

On Multivariate Algorithmics and Structure Detection

Rolf Niedermeier

TU Berlin
Institut für Softwaretechnik und Theoretische Informatik
Algorithmics and Complexity Theory
<http://www.akt.tu-berlin.de>

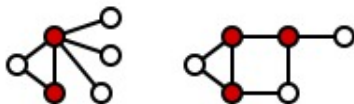
Structure in Hard Combinatorial Problems, Vienna, May 2013

Motivating (Standard) Example I

Established efficiency race for NP-hard **Vertex Cover** problem:

Input: An undirected graph $G = (V, E)$ and a nonnegative integer k .

Task: Find a subset of vertices $C \subseteq V$ with k or fewer vertices such that each edge in E has at least one of its endpoints in C .



Currently best upper bound: $O(1.274^k + k|V|)$ time.
(Chen, Kanj, Xia: Theor. Comput. Sci. 2010)

Grain of salt: In many applications, the parameter k is not small and grows with the graph size.

Motivating Example Vertex Cover

Input: An undirected graph $G = (V, E)$ and a nonnegative integer k .

Task: Find a subset of vertices $C \subseteq V$ with k or fewer vertices such that each edge in E has at least one of its endpoints in C .

New (above guarantee) parameter for Vertex Cover: $k' := k - LP$, where LP denotes the value of the linear programming relaxation of the standard ILP for Vertex Cover...

(Narayanaswamy et al., STACS 2012)

So, LP bound is a guaranteed lower bound for solution size!

Clearly, k' is “stronger” than k .

Central result: Vertex Cover solvable in $2.62^{k'} \cdot (|V| + |E|)^{O(1)}$ time.

↪ **Our general theme:** Are there structures (that is, parameterizations) that can be exploited for deriving “efficient” solutions for NP-hard problems?

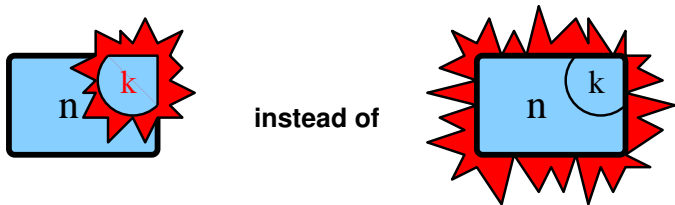
Parameterized Algorithmics in a Nutshell

NP-hard problem X : Input size n and problem parameter k .

If there is an algorithm solving X in time

$$f(k) \cdot n^{O(1)},$$

then X is called **fixed-parameter tractable (FPT)**:



Completeness program developed by Downey and Fellows (1999).

$$\text{FPT} \subseteq \overbrace{W[1] \subseteq W[2] \subseteq \dots \subseteq W[P]}^{\text{Presumably fixed-parameter intractable}} \subseteq \text{XP}$$

Parameterized Complexity Hierarchy

FPT vs W[1]-hard vs para-NP-hard:

- Vertex Cover, parameterized by solution size is FPT;
- Clique parameterized by solution size is W[1]-hard (but in XP);
- k -Coloring is para-NP-hard (and thus not in XP unless P=NP) (because it is NP-hard for $k = 3$ colors (that is, constant parameter value)).

“Function battle” concerning allowed running time:

$$\text{FPT: } f(k) \cdot n^{O(1)} \quad \text{vs} \quad \text{XP: } f(k) \cdot n^{g(k)}$$

Assumption: $\text{FPT} \neq \text{W}[1]$

For instance, if $\text{W}[1] = \text{FPT}$ then 3-SAT for a Boolean formula F with n variables can be solved in $2^{o(n)} \cdot |F|^{O(1)}$ time.

The “Art” of Parameter Identification

Central question: How to find relevant parameterizations?

Central fact: One problem may have a large number of different (relevant) parameterizations, that is, structures to exploit...

Consequence: We achieve a more fine-grained but also more complicated picture of the computational complexity of problems.

↪ Parameterized algorithmics goes multivariate...

