10. Iterative Compression

COMP6741: Parameterized and Exact Computation

Serge Gaspers[1,2]

[1] School of Computer Science and Engineering, UNSW Australia
[2] Optimisation Resarch Group, NICTA

Semester 2, 2015

Outline

Outline

Iterative Compression

For a minimization problem:

- **Compression step:** Given a solution of size $k+1$, compress it to a solution of size k or prove that there is no solution of size k
- **Iteration step:** Incrementally build a solution to the given instance by deriving solutions for larger and larger subinstances

Iterative Compression

For a minimization problem:

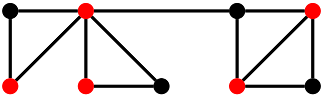
- **Compression step:** Given a solution of size $k+1$, compress it to a solution of size k or prove that there is no solution of size k
- **Iteration step:** Incrementally build a solution to the given instance by deriving solutions for larger and larger subinstances

- Seen a lot of success in Parameterized Complexity
- Examples: best known fixed parameter algorithms for (Directed) Feedback Vertex Set, Edge Bipartization, Almost 2-SAT, ...

Example: Vertex Cover

A vertex cover in a graph $G = (V, E)$ is a subset of its vertices $S \subseteq V$ such that every edge of G has at least one endpoint in S.

Vertex Cover	
Input:	A graph $G = (V, E)$ and an integer k
Parameter:	k
Question:	Does G have a vertex cover of size k?



We will design a (slow) iterative compression algorithm for Vertex Cover to illustrate the technique.

Vertex Cover: Compression Step

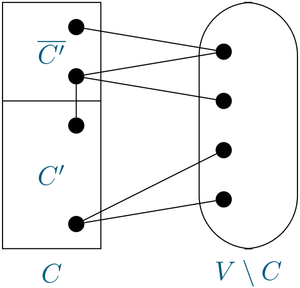
Comp-VC

Input:	graph $G = (V, E)$, integer k, vertex cover C of size $k + 1$ of G
Output:	a vertex cover C^* of size $\leq k$ of G if one exists

Vertex Cover: Compression Step

Comp-VC

Input:	graph $G = (V, E)$, integer k, vertex cover C of size $k+1$ of G
Output:	a vertex cover C^* of size $\leq k$ of G if one exists



- Go over all partitions $(C', \overline{C'})$ of C
- $C^* = C' \cup N(\overline{C'})$
- If $\overline{C'}$ is an independent set and $|C^*| \leq k$ then return C^*

VERTEX COVER: Iteration Step

Use algorithm for COMP-VC to solve VERTEX COVER.

Vertex Cover: Iteration Step

Use algorithm for Comp-VC to solve Vertex Cover.

- Order vertices: $V = \{v_1, v_2, \ldots, v_n\}$
- Define $G_i = G[\{v_1, v_2, \ldots, v_i\}]$
- $C_0 = \emptyset$
- For $i = 1..n$, find a vertex cover C_i of size $\leq k$ of G_i using the algorithm for Comp-VC with input G_i and $C_{i-1} \cup \{v_i\}$. If G_i has no vertex cover of size $\leq k$, then G has no vertex cover of size $\leq k$.

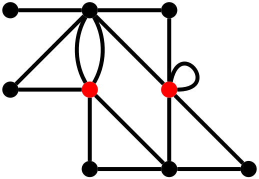
Final running time: $O^*(2^k)$

Outline

Feedback Vertex Set

A feedback vertex set of a multigraph $G = (V, E)$ is a set of vertices $S \subseteq V$ such that $G - S$ is acyclic.

FEEDBACK VERTEX SET (FVS)	
Input:	Multigraph $G = (V, E)$, integer k
Parameter:	k
Question:	Does G have a feedback vertex set of size at most k?



Note: We already saw an $O^*((3k)^k)$ time algorithm for FVS.
We will now aim for a $O^*(c^k)$ time algorithm, with $c \in O(1)$.

Compression Problem

Comp-FVS

Input:	graph $G = (V, E)$, integer k, feedback vertex set S of size $k + 1$ of G
Output:	a feedback vertex set S^* of size $\leq k$ of G if one exists

Iteration step

- Order vertices: $V = \{v_1, v_2, \dots, v_n\}$
- Define $G_i = G[\{v_1, v_2, \dots, v_i\}]$
- $S_0 = \emptyset$
- For $i = 1..n$, find a feedback vertex set S_i of size $\le k$ of G_i using the algorithm for $\textsc{Comp-FVS}$ with input G_i and $S_{i-1} \cup \{v_i\}$. If G_i has no feedback vertex set of size $\le k$, then G has no feedback vertex set of size $\le k$.

