertex pruning technique is more apparent. This is also because ExactItVP can derive near-global optimum results quicker, so the vertex pruning becomes more effective according to Lemma 9.

Table 7: Scalability w.r.t. $|\mathcal{P}|$ (in seconds)

Dataset	$\|\mathcal{P}\|$=5	$\|\mathcal{P}\|$=6	$\|\mathcal{P}\|$=7	$\|\mathcal{P}\|$=8	$\|\mathcal{P}\|$=9
DBpedia	8,792	8,139	7,321	5,201	3,989
Freebase	10,321	8,761	7,801	6,891	6,173

Vertex pruning vs iRM-Set pruning. In Table 5, columns ExactGVVP and ExactItVP report the running times for the exact algorithms with the vertex pruning technique, and AdvExactGV and AdvExactGVIt display the running times under both vertex and iRM-set pruning. However, the further step of applying iRM-set pruning, makes AdvExactGV and AdvExactGVIt significantly faster. Especially for AdvExactGVIt, it can finish reasonably fast (i.e., 9,600 seconds) on the largest dataset Freebase. The results verify the effectiveness of the proposed iRM-set pruning technique and demonstrate that pruning iRM-sets is a more critical step for our studied problem. This is understandable since reducing the number of evaluated iRM-sets clearly saves time for finding min-cuts.

AdvExactGV vs AdvExactGVIt. By further analyzing the running time differences of AdvExactGVIt and AdvExactGV, the importance of iRM-set pruning becomes clearer. By pruning more iRM-sets, AdvExactGVIt runs several times faster than AdvExactGV, and the larger the datasets are, the more significant the differences are.

Approximation vs AdvExactGVIt. Finally, it is interesting to see that the running times of Approximate are much slower than our best exact algorithm AdvExactGVIt. This is mainly because AdvExactGVIt evaluates much fewer $\mathcal{M}$, and for each $\mathcal{M}$ that cannot be pruned, vertex pruning further reduces the cost. It is worth mentioning that Approximate has the ability to find $\mathcal{P}$-partite subgraphs with high density after trying a few $\mathcal{M}$. However, such high density cannot be utilized for pruning iRM-sets for Approximate as the pruning is based on the results of min-cut. It is also worth mentioning that AdvExactGVIt takes advantage of Approximate at the early stage, which is another reason for making AdvExactGVIt fast.

Efficiencies of different models. We report the time cost for each compared model that finds the optimum result for each metric in Table 6. The first observation is that they are not running as fast as reported in the corresponding papers. This is because they have to attempt different parameters to maximise the corresponding metric. The second is that our proposed best algorithm (AdvExactGVIt in Table 5) runs slower than all the compared models in almost all cases because the complexity of our algorithm is high. It is worth noticing that AdvExactGVIt does not slow greatly, given that the effectiveness of our proposed model is clearly better. Our model is a great choice if the user needs high-quality $\mathcal{P}$-partite subgraphs and would like to spend reasonable time.

Actual approximation ratio. The actual approximation ratios of the approximate algorithm are reported in column Approximate in Table 5, appended after the running times in the brackets. As guaranteed by the theoretical approximation ratio of $\frac{1}{|\mathcal{P}|}$, the actual ratios are not only larger than $\frac{1}{3}$ and $\frac{1}{4}$ but larger by clear margins. For instance, when $|\mathcal{P}|$=4 for cisco(g21), the actual approximation ratio 0.71 is 2.84 times better than the theoretical approximation ratio 0.25. Since approximate results can be used for pruning purposes, this also contributes to the advanced exact algorithms.

8 RELATED WORK

Cohesive subgraph search in HINs. Recently, many novel cohesive models have been proposed for HINs based on well-studied cohesive models in unipartite and bipartite graphs. For instance, k-core model has been adapted to $(k,\mathcal{P})$-core [7, 15] and relational community [14], k-truss to $(k,\mathcal{P})$-truss [34], and k-clique to m-clique [13]. The Butterfly-core model [6] combines both the k-core and k-bitruss models. Most of them aim to group the same type of vertices cohesively and target different application scenarios. A few of them, such as [14, 32, 33] can be adapted to search cohesive multipartite subgraphs, which require users to have great prior knowledge in specifying input cohesive parameters. In contrast, our model is free of cohesive parameters.

