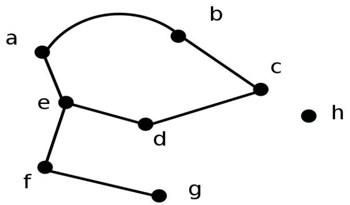
Parameterized and Exact Algorithms for Class Domination Coloring

R. Krithika[1] A. Rai[1] S. Saurabh[1,2] and P. Tale[1]

[1] The Institute of Mathematical Sciences, HBNI, Chennai, India
[2] University Of Bergen, Norway

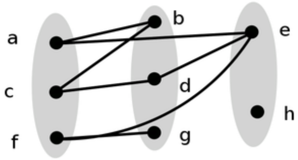
January 16, 2017

Coloring of graph

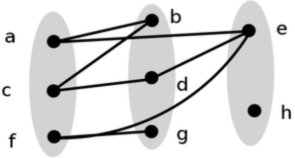


COLORING
Input: A graph G
Question: Find minimum integer q such that graph G can be partitioned into q independent sets

Coloring of graph

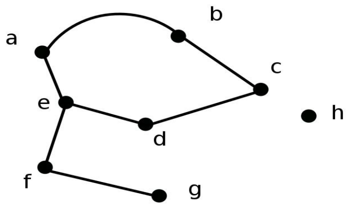


Coloring of graph



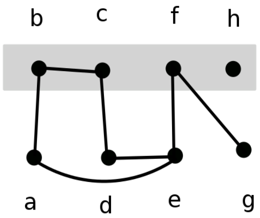
- One of Karp's 21 NP-Complete problems
- Computational Complexity : Determining whether given planer graph (which can be 4-colored) is 3-coloring or not is NP-Complete
- Exact Algorithms : $\mathcal{O}(2^n)$ which optimal under some widely believed complexity assumption
- Parameterized Complexity : Reduction from COLORING to refute existence of certain kind of algorithms

Dominating Set of Graph

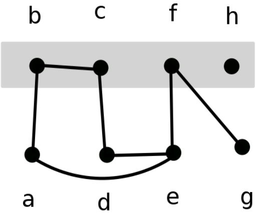


DOMINATING SET
Input: A graph G
Question: Find minimum int k such that there exists set dominating set D of cardinality k i.e. $V(G) = N[D]$

Dominating Set of Graph

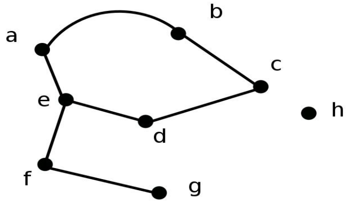


Dominating Set of Graph



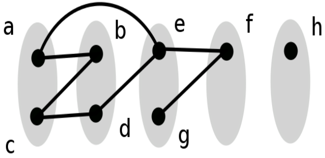
- Exact Algorithm : $\mathcal{O}(1.4969^n)$time and polynomial space.
- $\mathcal{O}(n^k)$ is optimal under certain complexity assumption
- *Complete* for certain classes in parameterized complexity

Class Domination Coloring

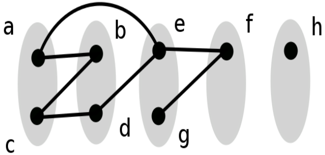


CD-COLORING
Input: A graph G
Question: Find minimum int q such that graph G can be partitioned into q independent sets and every independent set is contained in closed neighbourhood of some vertex

Class Domination Coloring



Class Domination Coloring



- Coloring such that for every color class, there is a vertex that dominates it
- flavour of both Coloring and Dominating Set
- NP-Complete even for Chordal Graphs

Outline

- Parameterized Complexity and Kernalization
  - Short introduction
  - FPT algorithms parameterized by solution size and tree-width
  - FPT algorithms on chordal graphs
  - Kernel for cd-coloring on graphs with girth ≥ 5
- Exact Algorithm
  - $\mathcal{O}^*(2^n)$ algorithm to compute cd-chromatic number
- cd-Partization problem
  - Hardness results
  - On Split graphs
  - Exact algorithm to solve

