11. Kernel Lower Bounds
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

1. Reminder
2. A kernel for Hamiltonian Cycle
3. A kernel for Edge Clique Cover
4. Compression
5. Kernel Lower Bounds
6. Further Reading

Outline

1. Reminder
2. A kernel for Hamiltonian Cycle
3. A kernel for Edge Clique Cover
4. Compression
5. Kernel Lower Bounds
6. Further Reading

Kernelization

Definition 1

A kernelization (kernel) for a parameterized problem Π is a **polynomial time** algorithm, which, for any instance I of Π with parameter k, produces an **equivalent** instance I' of Π with parameter k' such that $|I'| \leq f(k)$ and $k' \leq f(k)$ for a computable function f.
We refer to the function f as the size of the kernel.

Fixed-parameter tractability

Definition 2

A parameterized problem Π is fixed-parameter tractable (FPT) if there is an algorithm solving Π in time $f(k) \cdot \mathrm{poly}(n)$, where n is the instance size, k is the parameter, poly is a polynomial function, and f is a computable function.

Theorem 3

Let Π be a decidable parameterized problem.
Π has a kernelization $\Leftrightarrow$ Π is FPT.

Outline

1. Reminder
2. A kernel for Hamiltonian Cycle
3. A kernel for Edge Clique Cover
4. Compression
5. Kernel Lower Bounds
6. Further Reading

Hamiltonian Cycle I

A Hamiltonian cycle of G is a subgraph of G that is a cycle on $|V(G)|$ vertices.

vc-Hamiltonian Cycle	
Input:	A graph $G = (V, E)$.
Parameter:	$k = vc(G)$, the size of a smallest vertex cover of G.
Question:	Does G have a Hamiltonian cycle?

Thought experiment: Imagine a very large instance where the parameter is tiny. How can you simplify such an instance?

Hamiltonian Cycle II

Issue: We do not actually know a vertex cover of size k.

Hamiltonian Cycle III

- Obtain a vertex cover of size $\leq 2k$ by applying Vertex Cover-kernelizations to $(G, 0), (G, 1), \ldots$ until the first instance where no trivial No-instance is returned.
- If C is a vertex cover of size $\leq 2k$, then $I = V \setminus C$ is an independent set of size $\geq |V| - 2k$.
- No two consecutive vertices in the Hamiltonian Cycle can be in I.
- A kernel with $\leq 4k$ vertices can now be obtained with the following simplification rule.

(Too-large)

Compute a vertex cover C of size $\leq 2k$ in polynomial time.
If $2|C| < |V|$, then return No

Outline

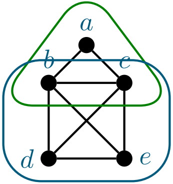
1. Reminder
2. A kernel for Hamiltonian Cycle
3. A kernel for Edge Clique Cover
4. Compression
5. Kernel Lower Bounds
6. Further Reading

Edge Clique Cover

Definition 4

An edge clique cover of a graph $G = (V, E)$ is a set of cliques in G covering all its edges.
In other words, if $\mathcal{C} \subseteq 2^V$ is an edge clique cover then each $S \in \mathcal{C}$ is a clique in G and for each $\{u, v\} \in E$ there exists an $S \in \mathcal{C}$ such that $u, v \in S$.

Example: $\{\{a, b, c\}, \{b, c, d, e\}\}$ is an edge clique cover for this graph.



Edge Clique Cover

Edge Clique Cover	
Input:	A graph $G = (V, E)$ and an integer k
Parameter:	k
Question:	Does G have an edge clique cover of size at most k?

The size of an edge clique cover $\mathcal{C}$ is the number of cliques contained in $\mathcal{C}$ and is denoted $|\mathcal{C}|$.

Helpful properties

Definition 4

A clique S in a graph G is a maximal clique if there is no other clique S' in G with $S \subset S'$.

Lemma 5

A graph G has an edge clique cover $\mathcal{C}$ of size at most k if and only if G has an edge clique cover $\mathcal{C}'$ of size at most k such that each $S \in \mathcal{C}'$ is a maximal clique.

Proof sketch.

$(\Rightarrow)$: Replace each clique $S \in \mathcal{C}$ by a maximal clique S' with $S \subseteq S'$.
$(\Leftarrow)$: Trivial, since $\mathcal{C}'$ is an edge clique cover of size at most k. $\square$

Simplification rules for Edge Clique Cover

Thought experiment: Imagine a very large instance where the parameter is tiny. How can you simplify such an instance?