Note: Parameterizations typically are of “structural nature”, modelling properties that input instances may have...

Basic philosophy: Different parameterizations allow for different views, resulting in a “holistic” approach to complexity analysis.

Revisiting hardness proofs, **deconstruct intractability!**

Simple Examples for Problem Parameterization

Leitmotif: Achieve a better understanding / more complete picture of a problem's computational complexity by means of a refined running time and complexity analysis employing problem-specific parameters.

Natural parameterizations include:

- solution size (“pay for what you get”);
- special properties of input instances (e.g. (maximum) vertex degree or treewidth of graphs or...);
- distance measures (e.g. allowed (solution) error);
- ...

A Theoretical Way of Spotting Parameters

Call parameter k_1 **stronger than** parameter k_2 if there is a constant c such that $k_1 \leq c \cdot k_2$ for all inputs, and there is no constant d such that $k_2 \leq d \cdot k_1$ for all inputs.

(Analogously: k_2 is **weaker than** k_1).

Examples:

- Average vertex degree of a graph is stronger than maximum vertex degree.
- Treewidth is a stronger parameter than vertex cover number of a graph.
- Also: Single parameter k_1 is stronger than combined parameter $k_1 + k_2$ whatever k_2 is.

Goals for Stronger and Weaker Parameterizations

Primary goals:

- 1 Whenever a problem is fixed-parameter tractable with respect to a parameter k_1 , then try to also show fixed-parameter tractability for a **stronger** parameter k_2 .
- 2 If a problem is $W[1]$ -hard with respect to a parameter k_1 , then try to show fixed-parameter tractability for a **weaker** parameter k_2 .

Secondary goals:

- Can similar upper bounds be achieved for stronger parameters?
- Provide a “complete” map of a problem’s (parameterized) computational complexity with respect to various parameterizations (partially) ordered by their respective “strength”.

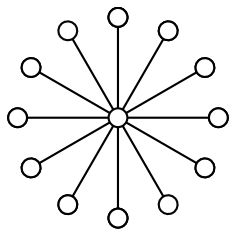
The Multivariate Landscape of Finding 2-Clubs

NP-hard **s-Club** problem (occurring in the analysis of social and biological networks):

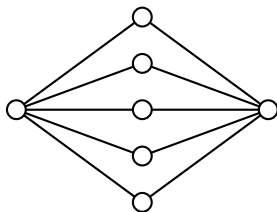
Input A graph $G = (V, E)$ and an integer k .

Question Is there a vertex set $V' \subseteq V$ of size at least k such that $G[V']$ has diameter at most s ?

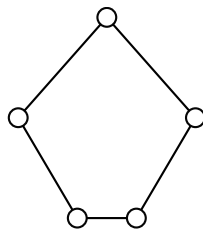
- 1-Club is equivalent to Clique.
- We focus on 2-Club:



star



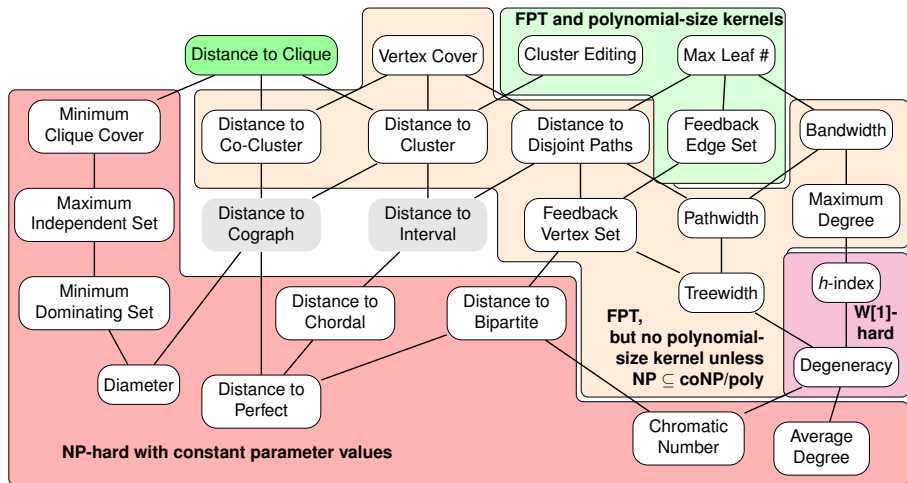
diamond



C_5

Navigating Through Parameter Space: 2-Clubs


[Hartung, Komusiewicz, Nichterlein, IPEC 2012 + SOFSEM 2013].



