The 16th International Conference on Advances in Social Networks Analysis and Mining -ASONAM-2024
ASONAM 2024
University of Calabria, Rende (CS), Calabria, Italy, September 02-05, 2024

General-purpose query processing on summary graphs

Francesco Gullo

University of L'Aquila – DISIM Department

Italy

francesco.gullo@univaq.it

https://fgullo.github.io/

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

DISIM Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

September 3rd, 2024

Joint work with

Aris
Anagnostopoulos

SAPIENZA
UNIVERSITÀ DI ROMA

Valentina
Arrigoni

Lorenzo
Severini

UniCredit

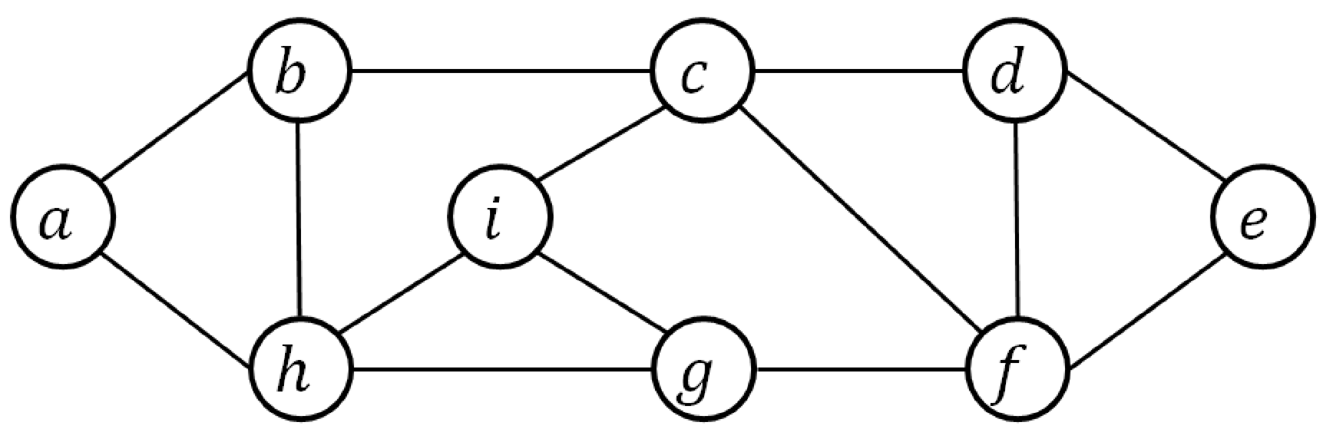
Giorgia
Salvatori

REPLY
TARGET

Thank you!

Introduction

Graphs



entities of interest (nodes) linked to one another via relationships (edges)

Graphs are ubiquitous!