Simplification rules for Edge Clique Cover II

The instance could have many degree-0 vertices.

(Isolated)

If there exists a vertex $v \in V$ with $d_G(v) = 0$, then set $G \leftarrow G - v$.

Lemma 6

(Isolated) is sound.

Proof sketch.

Since no edge is incident to v, a smallest edge clique cover for $G - v$ is a smallest edge clique cover for G, and vice-versa. $\square$

Simplification rules for Edge Clique Cover II

The instance could have many degree-0 vertices.

(Isolated)

If there exists a vertex $v \in V$ with $d_G(v) = 0$, then set $G \leftarrow G - v$.

Lemma 6

(Isolated) is sound.

Proof sketch.

Since no edge is incident to v, a smallest edge clique cover for $G - v$ is a smallest edge clique cover for G, and vice-versa. □

(Isolated-Edge)

If $\exists uv \in E$ such that $d_G(u) = d_G(v) = 1$, then set $G \leftarrow G - \{u, v\}$ and $k \leftarrow k - 1$.

Simplification rules for EDGE CLIQUE COVER III

(Twins)

If $\exists u, v \in V$, $u \neq v$, such that $N_G[u] = N_G[v]$, then set $G \leftarrow G - v$.

Lemma 7

(Twins) is sound.

Simplification rules for Edge Clique Cover III

(Twins)

If $\exists u, v \in V$, $u \neq v$, such that $N_G[u] = N_G[v]$, then set $G \leftarrow G - v$.

Lemma 7

(Twins) is sound.

Proof.

We need to show that G has an edge clique cover of size at most k if and only if $G - v$ has an edge clique cover of size at most k.
($\Rightarrow$): If $\mathcal{C}$ is an edge clique cover of G of size at most k, then $\{S \setminus \{v\} : S \in \mathcal{C}\}$ is an edge clique cover of $G - v$ of size at most k.
($\Leftarrow$): Let $\mathcal{C}'$ be an edge clique cover of $G - v$ of size at most k. Partition $\mathcal{C}$ into $\mathcal{C}_u = \{S \in \mathcal{C} : u \in S\}$ and $\mathcal{C}_{\neg u} = \mathcal{C} \setminus \mathcal{C}_u$. Note that each set in $\mathcal{C}'_u = \{S \cup \{v\} : S \in \mathcal{C}_u\}$ is a clique since $N_G[u] = N_G[v]$ and that each edge incident to v is contained in at least one of these cliques. Now, $\mathcal{C}'_u \cup \mathcal{C}_{\neg u}$ is an edge clique cover of G of size at most k. □

Simplification rules for Edge Clique Cover IV

(Size-V)

If the previous simplification rules do not apply and $|V| > 2^k$, then return No.

Lemma 8

(Size-V) is sound.

Simplification rules for EDGE CLIQUE COVER IV

(Size-V)

If the previous simplification rules do not apply and $|V| > 2^k$, then return NO.

Lemma 8

(Size-V) is sound.

Proof.

For the sake of contradiction, assume neither (Isolated) nor (Twins) are applicable, $|V| > 2^k$, and G has an edge clique cover $\mathcal{C}$ of size at most k. Since $2^{\mathcal{C}}$ (the set of all subsets of $\mathcal{C}$) has size at most 2^k, and every vertex belongs to at least one clique in $\mathcal{C}$ by (Isolated), we have that there exists two vertices $u, v \in V$ such that $\{S \in \mathcal{C} : u \in S\} = \{S \in \mathcal{C} : v \in S\}$. But then, $N_G[u] = \bigcup_{S \in \mathcal{C} : u \in S} S = \bigcup_{S \in \mathcal{C} : v \in S} S = N_G[v]$, contradicting that (Twin) is not applicable. $\square$

Kernel for EDGE CLIQUE COVER

Theorem 9

EDGE CLIQUE COVER *has a kernel with* $O(2^k)$ *vertices and* $O(4^k)$ *edges.*

Corollary 10

EDGE CLIQUE COVER *is* FPT.

Possible issues designing simplification rules

Issue 1: A kernelization needs to produce an instance of the same problem.

How could we turn the following lemma into a simplification rule?