A More Pragmatic Way of Spotting Parameters

A simple way to spot interesting parameterizations (structure) in real-world graph problems: Measure “all” possible parameters...:

Use tool Graphana for data-driven parameterization:

Algorithmik & Komplexitätstheorie **Graphana Web Interface** 

Configuration

Graph

Insert graph as DIMACS or dot:

```
v1 v2
v1 v3
v2 v5
v2 v4
v3 v5
v3 v4
v4 v6
```

Alternative: upload a DIMACS or dot file:

Algorithms

Graph parameters:

- Average degree / max degree
- Connected components
- Vertex cover size (upper bound, greedy)
- Vertex cover size (upper bound, 2-approx)
- h-index
- Treewidth (upper bound)
- Treewidth (lower bound)
- Degeneracy
- Feedback edge set size
- Feedback vertex set size (upper bound)
- Diameter
- Cluster vertex deletion size (upper bound)

Distributions (in CSV format):

- Degrees
- Edge connectivity
- Distances

Results

Vertices / Edges:	6 / 8
Average degree / max degree:	2.6666667 / 3
Connected components:	1
Vertex cover size (upper bound, greedy):	3
h-index:	3
Degeneracy:	2
Feedback edge set size:	3
Feedback vertex set size (upper bound):	2

www.akt.tu-berlin.de/menue/software

Outline of the Remaining Talk

We discuss four recent examples where structure detection and parameterized complexity analysis were instrumental:

- Network analysis:
 - biological (Highly Connected Deletion problem);
 - social (Graph Anonymization problem);
- Combinatorial feature selection (Distinct Vectors problem).
- Computational social choice (Lobbying problem).

Case Study: Highly Connected Deletion

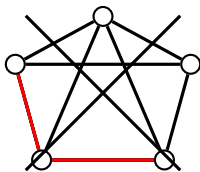
Typical graph clustering situation:

Task: Partition a graph into clusters such that

- each cluster is **dense** and
- there are few edges between clusters.

↪ **Definition:** [Hartuv & Shamir, IPL '00]

A graph with n vertices is **highly connected** if more than $n/2$ edges need to be deleted to make it disconnected.



Properties:

- diameter two;
- each vertex has degree $\geq \lfloor n/2 \rfloor$.

A MinCut Heuristic for Highly-Connected Clustering

Task: Partition the network into clusters such that

- each cluster is highly connected and
- there are few edges between clusters.

Challenge: How to find highly connected clusters?

Min-Cut Algorithm:

[Hartuv & Shamir, IPL 2000]

Input: $G = (V, E)$

- 1 $(A, B) = \text{min-cut}(G)$
- 2 **if** (A, B) has $> |V|/2$ edges : **output** V
- 3 **else:** recurse on $G[A]$ and $G[B]$

Biological applications:

- Clustering cDNA fingerprints
- Complex identification in protein–interaction networks
- Hierarchical clustering of protein sequences
- Clustering regulatory RNA structures

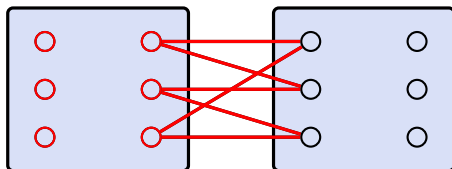
Edge Coverage


Task: Partition the network into clusters such that

- each cluster is highly connected and
- the number of **edges between clusters** is minimal.

Does the MinCut heuristic achieve **the second goal**?

↪ Comparison with optimal solution...