Suppose $\textsc{Comp-FVS}$ can be solved in $O^*(c^k)$ time.
Then, using this iteration, FVS can be solved in $O^*(c^k)$ time.

Compression step

To solve $\textsc{Comp-FVS}$, go through all partitions $(S', \overline{S'})$ of S. For each of them, we will want to find a feedback vertex set S^* of G with $|S^*| < |S|$ and $S' \subseteq S^* \subseteq V \setminus \overline{S'}$ if one exists.

Compression step

To solve Comp-FVS, go through all partitions $(S', \overline{S'})$ of S. For each of them, we will want to find a feedback vertex set S^* of G with $|S^*| < |S|$ and $S' \subseteq S^* \subseteq V \setminus \overline{S'}$ if one exists.
Equivalently, find a feedback vertex set S'' of $G - S'$ with $|S''| < |\overline{S'}|$ and $S'' \cap \overline{S'} = \emptyset$.

Compression step

To solve COMP-FVS, go through all partitions $(S', \overline{S'})$ of S. For each of them, we will want to find a feedback vertex set S^* of G with $|S^*| < |S|$ and $S' \subseteq S^* \subseteq V \setminus \overline{S'}$ if one exists.
Equivalently, find a feedback vertex set S'' of $G - S'$ with $|S''| < |\overline{S'}|$ and $S'' \cap \overline{S'} = \emptyset$.
We arrive at the following problem:

DISJOINT-FVS

Input:	graph $G = (V, E)$, integer k, feedback vertex set S of size $k + 1$ of G
Output:	a feedback vertex set S^* of G with $|S^*| \leq k$ and $S^* \cap S = \emptyset$, if one exists

Compression step

To solve COMP-FVS, go through all partitions $(S', \overline{S'})$ of S. For each of them, we will want to find a feedback vertex set S^* of G with $|S^*| < |S|$ and $S' \subseteq S^* \subseteq V \setminus \overline{S'}$ if one exists.
Equivalently, find a feedback vertex set S'' of $G - S'$ with $|S''| < |\overline{S'}|$ and $S'' \cap \overline{S'} = \emptyset$.
We arrive at the following problem:

DISJOINT-FVS	
Input:	graph $G = (V, E)$, integer k, feedback vertex set S of size $k + 1$ of G
Output:	a feedback vertex set S^* of G with $|S^*| \leq k$ and $S^* \cap S = \emptyset$, if one exists

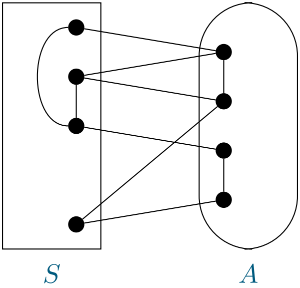
If DISJOINT-FVS can be solved in $O^*(d^k)$ time, then COMP-FVS can be solved in

$$O^*\left(\sum_{i=0}^{k+1}\binom{k+1}{i}d^i\right) \subseteq O^*((d+1)^k) \text{ time.}$$

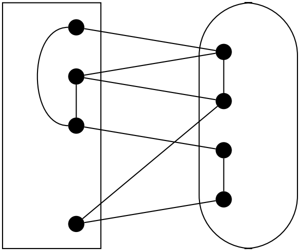
Algorithm for Disjoint-FVS

Disjoint-FVS	
Input:	graph $G = (V, E)$, integer k, feedback vertex set S of size $k+1$ of G
Output:	a feedback vertex set S^* of G with $\|S^*\| \leq k$ and $S^* \cap S = \emptyset$, if one exists

Denote $A := V \setminus S$.



Simplification rules for DISJOINT-FVS



Start with $S^* = \emptyset$.

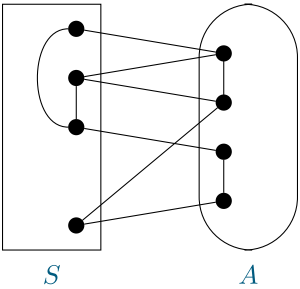
(cycle-in-S)

If $G[S]$ is not acyclic, then return NO.

