Towards an efficient generic solver for fixed-cardinality optimization in graphs

Christian Komusiewicz and Frank Sommer

Philipps Universität Marburg

NII Shonan Meeting No. 144

Parameterized Graph Algorithms & Data Reduction: Theory Meets Practice

March 2019

Fixed-Cardinality Optimization

Input: A graph $G = (V, E)$, an objective function $\phi : \mathcal{G} \to \mathbb{Z}$, $k \in \mathbb{N}$.
Task: Find $S \subseteq V$ such that

- $|S| = k$, and
- $\phi(G[S])$ is maximum.

Fixed-Cardinality Optimization

Input: A graph $G = (V, E)$, an objective function $\phi : \mathcal{G} \to \mathbb{Z}$, $k \in \mathbb{N}$.
Task: Find $S \subseteq V$ such that

- $|S| = k$, and
- $\phi(G[S])$ is maximum.

Examples:

- $\phi(H) = |E(H)| \rightsquigarrow$ **Densest-k-Subgraph**

Fixed-Cardinality Optimization

Input: A graph $G = (V, E)$, an objective function $\phi : \mathcal{G} \to \mathbb{Z}$, $k \in \mathbb{N}$.
Task: Find $S \subseteq V$ such that

- $|S| = k$, and
- $\phi(G[S])$ is maximum.

Examples:

- $\phi(H) = |E(H)| \rightsquigarrow$ **Densest-k-Subgraph**
- $\phi(H) = \begin{cases} 1 & H \text{ is 2-regular and connected,} \\ 0 & \text{otherwise.} \end{cases}$

  $\rightsquigarrow$ **Induced k-Cycle**

Previous Work

FPT Algorithm for (Δ, k) where Δ is the maximum degree of G:

Theorem (K. & Sorge, Discr Appl Math 2015)

If $\phi(H) = -\infty$ for all nonconnected graphs H, and $\phi(H)$ can be evaluated in $T(k)$ time, then **Fixed-Cardinality Optimization** can be solved in $\mathcal{O}((e(\Delta - 1))^{k-1}(\Delta + k) \cdot n) \cdot T(k)$ time.

Previous Work

FPT Algorithm for (Δ, k) where Δ is the maximum degree of G:

Theorem (K. & Sorge, Discr Appl Math 2015)

If $\phi(H) = -\infty$ for all nonconnected graphs H, and $\phi(H)$ can be evaluated in $T(k)$ time, then **Fixed-Cardinality Optimization** can be solved in $\mathcal{O}((e(\Delta - 1))^{k-1}(\Delta + k) \cdot n) \cdot T(k)$ time.

Algorithm:

- Enumerate all connected induced subgraphs $G[S]$ of order k,
- evaluate $\phi(G[S])$ for each,
- output the best set S.

Previous Work

FPT Algorithm for (Δ, k) where Δ is the maximum degree of G:

Theorem (K. & Sorge, Discr Appl Math 2015)

If $\phi(H) = -\infty$ for all nonconnected graphs H, and $\phi(H)$ can be evaluated in $T(k)$ time, then **Fixed-Cardinality Optimization** can be solved in $\mathcal{O}((e(\Delta - 1))^{k-1}(\Delta + k) \cdot n) \cdot T(k)$ time.

Algorithm:

- Enumerate all connected induced subgraphs $G[S]$ of order k,
- evaluate $\phi(G[S])$ for each,
- output the best set S.

$\rightsquigarrow$ Implementation for **Connected Densest-k-Subgraph**

[K., Stahl & Sorge, SEA '15]

Previous Work

FPT Algorithm for (Δ, k) where Δ is the maximum degree of G:

Theorem (K. & Sorge, Discr Appl Math 2015)

If $\phi(H) = -\infty$ for all nonconnected graphs H, and $\phi(H)$ can be evaluated in $T(k)$ time, then **Fixed-Cardinality Optimization** can be solved in $\mathcal{O}((e(\Delta - 1))^{k-1}(\Delta + k) \cdot n) \cdot T(k)$ time.

Algorithm:

- Enumerate all connected induced subgraphs $G[S]$ of order k,
- evaluate $\phi(G[S])$ for each,
- output the best set S.