Internet

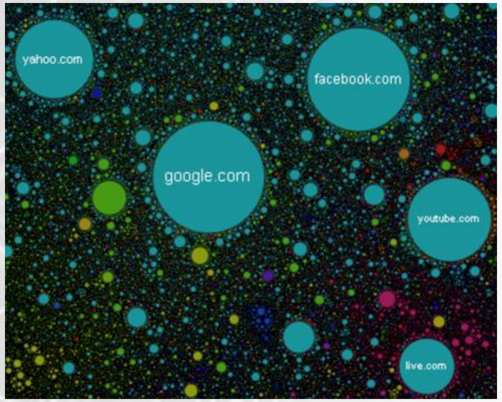


image source

Web

image source

computer networks

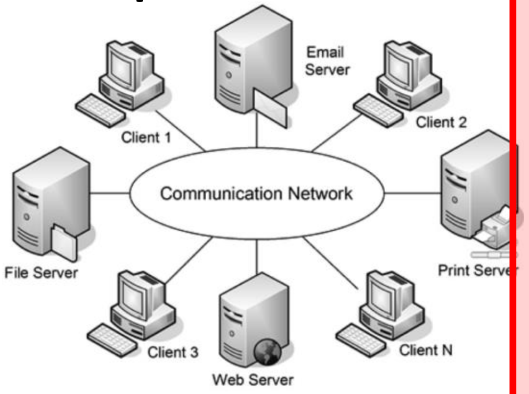


image source

social networks

image source

subway maps

image source

food web

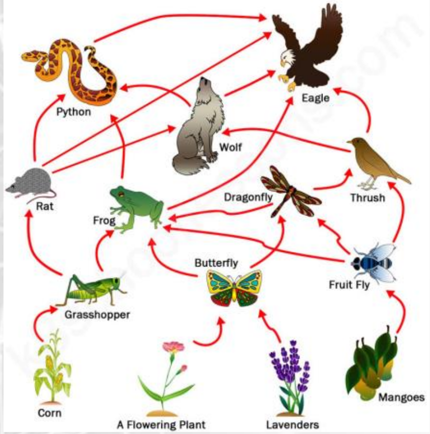


image source

protein networks

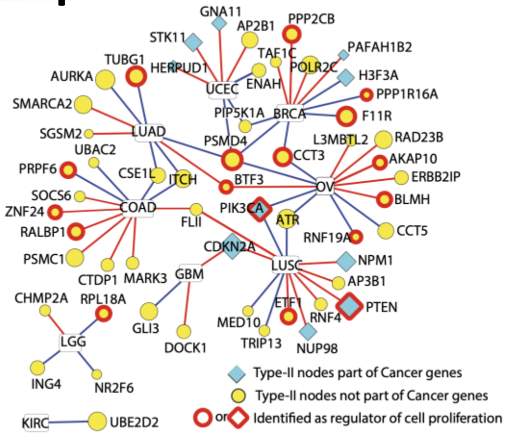


image source

financial networks

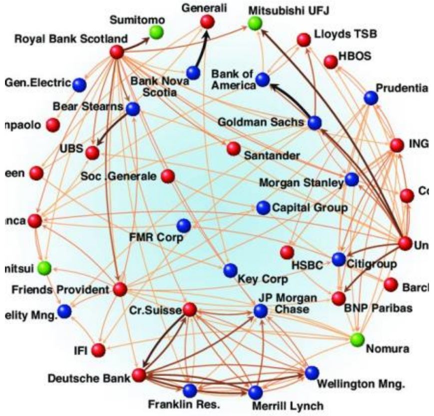


image source

text networks

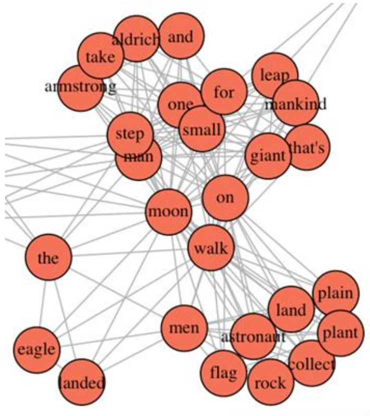


image source

road networks

image source

Today’s real graphs may be gigantic!