Parameterized complexity

- Pioneered by Downey and Fellows around 1978
- Goal : Find **better** ways to solve NP-hard problems
- Associate (*small*) parameter k to each instance I
- Restrict the combinatorial explosion to a parameter k
- Parameterized problem (I, k) is *fixed-parameter tractable (FPT)* if there is an algorithm that solves it in time $\mathcal{O}(f(k) \cdot |I|^{\mathcal{O}(1)})$

Parameterized complexity

- Pioneered by Downey and Fellows around 1978
- Goal : Find **better** ways to solve NP-hard problems
- Associate (*small*) parameter k to each instance I
- Restrict the combinatorial explosion to a parameter k
- Parameterized problem (I, k) is *fixed-parameter tractable (FPT)* if there is an algorithm that solves it in time $\mathcal{O}(f(k) \cdot |I|^{\mathcal{O}(1)})$

Parameterized complexity

- Pioneered by Downey and Fellows around 1978
- Goal : Find **better** ways to solve NP-hard problems
- Associate (*small*) parameter k to each instance I
- Restrict the combinatorial explosion to a parameter k
- Parameterized problem (I, k) is *fixed-parameter tractable (FPT)* if there is an algorithm that solves it in time $\mathcal{O}(f(k) \cdot |I|^{\mathcal{O}(1)})$

Parameterized complexity

- Pioneered by Downey and Fellows around 1978
- Goal : Find **better** ways to solve NP-hard problems
- Associate (*small*) parameter k to each instance I
- Restrict the combinatorial explosion to a parameter k
- Parameterized problem (I, k) is *fixed-parameter tractable (FPT)* if there is an algorithm that solves it in time $\mathcal{O}(f(k) \cdot |I|^{\mathcal{O}(1)})$

Parameterized complexity

- Pioneered by Downey and Fellows around 1978
- Goal : Find **better** ways to solve NP-hard problems
- Associate (*small*) parameter k to each instance I
- Restrict the combinatorial explosion to a parameter k
- Parameterized problem (I, k) is *fixed-parameter tractable (FPT)* if there is an algorithm that solves it in time $\mathcal{O}(f(k) \cdot |I|^{\mathcal{O}(1)})$

- Pioneered by Downey and Fellows around 1978
- Goal : Find **better** ways to solve NP-hard problems
- Associate (*small*) parameter k to each instance I
- Restrict the combinatorial explosion to a parameter k
- Parameterized problem (I, k) is *fixed-parameter tractable (FPT)* if there is an algorithm that solves it in time $\mathcal{O}(f(k) \cdot |I|^{\mathcal{O}(1)})$

CD-COLORING
Input: A graph G
Question: Find minimum int q such that graph G can be partitioned into q independent sets and every independent set is contained in closed neighbourhood of some vertex

Parameterized complexity

- Pioneered by Downey and Fellows around 1978
- Goal : Find **better** ways to solve NP-hard problems
- Associate (*small*) parameter k to each instance I
- Restrict the combinatorial explosion to a parameter k
- Parameterized problem (I, k) is *fixed-parameter tractable (FPT)* if there is an algorithm that solves it in time $\mathcal{O}(f(k) \cdot |I|^{\mathcal{O}(1)})$

CD-COLORING **Parameter:** q
Input: A graph G, integer q
Question: Can graph G be partitioned into q independent sets such that every independent set is contained in closed neighbourhood of some vertex?

Parameterized complexity

Problem	$f(k)$
Vertex Cover(G, k)	$\mathcal{O}(1.27^k \cdot n^2)$
Feedback Vertex Set(G, k)	$\mathcal{O}(3.6181^k \cdot n^c)$
Independent Set(G, k)	No $f(k) \cdot \lvert I \rvert^{\mathcal{O}(1)}$ algorithm
Coloring(G, k)	No $f(k) \cdot \lvert I \rvert^{\mathcal{O}(1)}$ algorithm

Kernelization

- Mathematical analysis of pre-processing
- Goal: Reduce the size of input instance without changing the answer (in polynomial time)
- Parameterized problem (I, k) admits a $h(k)$-kernel if there is a polynomial time algorithm that reduces (I, k) to an equisatisfiable instance (I', k') such that $|I'| + k' \leq h(k)$.