Cohesive bipartite subgraph search. Searching for cohesive subgraphs in bipartite graphs has also drawn considerable attention as a special instance of HINs. Popular models, along with their corresponding searching algorithms, have been studied, ordered by their searching difficulties, (α,β)-core [19], bitruss [32, 33], densest bipartite subgraph [28], biclique [4, 21], and biplex [20, 35], where the first three models can be solved in polynomial time and the rest cannot due to NP-hardness. These models for bipartite networks are unsuitable for our problem since a bipartite network consists of just one type of edge. Compared to our model, the densest bipartite subgraph model is the closest one, and our model utilizes instances of a meta-path consisting of multiple edges of different types to tightly bind multiple types of vertices. Our proposed algorithms run in polynomial time but have a higher time complexity than [28] since our model nontrivially generalizes the densest bipartite subgraph model to the densest $\mathcal{P}$-partite subgraph case.

Cohesive subgraph search in multipartite graphs. There are a few works [5, 36] focusing on cohesive subgraph search in multipartite graphs, which are a generalization of bipartite graphs. These models focus more on clique-based models in star multipartite graphs and lead to NP-hard searching problems. In contrast, our proposed problem is meta-path driven and can be solved in polynomial time.

Densest subgraph (DS) problem. The original DS problem studies maximizing the ratio of the number of edges and the number of vertices in a subgraph [11]. This problem has been extended to higher-order DS problems, such as [30] and [3]. The DS problem was also studied on directed graphs [16]. Recently, many breakthroughs have been made in approximating DS in unipartite graphs (including directed graphs). The breakthroughs involved the novel approaches of convex programming [26], flow and supermodularity techniques [2], which generalize $\frac{1}{2}$-approximation to $(1-\epsilon)$-approximation. In [2], the authors showed that the directed DS problem can be reduced to the vertex-weighted DS problem and connections exist among variants of the DS problem in unipartite graphs. In practice, more efficient methods have been proposed, such as those in [8, 22]. Despite the novel breakthroughs, a key technical factor that bridges the directed densest subgraph problem to the vertex-weighted densest subgraph problem is a ratio parameter proposed in [1]. The parameter was used in all the latest methods. In this paper, we generalize such a ratio parameter to our iRM-set, which can deal with a density function whose denominator is in a multiplication form of multiple vertex sets instead of just two.

9 CONCLUSION

In this paper, we propose a novel densest $\mathcal{P}$-partite subgraph model. We first introduce an exact algorithm to search for the densest $\mathcal{P}$-partite subgraph model, which requires solving $O((\frac{n}{i})^i)$ instances of the min-cut problem. To achieve reasonable scalability, we propose a $\frac{1}{i}$-approximation algorithm. Although the approximation algorithm can avoid the expensive min-cut problem, it still needs to solve $O((\frac{n}{i})^i)$ instances of the peeling subproblem. Therefore, we propose novel pruning rules that can significantly reduce subproblem instances that need to be solved and the input size of each aforementioned subproblem. The advanced exact algorithm equipped with the proposed prunings can even outperform the approximation algorithm on real datasets. We conducted extensive experiments to justify the effectiveness of the densest $\mathcal{P}$-partite subgraph model and the efficiency of the proposed algorithms.

REFERENCES

[1] Moses Charikar. 2003. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization: Third International Workshop, APPROX 2000 Saarbrücken, Germany, September 5–8, 2000 Proceedings*. Springer, 84–95.

[2] Chandra Chekuri, Kent Quanrud, and Manuel R. Torres. [n. d.]. *Densest Subgraph: Supermodularity, Iterative Peeling, and Flow*. 1531–1555. https://doi.org/10.1137/1.9781611977073.64

[3] Lu Chen, Chengfei Liu, Kewen Liao, Jianxin Li, and Rui Zhou. 2019. Contextual Community Search Over Large Social Networks. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 88–99. https://doi.org/10.1109/ICDE.2019.00017

[4] Lu Chen, Chengfei Liu, Rui Zhou, Jiajie Xu, and Jianxin Li. 2021. Efficient Exact Algorithms for Maximum Balanced Biclique Search in Bipartite Graphs. In *SIGMOD*. ACM, 248–260.