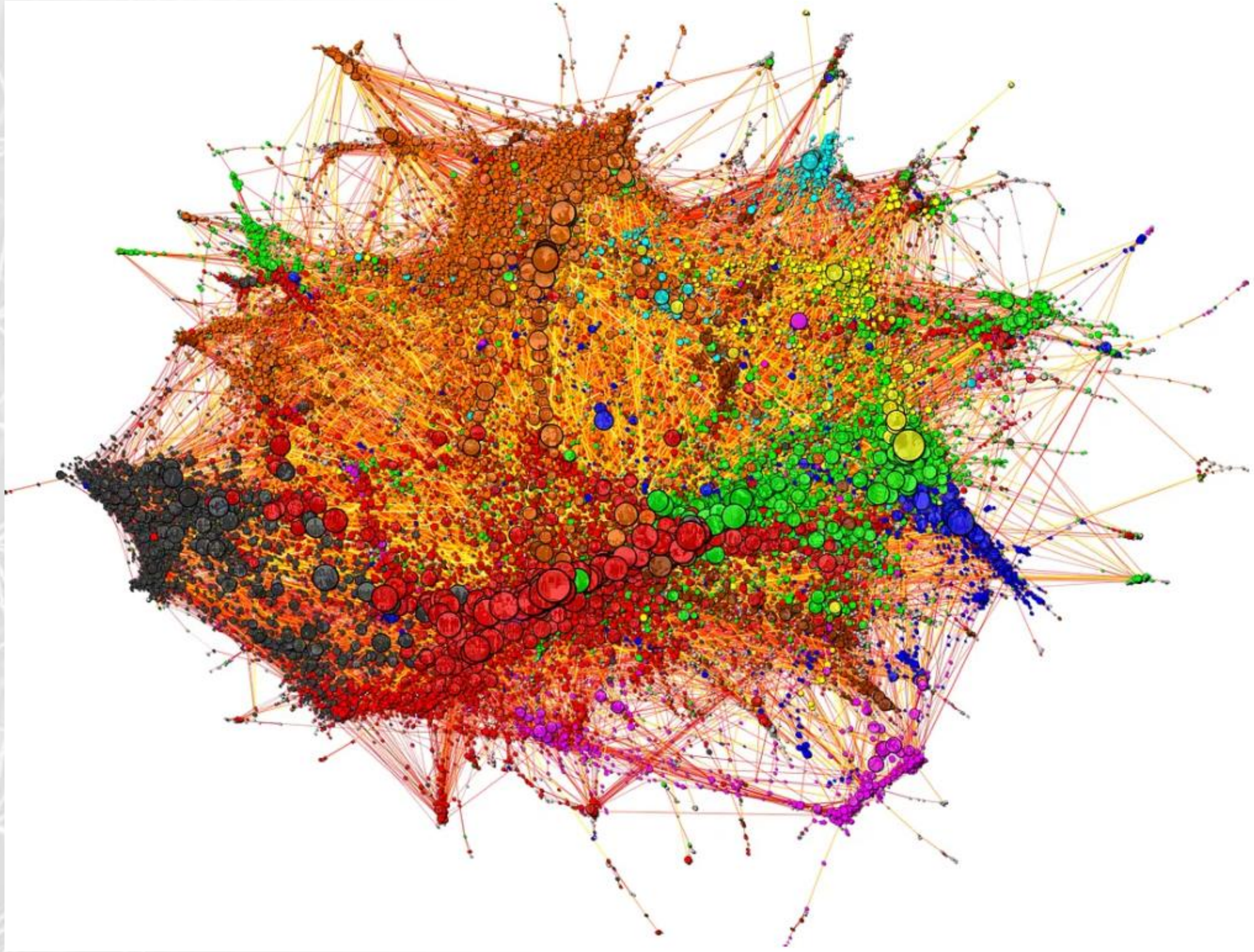