Problem	$h(k)$
Vertex Cover(G, k)	$2k$
Feedback Vertex Set(G, k)	$4k^2$
Independent Set(G, k)	No such $h(k)$ exists
Coloring(G, k)	No such $h(k)$ exists

Kernelization

- Mathematical analysis of pre-processing
- Goal: Reduce the size of input instance without changing the answer (in polynomial time)
- Parameterized problem (I, k) admits a $h(k)$-kernel if there is a polynomial time algorithm that reduces (I, k) to an equisatisfiable instance (I', k') such that $|I'| + k' \leq h(k)$.

Problem	$h(k)$
Vertex Cover(G, k)	$2k$
Feedback Vertex Set(G, k)	$4k^2$
Independent Set(G, k)	No such $h(k)$ exists
Coloring(G, k)	No such $h(k)$ exists

Kernelization

- Mathematical analysis of pre-processing
- Goal: Reduce the size of input instance without changing the answer (in polynomial time)
- Parameterized problem (I, k) admits a $h(k)$-kernel if there is a polynomial time algorithm that reduces (I, k) to an equisatisfiable instance (I', k') such that $|I'| + k' \leq h(k)$.

Problem	$h(k)$
Vertex Cover(G, k)	$2k$
Feedback Vertex Set(G, k)	$4k^2$
Independent Set(G, k)	No such $h(k)$ exists
Coloring(G, k)	No such $h(k)$ exists

Kernelization

- Mathematical analysis of pre-processing
- Goal: Reduce the size of input instance without changing the answer (in polynomial time)
- Parameterized problem (I, k) admits a $h(k)$-kernel if there is a polynomial time algorithm that reduces (I, k) to an equisatisfiable instance (I', k') such that $|I'| + k' \leq h(k)$.

Problem	$h(k)$
Vertex Cover(G, k)	$2k$
Feedback Vertex Set(G, k)	$4k^2$
Independent Set(G, k)	No such $h(k)$ exists
Coloring(G, k)	No such $h(k)$ exists

Kernelization

- Mathematical analysis of pre-processing
- Goal: Reduce the size of input instance without changing the answer (in polynomial time)
- Parameterized problem (I, k) admits a $h(k)$-kernel if there is a polynomial time algorithm that reduces (I, k) to an equisatisfiable instance (I', k') such that $|I'| + k' \leq h(k)$.

Problem	$h(k)$
Vertex Cover(G, k)	$2k$
Feedback Vertex Set(G, k)	$4k^2$
Independent Set(G, k)	No such $h(k)$ exists
Coloring(G, k)	No such $h(k)$ exists

FPT algorithms and Kernalization

Problem	FPT	Kernel
VERTEX COVER	$\mathcal{O}(1.27^k \cdot n^2)$	$2k$
FEEDBACK VERTEX SET	$\mathcal{O}(3.6181^k \cdot n^c)$	$4k^2$
INDEPENDENT SET	No $f(k) \cdot \|I\|^{\mathcal{O}(1)}$	No $h(k)$
COLORING	No $f(k) \cdot \|I\|^{\mathcal{O}(1)}$	No $h(k)$

Theorem

A parameterized problem (I, k) is FPT if and only if it admits a kernel.

FPT algorithms and Kernalization

Problem	FPT	Kernel
Vertex Cover	$\mathcal{O}(1.27^k \cdot n^2)$	$2k$
Feedback Vertex Set	$\mathcal{O}(3.6181^k \cdot n^c)$	$4k^2$
Independent Set	No $f(k) \cdot \|I\|^{\mathcal{O}(1)}$	No $h(k)$
Coloring	No $f(k) \cdot \|I\|^{\mathcal{O}(1)}$	No $h(k)$

Theorem

A parameterized problem (I, k) is FPT if and only if it admits a kernel.

FPT algorithm on general graph

- Parameter : Solution size
- $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING?

FPT algorithm on general graph

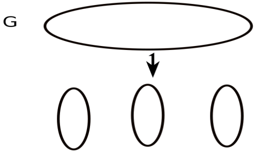
- Parameter : Solution size
- $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING?

FPT algorithm on general graph

- Parameter : Solution size
- $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING?

Theorem

COLORING$(G, 3)$ is NP-complete.