$\rightsquigarrow$ Implementation for **Connected Densest-k-Subgraph**

[K., Stahl & Sorge, SEA '15]

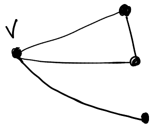
Aim: Implement & engineer general algorithm

Connected Subgraph Enumeration

General approach: starting from a single vertex, grow a connected subgraph by adding neighbors of current subgraph

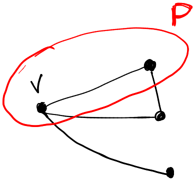
Connected Subgraph Enumeration

General approach: starting from a single vertex, grow a connected subgraph by adding neighbors of current subgraph



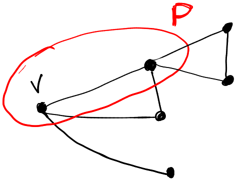
Connected Subgraph Enumeration

General approach: starting from a single vertex, grow a connected subgraph by adding neighbors of current subgraph



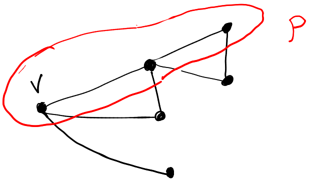
Connected Subgraph Enumeration

General approach: starting from a single vertex, grow a connected subgraph by adding neighbors of current subgraph



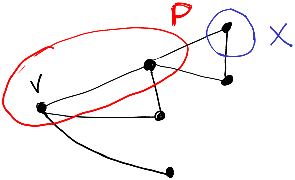
Connected Subgraph Enumeration

General approach: starting from a single vertex, grow a connected subgraph by adding neighbors of current subgraph



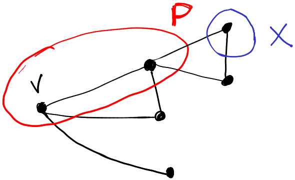
Connected Subgraph Enumeration

General approach: starting from a single vertex, grow a connected subgraph by adding neighbors of current subgraph



Connected Subgraph Enumeration

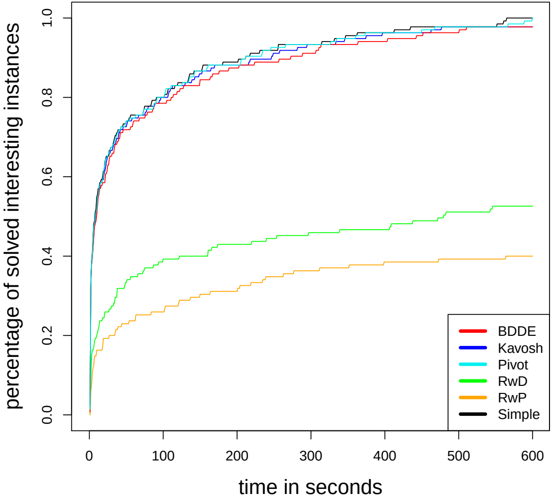
General approach: starting from a single vertex, grow a connected subgraph by adding neighbors of current subgraph



Variants:

- Simple: Try each neighbor u of P [Wernicke, WABI '15]
- Kavosh: Try each $Q \subseteq N(P)$ such that $|Q| \leq k - |P|$ [Kashani et al., BMC Bioinformatics '09]
- Pivot: $P = \{p_1, p_2, \ldots, p_{|P|}\}$, first add neighbors of p_1, then neighbors of p_2, ... [K. & Sorge, IPEC '12]

Connected Subgraph Enumeration: Experiments



Theoretical guarantee: Simple and Pivot have worst-case delay $\mathcal{O}(k^2\Delta)$ [K. & Sommer, SOFSEM '18]

Experimental Setup

Test problems:

- **Dense:** Densest-k-Subgraph, Max-Min-Degree-Subgraph
- **Sparse:** Min-Max-Degree-Subgraph, Acyclic Subgraph, Triangle-Free Subgraph, Max-Diameter Subgraph
- **Degree-Constraint:** 2-Regular Subgraph, $(2,5)$-Degree Constrained Subgraph

Data:

- 30 sparse real-world networks, $n \in [27, 541\,000]$, $m \in [78, 16 \cdot 10^6]$
- 20 $G_{n,p}$ graphs, $n \in [100, 1000]$, $p \in \{0.1, 0.2\}$
- $k \in [3, 10]$

Implementation: Python + igraph

Timeout: 60s per instance

Implementation of Objective Functions

```
def evaluate(graph):
    global problem
    global parameters
    if problem == Fco.DENSEST: # densest k-subgraph
        return graph.ecount()
    elif problem == Fco.MINDEG: # maximize min-degree
        min_deg = graph.vcount()
        for i in range(graph.vcount()):
            min_deg = min(graph.degree(i),min_deg)
        return min_deg
    elif problem == Fco.MINMAXDEG: # minimize max-degree
        return -graph.maxdegree()
    elif problem == Fco.ACYCLIC: # find an acyclic graph
        if graph.ecount() == graph.vcount() - 1:
            return 1
        else:
            return 0
    ...
```

Pruning Rules and Problem Properties

Approach: For each P, compute whether P may be extended to an optimal solution. If not, return to parent node of the search tree.