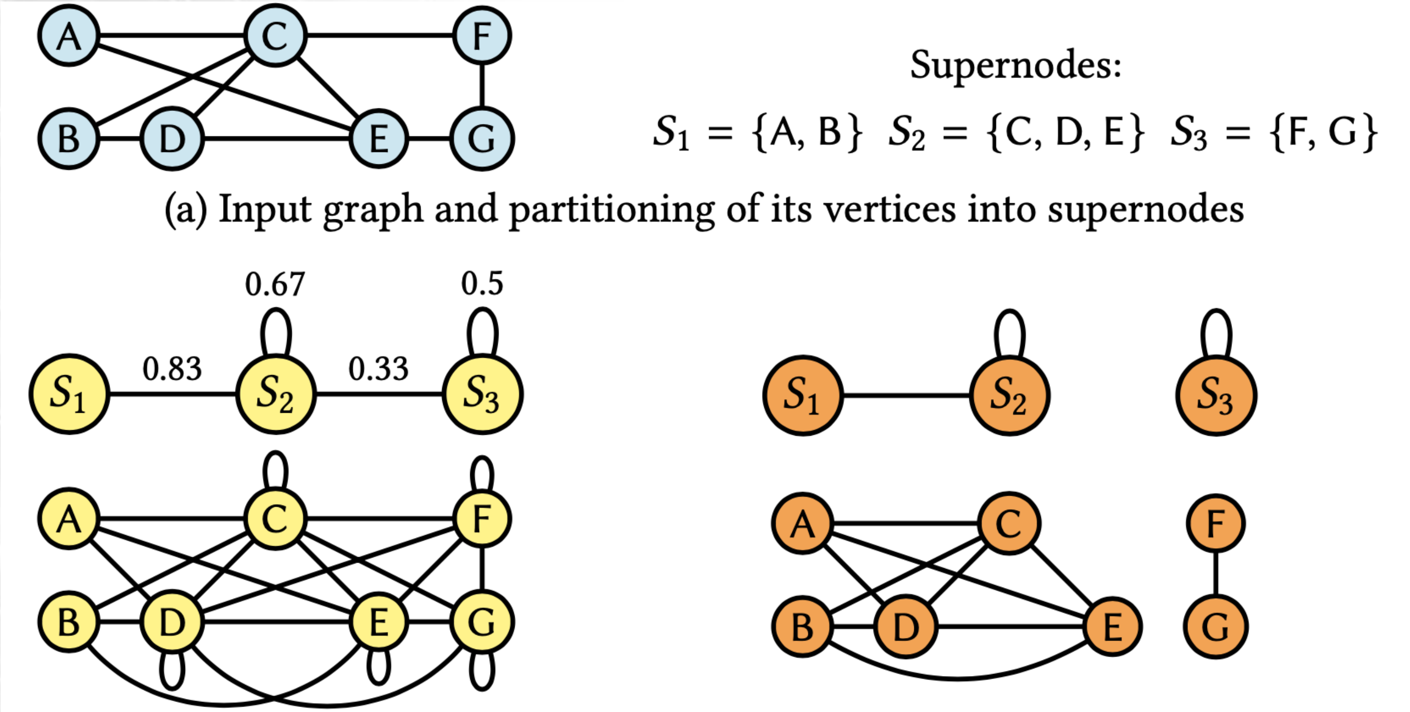