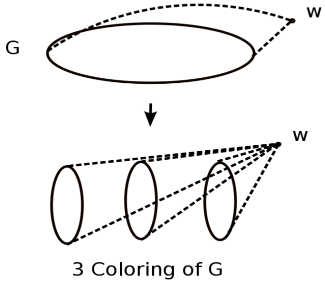


FPT algorithm on general graph

- Parameter : Solution size
- $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING?

Theorem

COLORING$(G, 3)$ is NP-complete.

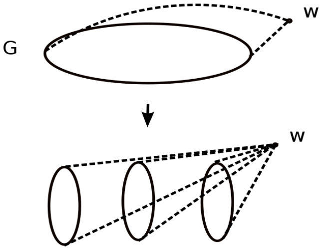


FPT algorithm on general graph

- Parameter : Solution size
- $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING?

Theorem

CD-COLORING$(G, 4)$ is NP-complete.



3 Coloring of G

FPT algorithm on general graph

- Parameter : Solution size
- $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING?
  - $\Rightarrow \mathcal{O}(f(4) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING when $q = 4$
  - $\Rightarrow \mathcal{O}(n^{\mathcal{O}(1)})$ algorithm for NP-Complete problem
- para-NP-hard problems

FPT algorithm on general graph

- Parameter : Solution size
- $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING?

  $\Rightarrow \mathcal{O}(f(4) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING when $q = 4$

  $\Rightarrow \mathcal{O}(n^{\mathcal{O}(1)})$ algorithm for NP-Complete problem

- para-NP-hard problems

FPT algorithm on general graph

- Parameter : Solution size
- $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING?

  $\Rightarrow \mathcal{O}(f(4) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING when $q = 4$

  $\Rightarrow \mathcal{O}(n^{\mathcal{O}(1)})$ algorithm for NP-Complete problem
- para-NP-hard problems

FPT algorithm on general graph

- Parameter : Solution size
- $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING?
  $\Rightarrow \mathcal{O}(f(4) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-COLORING when $q = 4$
  $\Rightarrow \mathcal{O}(n^{\mathcal{O}(1)})$ algorithm for NP-Complete problem
- para-NP-hard problems

FPT algorithm on general graph

- Parameter : Solution size
- $\mathcal{O}(f(k) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-Coloring?

  $\Rightarrow \mathcal{O}(f(4) \cdot n^{\mathcal{O}(1)})$ algorithm for CD-Coloring when $q = 4$

  $\Rightarrow \mathcal{O}(n^{\mathcal{O}(1)})$ algorithm for NP-Complete problem
- para-NP-hard problems

FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

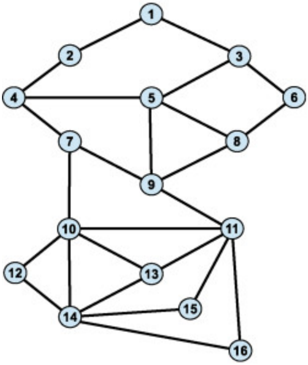
FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

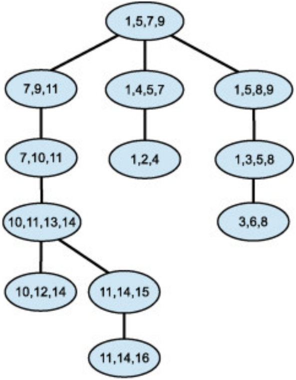
FPT algorithm on general graph

- Parameter : Treewidth
- Used by Robertson and Seymour in their work on Graph Minors
- Important in algorithm design
- Structural Parameter: measures resemblance with tree
  - #vertices needs to be deleted to get a tree
  - #edges needs to be deleted to get a tree
  - #cycles
  - Separators can be arranged in tree-like fashion
- Treewidth : Size of maximum separator is *best* tree-like arrangement of separator
- Most of NP-hard problems on general graph are polynomial time solvable on trees
- Dynamic Programming works on trees
- Dynamic Programming on sub-graphs with small boundary

Treewidth



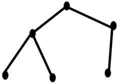
(a) A graph G.



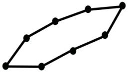
(b) A tree-decomposition of G.