Lemma 11

If there is an edge $\{u, v\} \in E$ *such that* $S = N_G[u] \cap N_G[v]$ *is a clique, then there is a smallest edge clique cover* $\mathcal{C}$ *with* $S \in \mathcal{C}$.

Proof.

By Lemma 5, we may assume the clique covering the edge $\{u, v\}$ is a maximal clique. But, S is the unique maximal clique covering $\{u, v\}$. □

Possible issues designing simplification rules

(Neighborhood-Clique)

If there exists $\{u, v\} \in E$ such that $S = N_G[u] \cap N_G[v]$ is a clique, then ...???

Edges with both endpoints in $S \setminus \{u, v\}$ are covered by S but might still be needed in other cliques.

Possible issues designing simplification rules

We could design a kernelization for a more general problem.

GENERALIZED EDGE CLIQUE COVER	
Input:	A graph $G = (V, E)$, a set of edges $R \subseteq E$, and an integer k
Parameter:	k
Question:	Is there a set $\mathcal{C}$ of at most k cliques in G such that each $e \in R$ is contained in at least one of these cliques?

(Neighborhood-Clique)

If there exists $\{u, v\} \in R$ such that $S = N_G[u] \cap N_G[v]$ is a clique, then set $G \leftarrow (V, E \setminus \{u, v\})$, $R \leftarrow R \setminus \{\{x, y\} : x, y \in S\}$, and $k \leftarrow k - 1$.

Possible issues designing simplification rules

Issue 2: A proposed simplification rule might not be sound.

Consider the following simplification rule for Vertex Cover.

(Optimistic-Degree-($\geq k$))

If $\exists v \in V$ such that $d_G(v) \geq k$, then set $G \leftarrow G - v$ and $k \leftarrow k - 1$.

To show that a simplification rule is not sound, we exhibit a counter-example.

Lemma 11

(Optimistic-Degree-($\geq k$)) is not sound for Vertex Cover.

Possible issues designing simplification rules

(Optimistic-Degree-($\geq k$))

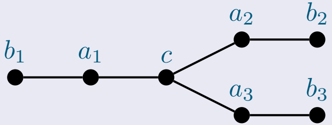
If $\exists v \in V$ such that $d_G(v) \geq k$, then set $G \leftarrow G - v$ and $k \leftarrow k - 1$.

Lemma 11

(Optimistic-Degree-($\geq k$)) is not sound for Vertex Cover.

Proof.

Consider the instance consisting of the following graph and $k = 3$.



Since $M = \{\{a_i, b_i\} : 1 \leq i \leq 3\}$ is a matching, a vertex cover contains at least one endpoint of each edge in M. The rule would add c to the vertex cover, leading to a vertex cover of size at least 4. However, $\{a_i : 1 \leq i \leq 3\}$ is a vertex cover of size 3. □

Possible issues designing simplification rules

Issue 3: A problem might be FPT, but only an exponential kernel might be known / possible to achieve.

Outline

1 Reminder

2 A kernel for HAMILTONIAN CYCLE

3 A kernel for EDGE CLIQUE COVER

4 Compression

5 Kernel Lower Bounds

6 Further Reading

Definition

Definition 11

A compression from a parameterized problem Π_1 to a problem Π_2 (the problem Π_2 is not necessarily parameterized) is a polynomial time algorithm, which, for any instance I_1 of Π_1 with parameter k_1, produces an **equivalent** instance I_2 of Π_2 such that $|I_2| \leq f(k_1)$ for a computable function f.
We refer to the function f as the size of the compression.

Note: A kernelization is a compression where $\Pi_1 = \Pi_2$.

Compressions lead to Kernels

Theorem 12

Let Π_1 be an NP-hard parameterized problem and Π_2 be a problem in NP. If Π_1 has a **polynomial** *compression to Π_2, then Π_1 has a* **polynomial** *kernel.*

Proof.