image source

The size of today’s real graphs may be huge: they can count billions nodes/edges or even more!

Graph compression is a valuable option to process big graphs in a more efficient and sustainable way.

Graph summarization



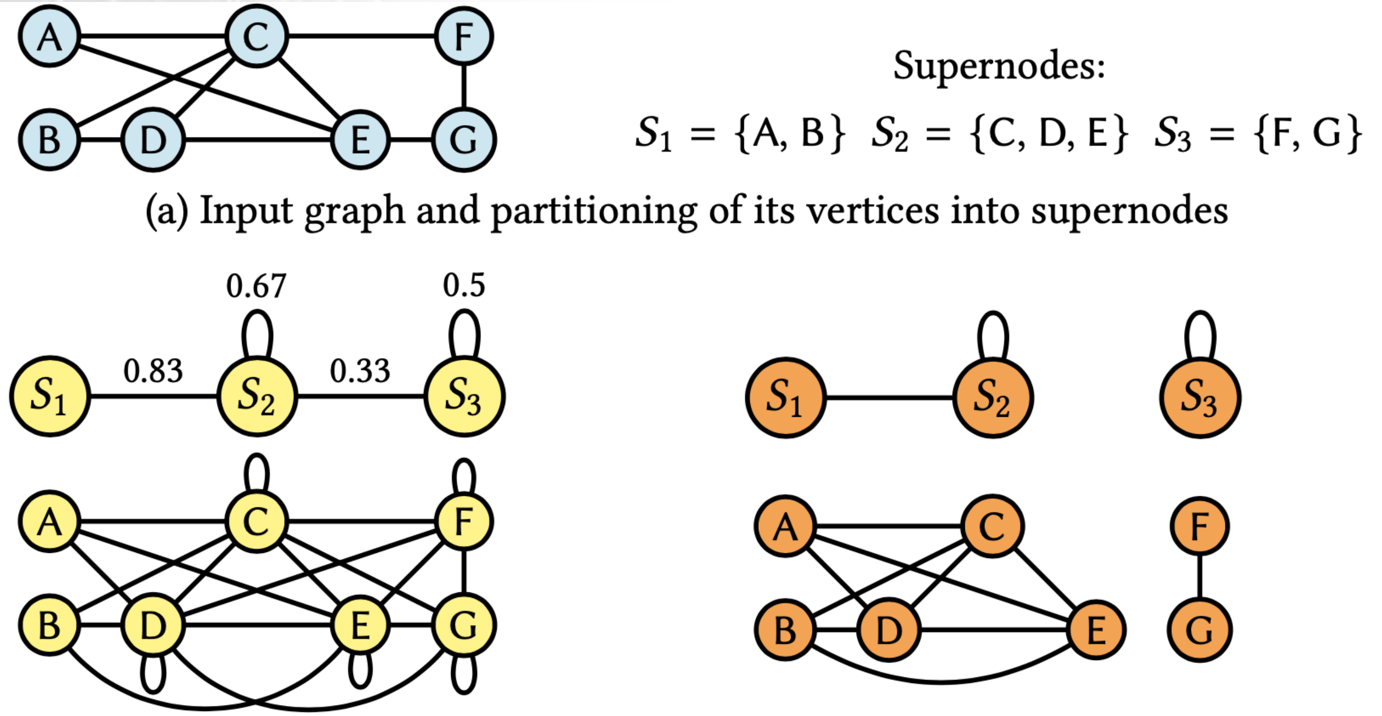
(a) Input graph and partitioning of its vertices into supernodes

(b) Summary graphs (Def. 2.1) and corresponding reconstructed graphs (Def. 2.2) according to S2L [70] (left) and SWeG [74] (right)

Graph summarization is one type (among the many existing ones) of graph compression which produces a summary graph.

Summary graph: coarse-grained representation of a graph in terms of supernodes and superedges.

Graph summarization: benefits



(a) Input graph and partitioning of its vertices into supernodes

(b) Summary graphs (Def. 2.1) and corresponding reconstructed graphs (Def. 2.2) according to S2L [70] (left) and SWeG [74] (right)

- **No need for redefining graph-processing methods**
  - A summary graph is a graph by itself!
- **No loss of information**
  - Every node is part of (at least) a supernode

Motivation

Existing query-processing methods on summary graphs are either:

- General-purpose, but reconstruct the original graph on-the-fly, while processing the query
- Special-purpose (i.e., query-specific)

No query-processing method on summary graphs exist that are **general-purpose** and **use the summary graph only** (without reconstructing the input graph).
=> In this work we fill this gap!

Contributions

We study for the first time the problem of *general-purpose (approximate) query processing on summary graphs* (GPQPS)

- We set the stage of this problem

We devise algorithms for GPQPS

We set up an evaluation methodology that constitutes a benchmark testbed for this and future GPQPS studies