 \equiv clique

↪ MinCut heuristic may delete $\Theta(\text{OPT}^2)$ many edges!

↪ New goal: find optimal clustering

Complexity of Highly Connected Deletion

Highly Connected Deletion

Input: An undirected graph.

Task: Delete a minimum number of edges such that each remaining connected component is highly connected.

Theorem: Highly Connected Deletion is NP-hard even on 4-regular graphs.

Theorem: If the Exponential Time Hypothesis (ETH) is true, then **Highly Connected Deletion** cannot be solved within $2^{o(m)}$ poly(n) time or $2^{o(n)}$ poly(n) time.

m := number of edges

n := number of vertices

Data Reduction Rules

Idea 1: Find vertex sets that are **inseparable** because any cut of this set has size $> k$

\rightsquigarrow **Too-Connected-Rule:** If G contains an inseparable vertex set S of size at least $2k$, then do the following. If $G[S]$ is not highly connected, return “no”. Otherwise, remove S from G and adapt k correspondingly.

Idea 2: Find highly connected clusters that are large compared to the number of “outgoing” edges

$\rightsquigarrow D(S) :=$ edge cut between S and $V \setminus S$

Small-Cut-Rule: If G contains a vertex set S such that

- $|S| \geq 4$,
- $G[S]$ is highly connected, and
- $|D(S)| \leq 0.3 \cdot \sqrt{|S|}$,

then remove S from G .

Theorem: Highly Connected Deletion admits polynomial-time data reduction to an equivalent instance with $\leq 10 \cdot k^{1.5}$ vertices.

FPT Algorithm and Further Data Reduction

Combination of branching, data reduction, and dynamic programming \rightsquigarrow

Theorem: Highly Connected Deletion can be solved in $O(3^{4k} \cdot k^2 + n^2 mk \cdot \log n)$ time.

Lemma: Let G be a highly connected graph. If two vertices in G are adjacent, they have at least one common neighbor.

\rightsquigarrow **Reduction rule:** If there are two vertices that are connected by an edge but have no common neighbors, then delete the edge.

Highly Connected Deletion: PPI Experiments

	n	m	Δk	Δk [%]	n'	m'
<i>C. elegans</i> phys.	157	153	100	92.6	11	38
<i>C. elegans</i> all	3613	6828	5204	80.1	373	1562
<i>M. musculus</i> phys.	4146	7097	5659	85.3	426	1339
<i>M. musculus</i> all	5252	9640	7609	84.8	595	1893
<i>A. thaliana</i> phys.	1872	2828	2057	83.1	187	619
<i>A. thaliana</i> all	5704	12627	8797	79.5	866	3323

n' , m' : size of largest connected component after data reduction

	min-cut without DR			min-cut with DR			column generation		
	k	s	t	k	s	t	k	s	t
CE-p	111	136	0.01	108	133	0.01	108	133	0.06
CE-a	6714	3589	86.46	6630	3521	6.36	6499	3436	2088.35
MM-p	7004	4116	126.30	6882	4003	7.42	6638	3845	898.13
MM-a	9563	5227	267.63	9336	5044	17.84	8978	4812	3858.62
AT-p	2671	1796	5.82	2567	1723	0.68	2476	1675	60.34
AT-a	12096	5559	434.52	11590	5213	32.09	11069	4944	34121.23

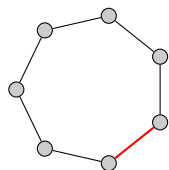
s : number of unclustered vertices; t : running time in seconds

Case Study: Graph Anonymization

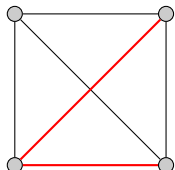
Degree Anonymization

Input: An undirected graph $G = (V, E)$ and two positive integers k and s .

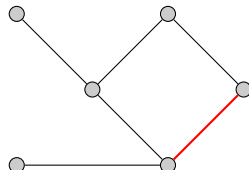
Question: Is there an edge set E' over V with $|E'| \leq s$ such that $G' = (V, E \cup E')$ is k -anonymous, that is, for every vertex $v \in V$ there are at least $k - 1$ other vertices in G' having the same degree?