(budget-exceeded)

If $k < 0$, then return NO.

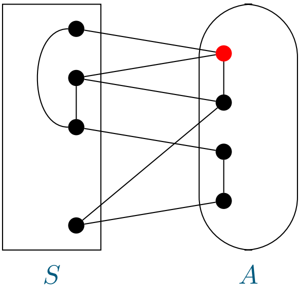
Simplification rules for DISJOINT-FVS



(finished)

If $G - S^*$ is acyclic, then return S^*.

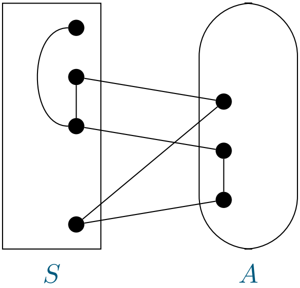
Simplification rules for DISJOINT-FVS



(creates-cycle)

If $\exists v \in A$ such that $G[S \cup \{v\}]$ is not acyclic, then add v to S^* and remove v from G.

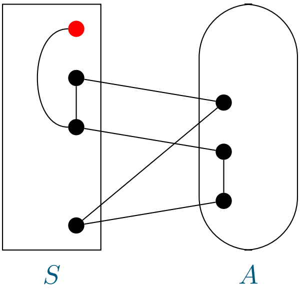
Simplification rules for DISJOINT-FVS



(creates-cycle)

If $\exists v \in A$ such that $G[S \cup \{v\}]$ is not acyclic, then add v to S^* and remove v from G.

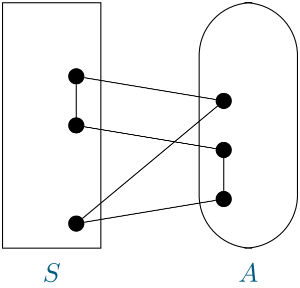
Simplification rules for DISJOINT-FVS



(Degree-(≤ 1))

If $\exists v \in V$ with $d_G(v) \leq 1$, then remove v from G.

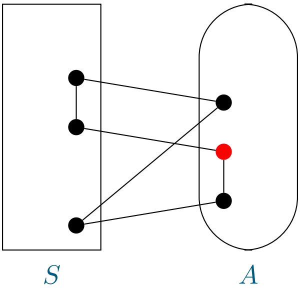
Simplification rules for DISJOINT-FVS



(Degree-(≤ 1))

If $\exists v \in V$ with $d_G(v) \leq 1$, then remove v from G.

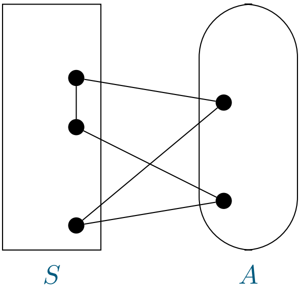
Simplification rules for Disjoint-FVS



(Degree-2)

If $\exists v \in V$ with $d_G(v) = 2$ and at least one neighbor of v is in A, then add an edge between the neighbors of v (even if there was already an edge) and remove v from G.

Simplification rules for Disjoint-FVS



(Degree-2)

If $\exists v \in V$ with $d_G(v) = 2$ and at least one neighbor of v is in A, then add an edge between the neighbors of v (even if there was already an edge) and remove v from G.

Branching rule for Disjoint-FVS

Select a vertex $v \in A$ with at least 2 neighbors in S.
Such a vertex exists if no simplification rule applies (for example, we can take a leaf in $G[A]$).
Branch into two subproblems:

- $v \in S^*$: add v to S^*, remove v from G, and decrease k by 1
- $v \notin S^*$: add v to S

Exercise: Running time

- Prove that this algorithm has running time $O^*(4^k)$.

Exercise: Running time

- Prove that this algorithm has running time $O^*(4^k)$.

Hint: Use the measure $k + cc(S)$, where $cc(S)$ is the number of connected components of $G[S]$.

Result for FEEDBACK VERTEX SET

Theorem 1

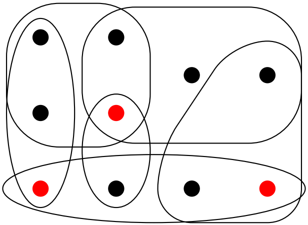
FEEDBACK VERTEX SET *can be solved in* $O^*(5^k)$ *time.*

Outline

Min r-Hitting Set

A set system $\mathcal{S}$ is a pair (V, H), where V is a finite set of elements and H is a set of subsets of V. The rank of $\mathcal{S}$ is the maximum size of a set in H, i.e., $\max_{Y \in H} |Y|$.
A hitting set of a set system $\mathcal{S} = (V, H)$ is a subset X of V such that X contains at least one element of each set in H, i.e., $X \cap Y \neq \emptyset$ for each $Y \in H$.