Diagram from 'Metric tree-like structures in real-life networks: an empirical study' by M. Abu-Ata and F. Dragan

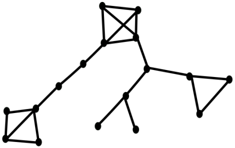
Treewidth



Trees

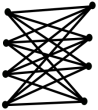


Cycles

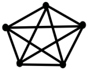


Chordal Graphs

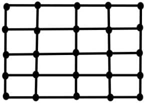
Tree like



Complete Bipartite graphs



Cliques



Grids

Not tree-like

FPT algorithms(treewidth + #colors)

Theorem (Courcelle's theorem, [Cou92])

If ϕ : a graph property that is expressible in $\mathbf{MSO}_2$ *then $\exists$ an algorithm that verifies whether ϕ is satisfied in G in $f(||\phi||, tw(G)) \cdot n$ time.*

- $\mathbf{MSO}_2$: variables for single vertices; single edges; subset of vertices; subset of edges
- f is some computable function
- $\phi(G, q)$: $\mathbf{MSO}_2$ formula which states that G has cd-chromatic number at most q.
- ($\Rightarrow$) CD-COLORING is FPT when parameterized by length of $\phi(G, q)$ plus treewidth of G

FPT algorithms(treewidth + #colors)

Theorem (Courcelle's theorem, [Cou92])

If ϕ : a graph property that is expressible in $\mathbf{MSO}_2$ then $\exists$ an algorithm that verifies whether ϕ is satisfied in G in $f(||\phi||, tw(G)) \cdot n$ time.

- $\mathbf{MSO}_2$: variables for single vertices; single edges; subset of vertices; subset of edges
- f is some computable function
- $\phi(G, q)$: $\mathbf{MSO}_2$ formula which states that G has cd-chromatic number at most q.
- ($\Rightarrow$) CD-COLORING is FPT when parameterized by length of $\phi(G, q)$ plus treewidth of G

FPT algorithms(treewidth + #colors)

Theorem (Courcelle's theorem, [Cou92])

If ϕ : a graph property that is expressible in $\mathbf{MSO}_2$ then $\exists$ an algorithm that verifies whether ϕ is satisfied in G in $f(||\phi||, tw(G)) \cdot n$ time.

- $\mathbf{MSO}_2$: variables for single vertices; single edges; subset of vertices; subset of edges
- f is some computable function
- $\phi(G, q)$: $\mathbf{MSO}_2$ formula which states that G has cd-chromatic number at most q.
- ($\Rightarrow$) CD-COLORING is FPT when parameterized by length of $\phi(G, q)$ plus treewidth of G

FPT algorithms(treewidth + #colors)

Theorem (Courcelle's theorem, [Cou92])

If ϕ : a graph property that is expressible in **$\mathbf{MSO_2}$** *then $\exists$ an algorithm that verifies whether ϕ is satisfied in G in $f(||\phi||, tw(G)) \cdot n$ time.*

- **$\mathbf{MSO_2}$** : variables for single vertices; single edges; subset of vertices; subset of edges
- f is some computable function
- $\phi(G, q)$: **$\mathbf{MSO_2}$** formula which states that G has cd-chromatic number at most q.
- ($\Rightarrow$) CD-COLORING is FPT when parameterized by length of $\phi(G, q)$ plus treewidth of G

FPT algorithms(treewidth + #colors)

Theorem (Courcelle's theorem, [Cou92])

If ϕ : a graph property that is expressible in $\mathbf{MSO}_2$ *then* $\exists$ *an algorithm that verifies whether* ϕ *is satisfied in* G *in* $f(||\phi||, tw(G)) \cdot n$ *time.*

- $\mathbf{MSO}_2$: variables for single vertices; single edges; subset of vertices; subset of edges
- f is some computable function
- $\phi(G, q)$: $\mathbf{MSO}_2$ formula which states that G has cd-chromatic number at most q.
- ($\Rightarrow$) CD-COLORING is FPT when parameterized by length of $\phi(G, q)$ plus treewidth of G

FPT algorithms(treewidth + #colors)

$\phi(G,q) \equiv$ There are q sets of $V(G)$[Which partitions $V(G)$ $\wedge$
Each of them is independent set $\wedge$
There exists a vertex dominating it]

$\phi(G,q) \equiv \exists V_1, V_2, \ldots, V_q \subseteq V(G)$ [Part$(V_1, V_2, \ldots, V_q) \wedge$
IndSet$(V_1) \wedge \cdots \wedge$ IndSet(V_q) $\wedge$
Dom$(V_1) \wedge \cdots \wedge$ Dom(V_q)]

$\phi(G,q)$ is **MSO**$_2$ formula of length $\mathcal{O}(q)$

Theorem

CD-Coloring *parameterized by the number of colors and the treewidth of the input graph is* FPT.