Problem: ϕ is unknown

Pruning Rules and Problem Properties

Approach: For each P, compute whether P may be extended to an optimal solution. If not, return to parent node of the search tree.

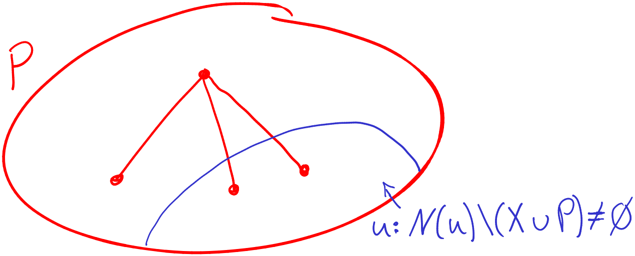
Problem: ϕ is unknown

Idea: use properties that are shared by many natural ϕ's.

- vertex-addition-bounded ϕ: by adding one vertex to H, the value of ϕ may increase by at most x
- edge-monotonicity: adding an edge to H does not decrease the value of $\phi(H)$
- edge-removal-monotonicity: removing an edge from H does not decrease the value of $\phi(H)$ (if we maintain connectivity)

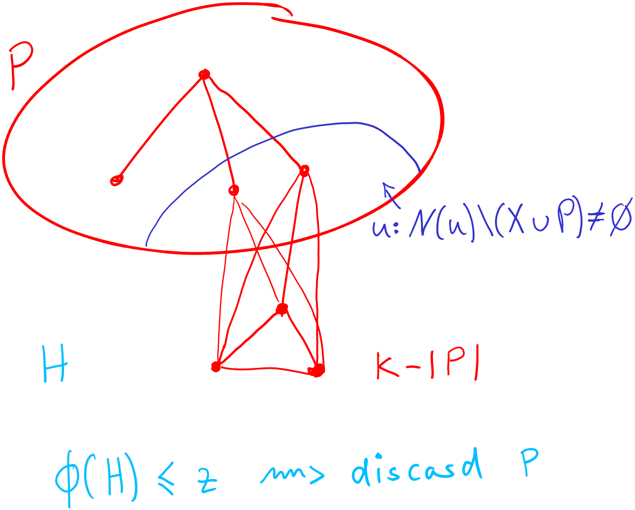
Clique Join Rule

z: current upper bound



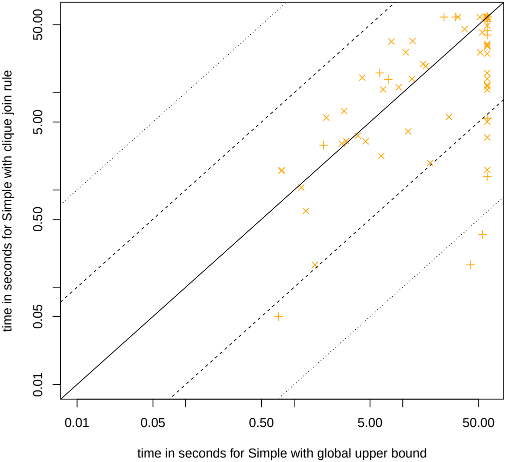
Clique Join Rule

z: current upper bound



Clique Join Rule

z: current upper bound



Vertex Addition Bounds

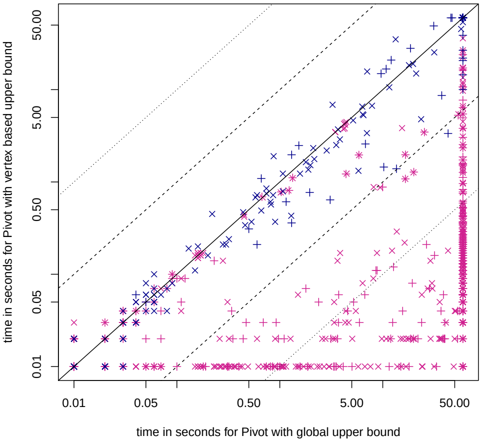
Assumption: ϕ is addition-bounded by x

Rule: If $\phi(G[P]) + (|P| - k)x \leq z$, then discard P.