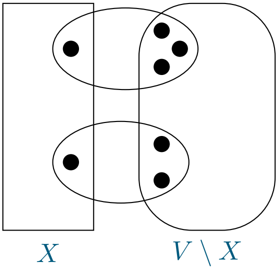
(universe)-MIN-r-HITTING SET (r-HS)	
Input:	A rank r set system $\mathcal{S} = (V, H)$
Parameter:	$n = \|V\|$
Output:	A smallest hitting set of $\mathcal{S}$



Note: The corresponsing decision problem is trivially FPT.

Compression Step

Comp-r-HS	
Input:	set system $\mathcal{S} = (V, H)$, integer k, hitting set X of size $k + 1$ of $\mathcal{S}$
Output:	a hitting set X^* of size $\leq k$ of $\mathcal{S}$ if one exists

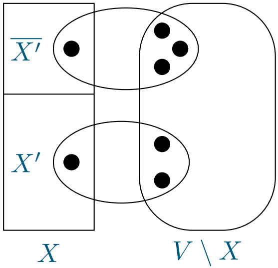


Compression Step

COMP-r-HS

Input: set system $\mathcal{S} = (V, H)$, integer k, hitting set X of size $k+1$ of $\mathcal{S}$

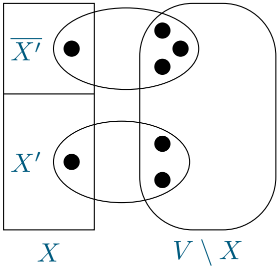
Output: a hitting set X^* of size $\leq k$ of $\mathcal{S}$ if one exists



Go over all partitions $(X', \overline{X'})$ of X such that $|X'| \geq 2|X| - n - 1$.

Compression Step

Comp-r-HS
Input: set system $\mathcal{S} = (V, H)$, integer k, hitting set X of size $k+1$ of $\mathcal{S}$
Output: a hitting set X^* of size $\leq k$ of $\mathcal{S}$ if one exists

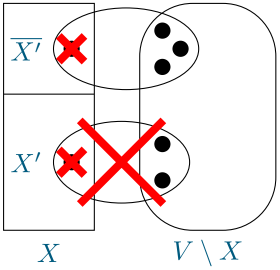


Reject a partition if there is a $Y \in H$ such that $Y \subseteq \overline{X'}$.

Compression Step

COMP-r-HS

Input:	set system $\mathcal{S} = (V, H)$, integer k, hitting set X of size $k+1$ of $\mathcal{S}$
Output:	a hitting set X^* of size $\leq k$ of $\mathcal{S}$ if one exists



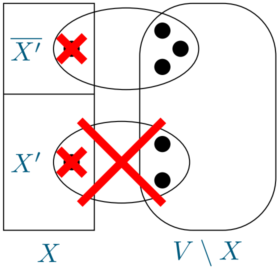
Compute a hitting set X'' of size $\leq k - |X'|$ for (V', H'), where $V' = V \setminus X$ and $H' = \{Y \cap V \ : \ Y \in H \wedge Y \cap X' = \emptyset\}$, if one exists.

Compression Step

COMP-r-HS

Input: set system $\mathcal{S} = (V, H)$, integer k, hitting set X of size $k+1$ of $\mathcal{S}$

Output: a hitting set X^* of size $\leq k$ of $\mathcal{S}$ if one exists



If one exists, then return $X^* = X' \cup X''$.