Denote by R a polynomial-time reduction from Π_2 to Π_1. Such a reduction exists by the definition of NP-hardness (a problem is NP-hard if every problem in NP can be reduced to it in polynomial time.)
Let I_1 be an instance for Π_1 with parameter k_1. Apply the polynomial compression to I_1 to obtain an equivalent instance I_2 for Π_2 such that $|I_2| \in (k_1)^{O(1)}$. Now, $|R(I_2)| \in (k_1)^{O(1)}$. □

Outline

1. Reminder
2. A kernel for Hamiltonian Cycle
3. A kernel for Edge Clique Cover
4. Compression
5. Kernel Lower Bounds
6. Further Reading

Polynomial vs. exponential kernels

- For some FPT problems, only exponential kernels are known.
- Could it be that all FPT problems have polynomial kernels?
- We will see that polynomial kernels for some fixed-parameter tractable parameterized problems would contradict complexity-theoretic assumptions.

Intuition by example

Long Path	
Input:	A graph $G = (V, E)$, and an integer $k \leq \lvert V \rvert$.
Parameter:	k
Question:	Does G have a path of length at least k (as a subgraph)?

Long Path is NP-complete but FPT.

Intuition by example

- Assume Long Path has a k^c kernel, where $c = O(1)$.
- Set $q = k^c + 1$ and consider q instances with the same parameter k:

$$(G_1, k), (G_2, k), \ldots, (G_q, k).$$

- Let $G = G_1 \oplus G_2 \oplus \cdots \oplus G_q$ be the disjoint union of all these graphs.
- Note that (G, k) is a Yes-instance if and only if at least one of $(G_i, k), 1 \leq i \leq q$, is a Yes-instance.
- Kernelizing (G, k) gives an instance of size k^c, i.e., on average less than one bit per original instance.
- "The kernelization must have solved at least one of the original NP-hard instances in polynomial time".

Distillation

Definition 13

Let Π_1, Π_2 be two problems. An OR-distillation (resp., AND-distillation) from Π_1 into Π_2 is a polynomial time algorithm D whose input is a sequence $I_1, \dots, I_q$ of instances for Π_1 and whose output is an instance I' for Π_2 such that

- $|I'| \leq \text{poly}(\max_{1 \leq i \leq q} |I_i|)$, and
- I' is a $\textsc{Yes}$-instance for Π_2 if and only if for at least one (resp., for each) $i \in \{1, \dots, q\}$ we have that I_i is a $\textsc{Yes}$-instance for Π_1.

NP-complete problems don't have distillations

Theorem 14 ([Fortnow, Santhanam, 2008])

If any NP-complete problem has an OR-distillation, then coNP $\subseteq$ NP/poly. [1]

Note: coNP $\subseteq$ NP/poly is not believed to be true and it would imply that the polynomial hierarchy collapses to the third level: PH $\subseteq \Sigma_3^p$.

Theorem 15 ([Drucker, 2012])

If any NP-complete problem has an AND-distillation, then coNP $\subseteq$ NP/poly.

[1] NP/poly is the class of all decision problems for which there exists a polynomial-time nondeterministic Turing Machine M with the following property: for every $n \geq 0$, there is an advice string A of length poly(n) such that, for every input I of length n, the machine M correctly decides the problem with input I, given I and A.

Composition algorithms

Definition 16

Let Π be a parameterized problem. An OR-composition (resp., AND-composition) of Π is a polynomial time algorithm A that receives as input a finite sequence $I_1, \ldots, I_q$ of Π with parameters $k_1 = \cdots = k_q = k$ and outputs an instance I' for Π with parameter k' such that

- $k' \leq \text{poly}(k)$, and
- I' is a $\textsc{Yes}$-instance for Π if and only if for at least one (resp., for each) $i \in \{1, \ldots, q\}$, I_i is a $\textsc{Yes}$-instance for Π.

Tool for showing kernel lower bounds

Theorem 17 (Composition Theorem)

Let Π be an NP-complete parameterized problem such that for each instance I of Π with parameter k, the value of the parameter k can be computed in polynomial time and $k \leq |I|$. If Π has an OR-composition or an AND-composition, then Π has no polynomial kernel, unless coNP $\subseteq$ NP/poly.

Tool for showing kernel lower bounds

Theorem 17 (Composition Theorem)

Let Π be an NP-complete parameterized problem such that for each instance I of Π with parameter k, the value of the parameter k can be computed in polynomial time and $k \leq |I|$. If Π has an OR-composition or an AND-composition, then Π has no polynomial kernel, unless coNP $\subseteq$ NP/poly.

Proof sketch.

Suppose Π has an OR/AND-composition and a polynomial kernel. Then, one can obtain an OR/AND-distillation from Π into OR(Π)/AND(Π).