Vertex Addition Bounds

Assumption: ϕ is addition-bounded by x

Rule: If $\phi(G[P]) + (|P| - k)x \leq z$, then discard P.



Vertex Addition Bounds and Objective Functions

Acyclic / Triangle-free Subgraph:

$$\phi(H) = \begin{cases} 1 & H \text{ is acyclic / triangle-free,} \\ 0 & \text{otherwise.} \end{cases} \qquad x = 0$$

Min-Max-Degree Subgraph:

$$\phi(H) = -\max_{v \in V(H)} \{\deg(v)\}, \qquad x = 0$$

Max-Diameter Subgraph:

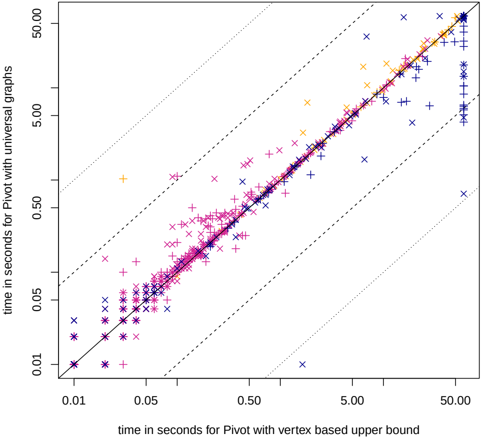
$$\phi(H) = \operatorname{diam}(H), \qquad x = 1$$

Universal Graph Rule

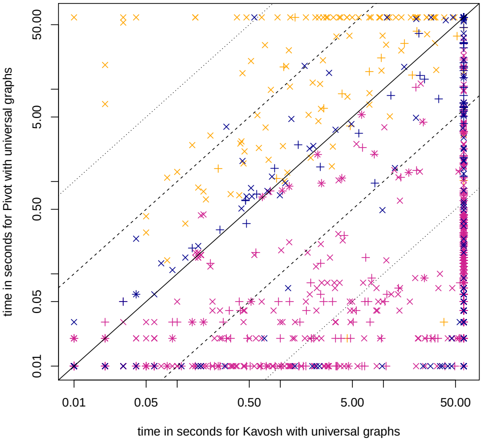
Rule: For each graph H of order $k - |P|$, try all possibilities to add edges between H and $G[P]$. If $\phi(G[H']) \leq z$ for all resulting H', then discard P.

Universal Graph Rule

Rule: For each graph H of order $k - |P|$, try all possibilities to add edges between H and $G[P]$. If $\phi(G[H']) \leq z$ for all resulting H', then discard P.



Kavosh vs. Pivot



ILP & Tractable k

Name	n	m	*Kavosh*	*Pivot*	*Simple*	*ILP*
ucidata-zachary	34	78	8	**9**	8	8
dolphins	62	159	9	9	8	**10**
adjnoun_adjacency	112	425	6	7	6	**10**
inf-USAir97	332	2 126	4	5	4	**7**
bio-celegans	453	2 025	5	6	5	**7**
soc-wiki-Vote	889	2 914	**6**	**6**	**6**	5
inf-euroroad	1 174	1 417	**8**	**8**	7	2
ca-CSphd	1 882	1 740	**6**	**6**	5	2
inf-openflights	2 939	15 677	4	**6**	5	2
inf-power	4 941	6 594	**7**	**7**	**7**	2
bio-dmela	7 393	25 569	**5**	4	4	2
ca-AstroPh	17 903	196 972	**4**	**4**	**4**	2
coAuthorsCiteseer	227 320	814 134	**5**	4	4	2
soc-twitter-follows	404 719	713 319	**3**	2	2	2
coPapersDBLP	540 486	15 245 729	**3**	**3**	**3**	2

Outlook

To-Do List:

- Data reduction: twin removal, domination rules
- Greedy initialization of upper bounds, searching for cliques for edge-monotone properties
- Further specialized data reduction rules: restricted versions of universal graph rule for edge-removal-monotonicity

Theory:

- Further improvement of the enumeration delay
- Generic tractability results for smaller parameters than Δ
- Improved running time bounds for (Δ, k)

Practice:

- Further pruning rules and objective function properties
- Framework for user-specified upper bounds
- Extension to weighted, directed, and colored graph variants