Compression Step II

- The algorithm considers only partitions into $(X', \overline{X'})$ such that $|X'| \geq 2|X| - n - 1$.
  Number of partitions:

$$O\left(\max\left\{2^{2n/3}, \max_{2n/3 \leq j \leq n} \binom{j}{2j-n}\right\}\right) = O\left(\max_{2n/3 \leq j \leq n} \binom{j}{2j-n}\right)$$

Compression Step II

- The algorithm considers only partitions into $(X', \overline{X'})$ such that $|X'| \geq 2|X| - n - 1$.
  Number of partitions:

$$O\left(\max\left\{2^{2n/3}, \max_{2n/3 \leq j \leq n} \binom{j}{2j-n}\right\}\right) = O\left(\max_{2n/3 \leq j \leq n} \binom{j}{2j-n}\right)$$

- The subinstances (V', H') where $V' = V \setminus X$ and $H' = \{Y \cap V \ : \ Y \in H \wedge Y \cap X' = \emptyset\}$ are instances of $(r-1)$-HS

Compression Step II

- The algorithm considers only partitions into $(X', \overline{X'})$ such that $|X'| \geq 2|X| - n - 1$.
  Number of partitions:

$$O\left(\max\left\{2^{2n/3}, \max_{2n/3 \leq j \leq n} \binom{j}{2j-n}\right\}\right) = O\left(\max_{2n/3 \leq j \leq n} \binom{j}{2j-n}\right)$$

- The subinstances (V', H') where $V' = V \setminus X$ and $H' = \{Y \cap V \ : \ Y \in H \wedge Y \cap X' = \emptyset\}$ are instances of $(r-1)$-HS
- Suppose $(r-1)$-HS can be solved in $O^*((\alpha_{r-1})^n)$ time. Then, r-HS can be solved in

$$O^*\left(\max_{2n/3 \leq j \leq n} \binom{j}{2j-n} (\alpha_{r-1})^{n-j}\right) \text{ time} \qquad (1)$$

- For example, using a $O(1.6278^n)$ algorithm for 3-HS [Wahlström '07], we obtain a $O(1.8704^n)$ time algorithm for 4-HS [1].

[1] the maximum in (1) is obtained for $j \approx 0.6824 \cdot n$

Iteration Step

- (V, H) instance of r-HS with $V = \{v_1, v_2, \ldots, v_n\}$
- $V_i = \{v_1, v_2, \ldots, v_i\}$ for $i = 1$ to n
- $H_i = \{Y \in H \ : \ Y \subseteq V_i\}$

Iteration Step

- (V, H) instance of r-HS with $V = \{v_1, v_2, \dots, v_n\}$
- $V_i = \{v_1, v_2, \dots, v_i\}$ for $i = 1$ to n
- $H_i = \{Y \in H \ : \ Y \subseteq V_i\}$
- Note that $|X_{i-1}| \leq |X_i| \leq |X_{i-1}| + 1$ where X_j is a minimum hitting set of the instance (V_i, H_i)

4-HS

Theorem 2 ([Fomin, Gaspers, Kratsch, Liedloff, and Saurabh, 2010])

4-HS can be solved in $O(1.8704^n)$ time.

4-HS

Theorem 2 ([Fomin, Gaspers, Kratsch, Liedloff, and Saurabh, 2010])

4-HS can be solved in $O(1.8704^n)$ time.

- One can generalize this result to the counting version of r-HS for any fixed r: count the number of minimum hitting sets of the given set system.

#r-Hitting Set

Theorem 3 ([Fomin, Gaspers, Kratsch, Liedloff, and Saurabh, 2010])

If there exists a $O^*((\alpha_{k-1})^n)$ time algorithm for $\#(r-1)$-HS with $\alpha_{r-1} \leq 2$, then $\#r$-HS can be solved in time

$$O^*\left(\max_{2n/3 \leq j \leq n} \left\{ \binom{j}{2j-n} (\alpha_{r-1})^{n-j} \right\}\right).$$

#r-Hitting Set

Theorem 3 ([Fomin, Gaspers, Kratsch, Liedloff, and Saurabh, 2010])

If there exists a $O^*((\alpha_{k-1})^n)$ time algorithm for $\#(r-1)$-HS with $\alpha_{r-1} \leq 2$, then $\#r$-HS can be solved in time

$$O^*\left(\max_{2n/3 \leq j \leq n} \left\{ \binom{j}{2j-n} (\alpha_{r-1})^{n-j} \right\}\right).$$

- If $\alpha_{r-1} \geq 1.6553$, then the following result is better

Theorem 4 ([Fomin, Gaspers, Kratsch, Liedloff, and Saurabh, 2010])