Tool for showing kernel lower bounds

Theorem 17 (Composition Theorem)

Let Π be an NP-complete parameterized problem such that for each instance I of Π with parameter k, the value of the parameter k can be computed in polynomial time and $k \leq |I|$. If Π has an OR-composition or an AND-composition, then Π has no polynomial kernel, unless coNP $\subseteq$ NP/poly.

Proof sketch.

Suppose Π has an OR/AND-composition and a polynomial kernel. Then, one can obtain an OR/AND-distillation from Π into OR(Π)/AND(Π).

$I_1 \quad I_2 \quad \ldots \quad I_q$	q instances of size $\leq n = \max_{1 \leq i \leq q} \|I_i\|$
$\{I_i : k_i = 0\} \ldots \{I_i : k_i = n\}$	group by parameter
$I'_0 \quad I'_1 \quad \ldots \quad I'_n$	After OR-composition: $n+1$ instances with $k'_i \leq$ poly(n)
$I''_0 \quad I''_1 \quad \ldots \quad I''_n$	After kernelization: $n+1$ instances of size poly(n) each
	This is an instance of OR(Π) of size poly(n).

□

Long Path has no polynomial kernel I

Theorem 18

Long Path *has no polynomial kernel unless* NP $\subseteq$ coNP/poly.

Proof.

Clearly, k can be computed in polynomial time and $k \leq |V|$.
We give an OR-composition for Long Path, which will prove the theorem by the previous lemma.
It receives as input a sequence of instances for Long Path: $(G_1, k), \ldots, (G_q, k)$, and it produces the instance $(G_1 \oplus \cdots \oplus G_q, k)$, which is a Yes-instance if and only if at least one of $(G_1, k), \ldots, (G_q, k)$ is a Yes-instance. □

var-SAT has no poly kernel I

var-SAT

Input:	A propositional formula F in conjunctive normal form (CNF)
Parameter:	$n = \lvert \mathrm{var}(F) \rvert$, the number of variables in F
Question:	Is there an assignment to $\mathrm{var}(F)$ satisfying all clauses of F?

Example:

$$(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (x_1 \vee x_4) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4)$$

or

$$\{\{x_1, x_2\}, \{\neg x_2, x_3, \neg x_4\}, \{x_1, x_4\}, \{\neg x_1, \neg x_3, \neg x_4\}\}$$

var-SAT has no poly kernel II

Theorem 19

var-SAT has no polynomial kernel unless NP $\subseteq$ coNP/poly.

Proof.

Clearly, var(F) can be computed in polynomial time and $n = |\text{var}(F)| \leq |F|$.
We give an OR-composition for var-SAT, which will prove the theorem by the previous lemma.

- Let $F_1, \dots, F_q$ be CNF formulas, $|F_i| \leq m$, $|\text{var}(F_i)| = n$.
- We can decide whether one of the formulas is satisfiable in time poly$(mt2^n)$. Hence, if $q > 2^n$, the check is polynomial. If some formula is satisfiable, we output this formula, otherwise we output F_1.

var-SAT has no poly kernel III

Proof (continued).

- It remains the case $q \leq 2^n$. We assume $\mathsf{var}(F_1) = \cdots = \mathsf{var}(F_q)$, otherwise we change the names of variables.
- Let $s = \lceil \log_2 q \rceil$. Since $q \leq 2^n$, we have that $s \leq n$.
- We take a set $Y = \{y_1, \ldots, y_s\}$ of new variables. Let $C_1, \ldots, C_{2^s}$ be the sequence of all 2^s possible clauses containing exactly s literals over the variables in Y.
- For $1 \leq i \leq q$ we let $F_i' = \{C \cup C_i : C \in F_i\}$.
- We define $F = \bigcup_{i=1}^{q} F_i' \cup \{C_i : q+1 \leq i \leq 2^s\}$.
- Claim: F is satisfiable if and only if F_i is satisfiable for some $1 \leq i \leq q$.
- Hence we have an OR-composition.

$\square$

Another tool for showing kernel lower bounds I

Definition 20

Let Π_1, Π_2 be parameterized problems. A polynomial parameter transformation from Π_1 to Π_2 is a polynomial time algorithm, which, for any instance I_1 of Π_1 with parameter k_1, produces an **equivalent** instance I_2 of Π_2 with parameter k_2 such that $k_2 \leq \text{poly}(k_1)$.