[5] Milind Dawande, Pinar Keskinocak, Jayashankar M Swaminathan, and Sridhar Tayur. 2001. On bipartite and multipartite clique problems. *Journal of Algorithms* 41, 2 (2001), 388–403.

[6] Zheng Dong, Xin Huang, Guorui Yuan, Hengshu Zhu, and Hui Xiong. 2021. Butterfly-Core Community Search over Labeled Graphs. *Proc. VLDB Endow.* 14, 11 (jul 2021), 2006–2018. https://doi.org/10.14778/3476249.3476258

[7] Yixiang Fang, Yixing Yang, Wenjie Zhang, Xuemin Lin, and Xin Cao. 2020. Effective and Efficient Community Search over Large Heterogeneous Information Networks. *Proc. VLDB Endow.* 13, 6 (mar 2020), 854–867. https://doi.org/10.14778/3380750.3380756

[8] Yixiang Fang, Kaiqiang Yu, Reynold Cheng, Laks V. S. Lakshmanan, and Xuemin Lin. 2019. Efficient Algorithms for Densest Subgraph Discovery. *Proc. VLDB Endow.* 12, 11 (2019), 1719–1732. https://doi.org/10.14778/3342263.3342645

[9] Giorgio Gallo, Michael D Grigoriadis, and Robert E Tarjan. 1989. A fast parametric maximum flow algorithm and applications. *SIAM J. Comput.* 18, 1 (1989), 30–55.

[10] Yali Gao, Xiaoyong Li, Hao Peng, Binxing Fang, and Philip S. Yu. 2022. HinCTI: A Cyber Threat Intelligence Modeling and Identification System Based on Heterogeneous Information Network. *IEEE Transactions on Knowledge and Data Engineering* 34, 2 (2022), 708–722. https://doi.org/10.1109/TKDE.2020.2987019

[11] AV Goldberg. 1984. *Finding a Maximum Density Subgraph.* Technical Report. Berkeley, CA, USA.

[12] Mukul Gupta and Pradeep Kumar. 2020. Recommendation generation using personalized weight of meta-paths in heterogeneous information networks. *European Journal of Operational Research* 284, 2 (2020), 660–674. https://doi.org/10.1016/j.ejor.2020.01.010

[13] Jiafeng Hu, Reynold Cheng, Kevin Chen-Chuan Chang, Aravind Sankar, Yixiang Fang, and Brian Y.H. Lam. 2019. Discovering Maximal Motif Cliques in Large Heterogeneous Information Networks. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. 746–757. https://doi.org/10.1109/ICDE.2019.00072

[14] Xun Jian, Yue Wang, and Lei Chen. 2021. Effective and Efficient Relational Community Detection and Search in Large Dynamic Heterogeneous Information Networks. *Proc. VLDB Endow.* 13, 10 (mar 2021), 1723–1736. https://doi.org/10.14778/3401960.3401969

[15] Yangqin Jiang, Yixiang Fang, Chenhao Ma, Xin Cao, and Chunshan Li. 2022. Effective Community Search over Large Star-Schema Heterogeneous Information Networks. *Proc. VLDB Endow.* 15, 11 (sep 2022), 2307–2320. https://doi.org/10.14778/3551793.3551795

[16] Ravi Kannan and V. Vinay. 1999. Analyzing the structure of large graphs. In *manuscript*.

[17] Samir Khuller and Barna Saha. 2009. On Finding Dense Subgraphs *(ICALP)*. Springer-Verlag, Berlin, Heidelberg, 597–608. https://doi.org/10.1007/978-3-642-02927-1˙50

[18] librahu. 2019. HIN-Datasets-for-Recommendation-and-Network-Embedding. https://github.com/librahu/HIN-Datasets-for-Recommendation-and-Network-Embedding

[19] Boge Liu, Long Yuan, Xuemin Lin, Lu Qin, Wenjie Zhang, and Jingren Zhou. 2019. Efficient (α, β)-Core Computation: An Index-Based Approach. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 1130–1141. https://doi.org/10.1145/3308558.3313522

[20] Wensheng Luo, Kenli Li, Xu Zhou, Yunjun Gao, and Keqin Li. 2022. Maximum Biplex Search over Bipartite Graphs. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 898–910. https://doi.org/10.1109/ICDE53745.2022.00072

[21] Bingqing Lyu, Lu Qin, Xuemin Lin, Ying Zhang, Shengping Qian, and Jingren Zhou. 2020. Maximum Biclique Search at Billion Scale. *PVLDB* 13, 9 (2020), 1359–1372.