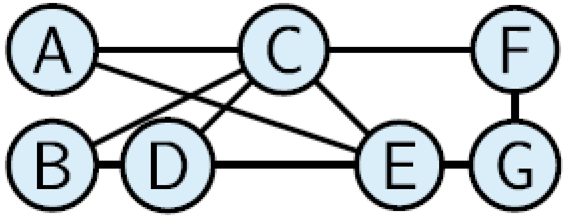
We perform extensive experiments

We provide nontrivial directions for further research

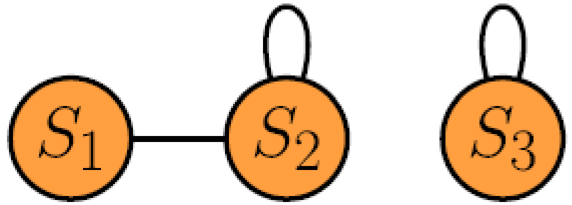
Problem definition

Summary graph

Definition 1 (Summary graph) A *summary graph* (or, simply, a *summary*) of a given graph $G = (V, E, w)$ is a *directed* and (possibly) *edge-weighted* graph $\mathcal{G} = (\mathcal{S}, \mathcal{E}, \omega)$, with vertices $\mathcal{S}$, edges $\mathcal{E} \subseteq \mathcal{S} \times \mathcal{S}$, and edge-weighting function $\omega : \mathcal{E} \rightarrow \mathbb{R}_{>0}$, such that every vertex $u \in V$ is assigned to one and only one $S \in \mathcal{S}$: $\forall S \in \mathcal{S} : S \subseteq V$, $\forall S, T \in \mathcal{S}, S \neq T : S \cap T = \emptyset$, $\bigcup_{S \in \mathcal{S}} = V$. We term vertices and edges of a summary *supernodes* and *superedges*, respectively. The supernode to which a vertex $u \in V$ belongs is denoted by S_u. $E(S, T) = \{(u, v) \in E \mid u \in S, v \in T\}$ and $\omega(S, T) = \sum_{e \in E(S,T)} w(e)$ denote the edges and the overall edge weight between all the vertices of supernodes S and T, respectively.



Supernodes: $S_1 = \{AB\}$
$S_2 = \{CDE\}$ $S_3 = \{FG\}$



Graph query

Graph query Q: computable function

Input: (weighted) **graph** $G = (V, E, w)$ and **query context** $\mathcal{C}$
Output: **query answer** $Q(G, \mathcal{C})$, an object from a domain $\mathcal{O}_Q$

Query context $\mathcal{C}$:
complementary input of the query (e.g., pairs or sets of vertices, subgraphs, functions, numerical values; it can also be empty).

Object from **domain** $\mathcal{O}_Q$:
a Boolean, a numerical value, a set of vertices, a subgraph, a partition of the vertices, and so on.

Graph query: examples

Global queries computing **numerical stats** on G (e.g., number of triangles, clustering coefficient, diameter):
$\mathcal{C} = \emptyset, \mathcal{O}_Q = \mathbb{R}$

Node embedding queries:
$\mathcal{C} = \{u\}, \mathcal{O}_Q = \mathbb{R}^d$

Reachability queries:
$\mathcal{C} = \{(u, v)\}, \mathcal{O}_Q = \{True, False\}$

Inner-most core queries:
$\mathcal{C} = \emptyset, \mathcal{O}_Q = 2^V$

Top-ranked centrality queries:
$\mathcal{C} = s, \mathcal{O}_Q = 2^V$

Community detection queries:
$\mathcal{C}$ = parameters of community-detection algorithm, $\mathcal{O}_Q$ = all partitions of V ($\boldsymbol{B}_V$)

Graph query

In this work, we restrict our study to query answers that are either numerical or sets/partitions of vertices,
that is $\mathcal{O}_Q = \{\mathbb{R}, 2^V, \boldsymbol{B}_V\}$

Summary-based approximate query answer

Answer to a query is *approximated by exploiting solely a summary* $\mathcal{G}$ of a graph G, *without accessing* G at all.

Query processing is required to be *agnostic* of both the specific query and the graph-summarization technique that has produced $\mathcal{G}$.

In other words, we are interested in:

Definition 3 (Summary-based approximate query answer) Given a graph G, a summary $\mathcal{G}$ of G, and a query Q on G with context $\mathcal{C}$, a *summary-based approximate answer* to Q on $\mathcal{G}$—denoted $\widetilde{Q}(G, \mathcal{G})$—is an approximation of $Q(G, \mathcal{C})$ obtained by making use only of $\mathcal{G}$.