2-anonymous graph

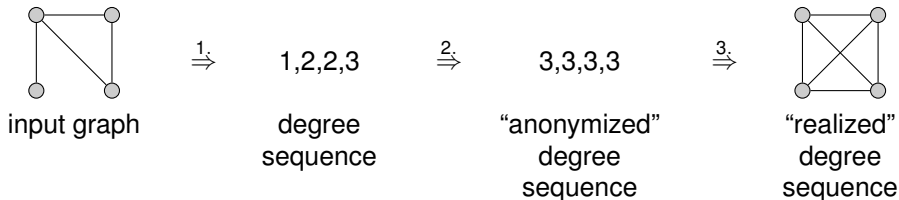


1-anonymous graph



1-anonymous graph

An Anon. Heuristic of Liu and Terzi [SIGMOD 2008]



Step 1: Sorting the degrees.

Step 2: Standard dynamic programming
(running time $O(n \cdot s \cdot k \cdot \Delta) = O(n^4)$).

Step 3: (Due to graph structure not always possible!)
If there exists a “realization”, then it can be constructed in polynomial time ($\rightsquigarrow f$ -factors).

The Third Step of the Liu-Terzi-Heuristic

Lemma

If the solution (edge set) found in the dynamic programming is “large” ($s > \Delta^4$), then there is always a realization of the anonymized degree sequence that is a supergraph of the input graph. This realization can be found in polynomial time.

Consequence: win-win situation. Either

- the problem is polynomial-time solvable or
- the solution is “small” ($\leq \Delta^4$).

Case Study: Distinct Vectors

Genre of *dimension reduction*: Represent objects in a lower-dimensional space that still “explains” their properties.

Distinct Vectors:

Input: A multiset $S = \{x_1, \dots, x_n\} \subseteq \Sigma^d$ containing n distinct points and $k \geq 1$.

Question: $\exists K \subseteq \{1, \dots, d\}$ with $|K| \leq k$ such that all points in $S|_K$ are still distinct?

Also known as finding unique keys in databases or the Minimal Reduct problem in *rough set theory*.

	1	2	3	4	5	6	7	8	9	10
x_1	0	1	1	1	1	0	1	1	0	0
x_2	0	0	0	1	1	0	1	1	0	0
x_3	0	1	0	0	0	0	1	1	1	1
x_4	1	0	1	1	0	0	1	0	1	0
x_5	1	1	0	1	1	0	0	0	1	0

Distinct Vectors

Let h be the maximum Hamming distance between two input (row) vectors.

Intuition: Data sets containing “similar” points are easy to solve?!

↪ Complexity dichotomy...

Theorem

Distinct Vectors restricted to binary input alphabet is

- 1 solvable in polynomial (cubic) time if $h \leq 3$ and
- 2 NP-hard for $h \geq 4$,
- 3 W[1]-hard parameterized by the number of dimensions to delete.

Algorithmic result follows from a search tree that “does not branch too much” due to “regular structure” (sort of staircase structure) if $h \leq 3$, while with $h \geq 4$ NP-hardness follows by a reduction from Distance-3 Independent Set.

Theorem

Distinct Vectors for arbitrary alphabets is W[2]-hard parameterized by the number of dimensions to keep.

Follows from a parameterized reduction from Hitting Set.

Case Study: Optimal Lobbying

Multiple Referenda

Each voter can accept or reject each of several issues.

Election: $A \in \{0, 1\}^{n \times m}$.

Result: $z \in \{0, 1\}^m$.

Lobbyist's goal: 1^m .

	issue 1	issue 2	issue 3
voter 1	0	0	1
voter 2	1	0	1
voter 3	0	1	0
voter 4	1	0	0
voter 5	0	1	0
result	0	0	0
lobbyist	1	1	1

Optimal Lobbying

Input: $A \in \{0, 1\}^{n \times m}$ and $k \leq n$.