If there exists a $O^*((\alpha_{r-1})^n)$ time algorithm for $\#(r-1)$-HS with $\alpha_{k-1} \leq 2$, then $\#r$-HS can be solved in time

$$\min_{0.5 \leq \beta \leq 1} \max \left\{ O^*\left(\binom{n}{\beta n}\right), O^*\left(2^{\beta n} (\alpha_{r-1})^{n-\beta n}\right) \right\}.$$

Results for r-HS and #r-HS

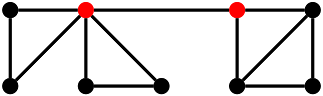
r	$\#r$-HS	r-HS
2	$O(1.2377^n)$ [Wahlström '08]	$O(1.2002^n)$ [Xiao, Nagamoshi '13]
3	$O(1.7198^n)$ [Theorem 3]	$O(1.6278^n)$ [Wahlström '07]
4	$O(1.8997^n)$ [Theorem 4]	$O(1.8704^n)$ [Theorem 3]
5	$O(1.9594^n)$ [Theorem 4]	$O(1.9489^n)$ [Theorem 4]
6	$O(1.9824^n)$ [Theorem 4]	$O(1.9781^n)$ [Theorem 4]
7	$O(1.9920^n)$ [Theorem 4]	$O(1.9902^n)$ [Theorem 4]

Exercise

A cluster graph is a graph where every connected component is a complete graph.

CLUSTER VERTEX DELETION

Input:	Graph $G = (V, E)$, integer k
Parameter:	k
Question:	Is there a set of vertices $S \subseteq V$ with $\|S\| \leq k$ such that $G - S$ is a cluster graph?



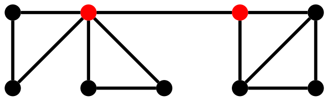
Recall that G is a cluster graph iff G contains no induced P_3.

- Design an $O^*(2^k)$ time algorithm for CLUSTER VERTEX DELETION.

Exercise

A cluster graph is a graph where every connected component is a complete graph.

CLUSTER VERTEX DELETION	
Input:	Graph $G = (V, E)$, integer k
Parameter:	k
Question:	Is there a set of vertices $S \subseteq V$ with $|S| \leq k$ such that $G - S$ is a cluster graph?



Recall that G is a cluster graph iff G contains no induced P_3.

- Design an $O^*(2^k)$ time algorithm for CLUSTER VERTEX DELETION.

Hints: (1) Show that the disjoint version of the problem can be solved in polynomial time: given $(G = (V, E), S, k)$ such that $|S| = k + 1$ and $G - S$ is a cluster graph, find a $S^* \subseteq V \setminus S$ with $|S^*| \leq k$ such that $G - S^*$ is a cluster graph. (2) Simplification rule for $v \in V \setminus S$ inducing a P_3 with 2 vertices in S. Reduce to maximum weight matching.

Solution sketch

Disjoint-CVD	
Input:	graph $G = (V, E)$, integer k, cluster vertex deletion set S of size $k+1$ of G
Output:	a cluster vertex deletion set S^* of G with $\lvert S^* \rvert \leq k$ and $S^* \cap S = \emptyset$, if one exists

Simplification rules:

- If $G[S]$ contains an induced P_3, then return No.
- If $\exists v \in V \setminus S$ such that $G[S \cup \{v\}]$ contains an induced P_3, then set $G \leftarrow G - v$ and $k \leftarrow k - 1$.

Now each vertex in $V \setminus S$ has either no neighbor in S or is adjacent to all the vertices of exactly one cluster of $G[S]$.
Reduce the problem to maximum weighted matching in an auxiliary graph where one independent set corresponds to the clusters in $G[S]$ and each vertex in the other independent set corresponds to cliques neighboring exactly one cluster in $G[S]$. It remains to define the edges of the auxiliary graph and their weights.

Outline

Further Reading

- Chapter 4, *Iterative Compression* in
  Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, MichałPilipczuk, and Saket Saurabh. Parameterized Algorithms. Springer, 2015.
- Section 11.3, *Iterative Compression* in
  Rolf Niedermeier. Invitation to Fixed Parameter Algorithms. Oxford University Press, 2006.
- Section 6.1, *Iterative Compression: The Basic Technique* in
  Rodney G. Downey and Michael R. Fellows. Fundamentals of Parameterized Complexity. Springer, 2013.
- Section 6.2, *Edge Bipartization* in
  Rodney G. Downey and Michael R. Fellows. Fundamentals of Parameterized Complexity. Springer, 2013.