GPQPS problem

The problem we tackle:

Problem 1 (General-Purpose Query Processing on Summary graphs (GPQPS)) Given a summary $\mathcal{G}$ of a graph G, and a query Q on G with context $\mathcal{C}$, compute $\widetilde{Q}(G, \mathcal{G})$ that is the closest to $Q(G, \mathcal{C})$.

Simply speaking, Problem 1 asks for summary-based query answers which approximate well the true answer to the given query.

Algorithms

Algorithm 1: Naïve-GPQPS

Naïve-GPQPS processes a query Q on summary $\mathcal{G}$ as if it were a normal graph, with the only precaution of letting each vertex u in the input graph G conceptually be identified with the supernode S_u of G it belongs to, and vice versa.

Input: graph $G = (V, E, w)$; summary $\mathcal{G}$ of G; graph query Q; query context $\mathcal{C}$; $\mathbf{c} \in \mathbb{R}^d$ (if $\mathcal{O}_Q = \mathbb{R}^d$)

Output: approximate answer $\widetilde{Q}(G, \mathcal{G})$

$\mathcal{C}_\mathcal{G} \leftarrow$ compute summary-aware context information from $\mathcal{C}$ and $\mathcal{G}$
$Q(\mathcal{G}, \mathcal{C}_\mathcal{G}) \leftarrow$ compute summary-processed query answer
if $\mathcal{O}_Q = \mathbb{R}^d$ **then**
    $\widetilde{Q}(G, \mathcal{G}) \leftarrow \mathbf{c} \circ Q(\mathcal{G}, \mathcal{C}_\mathcal{G})$
else if $\mathcal{O}_Q = 2^{2^V}$ **then**
    $\widetilde{Q}(G, \mathcal{G}) \leftarrow \{\{\bigcup_{S \in \mathbf{S}} S\} \mid \mathbf{S} \in Q(\mathcal{G}, \mathcal{C}_\mathcal{G})\}$
end if

Algorithm 2: Probabilistic-GPQPS

Probabilistic-GPQPS interprets a summary $\mathcal{G}$ as an ***uncertain*** (or ***probabilistic***) graph, that is, a graph whose edges are assigned a **probability of existence**:

Definition 6 (Uncertain graph) An *uncertain* (or *probabilistic*) *graph* is a triple $\mathbb{G} = (V, E, \pi)$, where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, and $\pi : E \rightarrow (0, 1]$ is a function assigning existence probabilities to edges. According to the *possible-world* semantics (Abiteboul et al. 1987; Dalvi and Suciu 2004), an uncertain graph $\mathbb{G} = (V, E, \pi)$ is interpreted as a set $\{G = (V, E_G)\}_{E_G \subseteq E}$ of $2^{|E|}$ deterministic graphs (*worlds*), each defined by a subset of E. Assuming independence among edge probabilities (Khan et al. 2018), the probability of observing any possible world $G = (V, E_G)$ drawn from $\mathbb{G}$ is $\Pr(G) = \prod_{e \in E_G} \pi(e) \prod_{e \in E \setminus E_G} (1 - \pi(e))$.

Edge-probability function π can be defined, e.g., as as the **expected number of edges** between two supernodes:

$$pr(S, T) = |E(S, T)| / (|S| \cdot |T|), \quad \overline{\omega}(S, T) = \omega(S, T) / |E(S, T)|.$$

$$\pi = pr(S_u, S_v) \cdot \overline{\omega}(S_u, S_v)$$

Algorithm 2: Probabilistic-GPQPS

1. Sample a set of worlds from $\mathcal{G}$
2. Compute query answer from any sampled world by Naïve-GPQPS algorithm
3. Aggregate answers from all the worlds into a single ultimate output answer

Input: graph $G=(V,E,w)$; summary $\mathcal{G}=(\mathcal{S},\mathcal{E},\omega)$ of G; function $\pi\colon \mathcal{E} \to (0,1]$; integer K; graph query Q; query context $\mathcal{C}$; clustering-aggregation algorithm AGG (if $\mathcal{O}_Q = \mathbf{B}_V$)