[22] Chenhao Ma, Yixiang Fang, Reynold Cheng, Laks V. S. Lakshmanan, and Xiaolin Han. 2022. A Convex-Programming Approach for Efficient Directed Densest Subgraph Discovery. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 845–859. https://doi.org/10.1145/3514221.3517837

[23] Xiu Ma, Leiqi Wang, Qiujian Lv, Yan Wang, Qi Zhang, and Jianguo Jiang. 2022. CyEvent2vec: Attributed Heterogeneous Information Network based Event Embedding Framework for Cyber Security Events Analysis. In *2022 International Joint Conference on Neural Networks (IJCNN)*. 01–08. https://doi.org/10.1109/IJCNN55064.2022.9892291

[24] Omid Madani, Sai Ankith Averineni, and Shashidhar Gandham. 2022. A Dataset of Networks of Computing Hosts. In *Proceedings of the 2022 ACM on International Workshop on Security and Privacy Analytics*. 100–104.

[25] Saurabh Sawlani and Junxing Wang. 2020. Near-optimal fully dynamic densest subgraph. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. 181–193.

[26] Saurabh Sawlani and Junxing Wang. 2020. Near-Optimal Fully Dynamic Densest Subgraph. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing* (Chicago, IL, USA) *(STOC 2020)*. Association for Computing Machinery, New York, NY, USA, 181–193. https://doi.org/10.1145/3357713.3384327

[27] Siegfried Schaible and Toshidide Ibaraki. 1983. Fractional programming. *European journal of operational research* 12, 4 (1983), 325–338.

[28] Dhara Shah, Sushil Prasad, and Danial Aghajarian. 2019. Finding densest subgraph in a bi-partite graph. (2019).

[29] Chuan Shi, Xiangnan Kong, Yue Huang, S Yu Philip, and Bin Wu. 2014. Hetesim: A general framework for relevance measure in heterogeneous networks. *IEEE Transactions on Knowledge and Data Engineering* 26, 10 (2014), 2479–2492.

[30] Charalampos Tsourakakis. 2015. The K-Clique Densest Subgraph Problem. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 1122–1132. https://doi.org/10.1145/2736277.2741098

[31] Charalampos Tsourakakis. 2015. The k-clique densest subgraph problem. In *Proceedings of the 24th international conference on world wide web*. 1122–1132.

[32] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2019. Vertex Priority Based Butterfly Counting for Large-scale Bipartite Networks. *PVLDB* (2019).

[33] Kai Wang, Xuemin Lin, Lu Qin, Wenjie Zhang, and Ying Zhang. 2020. Efficient bitruss decomposition for large-scale bipartite graphs. In *ICDE*. IEEE, 661–672.

[34] Yixing Yang, Yixiang Fang, Xuemin Lin, Wenjie Zhang, and Yixiang Fang. 2020. Effective and Efficient Truss Computation over Large Heterogeneous Information Networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 901–912. https://doi.org/10.1109/ICDE48307.2020.00083

[35] Kaiqiang Yu, Cheng Long, Shengxin Liu, and Da Yan. 2022. Efficient Algorithms for Maximal K-Biplex Enumeration. In *Proceedings of the 2022 International Conference on Management of Data* (Philadelphia, PA, USA) *(SIGMOD '22)*. Association for Computing Machinery, New York, NY, USA, 860–873. https://doi.org/10.1145/3514221.3517847

[36] Alexander Zhou, Yue Wang, and Lei Chen. 2020. Finding Large Diverse Communities on Networks: The Edge Maximum K*-Partite Clique. *Proc. VLDB Endow.* 13, 12 (sep 2020), 2576–2589. https://doi.org/10.14778/3407790.3407846