Question: Can we change at most k rows of A such that each column has more 1s than 0s?

[Christian, Fellows, Rosamond, and Slinko, Review of Economic Design'07]

Parameters in Optimal Lobbying

	issue 1	issue 2	issue 3
voter 1	0	0	1
voter 2	1	0	1
voter 3	0	1	0
voter 4	1	0	0
voter 5	0	1	0
result	0	0	0
target	1	1	1
gap	1	1	1

Interesting parameters

- Number n of voters (rows)
- Number m of issues (columns)
- Max. number s of 1s per row
- Max. number t of 0s per row
- Max. gap value g in a column

$$k = 2 \quad s = 2 \quad t = 2 \quad g = 1$$

	m	k	g	s
m	ILP-FPT	FPT	FPT	ILP-FPT
k		W[2]-c.	W[2]-c.	FPT
g			W[2]-h. LOGSNP-c.	FPT
s				$s \geq 3$: NP-h. $s \leq 2$: P

A Simple Greedy Algorithm for Lobbying

- 1: Compute the gap values of all columns;
- 2: **while** \exists column with positive gap value **do**
- 3: Compute the **costs** of each row i

$$\text{cost}(i) = \frac{1}{\sum_{j=1}^m (1 - A_{i,j}) \cdot m^{g_j - 1}}$$

- 4: Modify a row with **minimum** cost
- 5: Update gap values and delete satisfied columns

$A =$	cost
$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$	$(1 \cdot 4^1 + 2 \cdot 4^0) - 1 = 1/6$ $(1 \cdot 4^1 + 1 \cdot 4^0) - 1 = 1/5$ 1/2 1/5 1/2 1/5 1 1/2 1 1/2 0

Properties of Greedy Algorithm for Lobbying

- Runs in quadratic time if the number of columns is constant.
- Provides optimal results for up to four columns; there is a counter-example for five columns.
- Provides a logarithmic factor approximation.
- On synthetic random data with up to 300 columns and 1000 rows the running times were far below one second and the delivered solutions with very few exceptions were almost always optimal...

Remark: While Optimal Lobbying is easily shown to be $W[2]$ -hard with respect to the gap parameter g , it could also be shown to be LOGSNP-complete for $g = 1$ (the problem is trivial for $g = 0$), using a reduction from Rich Hypergraph Cover.

Discussion and Outlook

- Every problem accompanied with a natural parameter space (that is, structure) to navigate through.
↪ Relevance of investigating the combinatorial relationships between different parameters and their combinations.
- Structure analysis connects very well with multivariate algorithmics.
- Potential practical relevance when taking into account structure occurring in real-world data.
↪ Algorithm engineering through parameter identification in real-world instances.
- Detecting “hidden parameters” may help in better understanding and exploiting the power of heuristics.
- Potential drawback from a practical point of view: all our studies still rely on worst-case analysis.
- Once a whole suite of (parameterized) algorithms is available, choosing the “right” one depending on the current input data becomes more relevant...

References

- Related survey: C. Komusiewicz, R. Niedermeier: New Races in Parameterized Algorithmics. MFCS 2012.
- Highly Connected Deletion: F. Hüffner, C. Komusiewicz, A. Liebtrau, R. Niedermeier. Partitioning Biological Networks into Highly Connected Clusters with Maximum Edge Coverage. ISBRA 2013.
- Graph Anonymization: S. Hartung, A. Nichterlein, R. Niedermeier, O. Suchý. A Refined Complexity Analysis of Degree Anonymization on Graphs. ICALP 2013.
- Distinct Vectors: V. Froese, R. van Bevern, R. Niedermeier, and M. Sorge. A Refined Complexity Analysis of Combinatorial Feature Selection Problems. Manuscript 2013.
- Lobbying: R. Brederbeck, J. Chen, S. Hartung, S. Kratsch, R. Niedermeier, O. Suchý, G. J. Woeginger: A Multivariate Complexity Analysis of Lobbying in Multiple Referenda. Manuscript 2013 based on AAI 2012 paper.