Output: approximate answer $\widetilde{Q}(G,\mathcal{G})$

$\mathcal{W}_1,\ldots,\mathcal{W}_K \leftarrow$ sample K possible worlds from $\mathcal{G}_\pi = (\mathcal{S},\mathcal{E},\pi)$
$\widetilde{Q}(G,\mathcal{W}_i) \leftarrow$ Naïve-GPQPS$(G,\mathcal{W}_i,Q,\mathcal{C})$, for all $i=1,\ldots,K$
if $\mathcal{O}_Q = \mathbb{R}^d$ **then**
    $\widetilde{Q}(G,\mathcal{G}) \leftarrow \frac{1}{K}\sum_{i=1}^{K} \widetilde{Q}(G,\mathcal{W}_i)$
else if $\mathcal{O}_Q = 2^V$ **then**
    $\widetilde{Q}(G,\mathcal{G}) \leftarrow \bigcap_{i=1}^{K} \widetilde{Q}(G,\mathcal{W}_i)$
else if $\mathcal{O}_Q = \mathbf{B}_V$ **then**
    $\widetilde{Q}(G,\mathcal{G}) \leftarrow$ run AGG on $\{\widetilde{Q}(G,\mathcal{W}_i)\}_{i=1}^{K}$
end if

Gionis et al. (2007) and Gullo et al. (2009)

Experiments

Experimental methodology

6 real datasets: Facebook, LastFM, Enron, Gnutella, Ubuntu, AS-Skitter

2 graph-summarization methods:
S2L (Riondato et al., 2017) and SWeG (Shin et al., 2019)

4 queries:

- Clustering coefficient (numerical query, $\mathcal{O}_Q = \mathbb{R}$)
- Community detection (partitioning query, $\mathcal{O}_Q = \boldsymbol{B}_V$)
- Top-ranked centrality and core decomposition (vertex-set queries, $\mathcal{O}_Q = 2^V$)

Experimental methodology

GPQPS methods:

- Naïve-GPQPS; On S2L summaries, in two variants:
  - Nw: superedge weights considered; N: superedge weights discarded
- Probabilistic-GPQPS, in 3 variants, depending on the superedge weights considered:
  - P: no weights; Pa: average weight; Pe: expected weight

Assessment criteria:

- Clustering coefficient: relative error
- Community detection: relative error in modularity
- Top-ranked centrality: centrality rank comparison in terms of precision and recall
  - Precision and recall computed on g-set (resp. s-set), i.e., the set of vertices with centrality score no less than x_g (resp. x_s)
- Core decomposition: similar criterion to top-ranked centrality
  - Taking the inner-most core of the graph as a ground-truth set and the top-z inner-most cores computed via GPQPS

Results (summary of main findings)

Promising effectiveness overall

Obstacles for a more effective GPQPS:

- weighted graphs handled with non-weighted summary graphs
- handling directed graphs
- summaries overly sparse or not well-connected

Consistent gain in storage space achieved by any tested GPQPS method

Drastic speedup by Naïve-GPQPS

Speedup by probabilistic-GPQPS appreciable for large datasets or expensive queries

Results (summary of main findings)

Increasing summary size corresponds to an increase of effectiveness and a decrease of speedup

Naive-GPQPS vs. Probabilistic-GPQPS: no clear winner

No clear winner among weighted and unweighted variants of the various GPQPS methods

Conclusion

We introduce *general-purpose (approximate) query processing on summary graph* (GPQPS), a new tool to support scalable data-management workloads on graphs

- Our major goal in studying this problem is to set its stage, and stimulate and drive further research on it, by devising initial, basic methodologies

We devise algorithms for GPQPS

We set up an evaluation methodology that constitutes a benchmark testbed for this and future GPQPS studies

We perform extensive experiments according to the proposed evaluation methodology. Results attest promising results achieved by the proposed methods.

Reproducibility: data and code are available at https://github.com/fgullo/GPQPS

Thanks!

Questions?

Backup slides