Another tool for showing kernel lower bounds II

Theorem 21

Let Π_1, Π_2 be parameterized problems such that Π_1 is NP-complete, Π_2 is in NP, and there is a polynomial parameter transformation from Π_1 to Π_2. If Π_2 has a polynomial kernel, then Π_1 has a polynomial kernel.

Remark: If we know that an NP-complete parameterized problem Π_1 has no polynomial kernel (unless NP $\subseteq$ coNP/poly), we can use the theorem to show that some other NP-complete parameterized problem Π_2 has no polynomial kernel (unless NP $\subseteq$ coNP/poly) by giving a polynomial parameter transformation from Π_1 to Π_2.

Another tool for showing kernel lower bounds III

Proof.

- We show that under the assumptions of the theorem Π_1 has a polynomial kernel.
- Let I_1 be an instance of Π_1 with parameter k_1.
- We obtain in polynomial time an equivalent instance I_2 of Π_2 with parameter $k_2 \le \text{poly}(k_1)$.
- We apply Π_2's kernelization and obtain I_2' of size $\le \text{poly}(k_1)$.
- Since Π_2 is in NP and Π_1 is NP-complete, there exists a polynomial time reduction that maps I_2' to an equivalent instance I_1' of Π_1.
- The size of I_1' is polynomial in k_1.

$\square$

2CNF-Backdoor Evaluation I

Definition 22

A CNF formula F is a 2CNF formula if each clause of F has at most 2 literals.

Note: SAT is polynomial time solvable when the input is restricted to be a 2CNF formula.

Definition 23

A 2CNF-backdoor of a CNF formula F is a set of variables $B \subseteq \text{var}(F)$ such that for each assignment $\alpha : B \to \{0,1\}$, the formula $F[\alpha]$ is a 2CNF formula.
Here, $F[\alpha]$ is obtained by removing all clauses containing a literal set to 1 by α, and removing the literals set to 0 from all remaining clauses.

2CNF-Backdoor Evaluation II

2CNF-Backdoor Evaluation	
Input:	A CNF formula F and a 2CNF-backdoor B of F
Parameter:	$k = \|B\|$
Question:	Is F satisfiable?

Note: the problem is FPT by trying all assignments to B and evaluating the resulting formulas.

2CNF-Backdoor Evaluation III

Theorem 24

2CNF-Backdoor Evaluation *has no polynomial kernel unless* NP $\subseteq$ coNP/poly.

Proof.

We give a polynomial parameter transformation from var-SAT to 2CNF-Backdoor Evaluation.
Let F be an instance for var-SAT.
Then, $(F, B = \text{var}(F))$ is an equivalent instance for 2CNF-Backdoor Evaluationwith $|B| \leq |\text{var}(F)|$. $\square$

Exercise

PATH PACKING

Input:	A graph G and an integer k
Parameter:	k
Question:	Are there k pairwise vertex-disjoint paths of length at least k each?

- Show that PATH PACKING has no polynomial kernel unless NP $\subseteq$ coNP/poly.

Hint: Compositions seem challenging.

Solution

Theorem 25

PATH PACKING *has no polynomial kernel unless* NP $\subseteq$ coNP/poly.

Proof.

We give a polynomial parameter transformation from LONG PATHto PATH PACKING.

Given an instance (G, k) to LONG PATHwe construct a graph G' from G by adding $k - 1$ vertex-disjoint paths of length k.

Now, G contains a path of length k if and only if G' contains k vertex-disjoint paths of length k. □

Outline

1. Reminder
2. A kernel for HAMILTONIAN CYCLE
3. A kernel for EDGE CLIQUE COVER
4. Compression
5. Kernel Lower Bounds
6. Further Reading

Further Reading

- Chapter 15, *Lower bounds for kernelization* in
  Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, MichałPilipczuk, and Saket Saurabh. Parameterized Algorithms. Springer, 2015.
- Chapter 30 (30.1–30.4), *Kernelization Lower Bounds* in
  Rodney G. Downey and Michael R. Fellows. Fundamentals of Parameterized Complexity. Springer, 2013.
- Neeldhara Misra, Venkatesh Raman, and Saket Saurabh. *Lower bounds on kernelization*. Discrete Optimization 8(1): 110-128 (2011).