FPT algorithms(treewidth + #colors)

$\phi(G, q) \equiv$ There are q sets of $V(G)$[Which partitions $V(G)$ $\wedge$
Each of them is independent set $\wedge$
There exists a vertex dominating it]

$\phi(G, q) \equiv \exists V_1, V_2, \ldots, V_q \subseteq V(G)$ [Part$(V_1, V_2, \ldots, V_q) \wedge$
IndSet$(V_1) \wedge \cdots \wedge$ IndSet(V_q) $\wedge$
Dom$(V_1) \wedge \cdots \wedge$ Dom(V_q)]

$\phi(G, q)$ is **MSO**$_2$ formula of length $\mathcal{O}(q)$

Theorem

CD-COLORING *parameterized by the number of colors and the treewidth of the input graph is* FPT.

FPT algorithms(treewidth + #colors)

$$\phi(G,q) \equiv \text{There are } q \text{ sets of } V(G)[\text{Which partitions } V(G) \wedge \text{Each of them is independent set } \wedge \text{There exists a vertex dominating it}]$$

$$\phi(G,q) \equiv \exists V_1, V_2, \ldots, V_q \subseteq V(G)\ [\mathrm{Part}(V_1, V_2, \ldots, V_q) \wedge \mathrm{IndSet}(V_1) \wedge \cdots \wedge \mathrm{IndSet}(V_q) \wedge \mathrm{Dom}(V_1) \wedge \cdots \wedge \mathrm{Dom}(V_q)]$$

$\phi(G,q)$ is $\mathbf{MSO}_2$ formula of length $\mathcal{O}(q)$

Theorem

CD-COLORING *parameterized by the number of colors and the treewidth of the input graph is* FPT.

FPT algorithms(treewidth + #colors)

$\phi(G,q) \equiv$ There are q sets of $V(G)$[Which partitions $V(G)$ $\wedge$
Each of them is independent set $\wedge$
There exists a vertex dominating it]

$$\phi(G,q) \equiv \exists V_1, V_2, \ldots, V_q \subseteq V(G)\ [\mathrm{Part}(V_1, V_2, \ldots, V_q) \wedge \mathrm{IndSet}(V_1) \wedge \cdots \wedge \mathrm{IndSet}(V_q) \wedge \mathrm{Dom}(V_1) \wedge \cdots \wedge \mathrm{Dom}(V_q)]$$

$\phi(G,q)$ is **MSO$_2$** formula of length $\mathcal{O}(q)$

Theorem

CD-Coloring *parameterized by the number of colors and the treewidth of the input graph is* FPT.

FPT algorithms on class of graphs

What is the class of graphs on which it is FPT when parameterized by #colors (only)?

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs and on graphs with girth at least* 5*.*

- Chordal Graphs : Every cycle on 4 or more vertices has a chord
- Graph of girth (lenght of shortest cycle) at least 5

What is the class of graphs on which it is FPT when parameterized by #colors (only)?

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs and on graphs with girth at least* 5.

- Chordal Graphs : Every cycle on 4 or more vertices has a chord
  - COLORING is poly-time solvable but CD-COLORING is NP-Complete
  - Existance of FPT algorithm on Chordal graphs
- Graph of girth (lenght of shortest cycle) at least 5
  - COLORING is para-NP-hard [LK07]. In contrast, CD-COLORING is FPT
  - Admits an algorithm running in $\mathcal{O}(2^{\mathcal{O}(q^3)} q^{q^3} \log q^3)$ time and an $\mathcal{O}(q^3)$ sized vertex kernel on graphs with girth at least 5.

FPT algorithms on class of graphs

What is the class of graphs on which it is FPT when parameterized by #colors (only)?

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs and on graphs with girth at least* 5*.*

- Chordal Graphs : Every cycle on 4 or more vertices has a chord
  - COLORING is poly-time solvable but CD-COLORING is NP-Complete
  - Existance of FPT algorithm on Chordal graphs
- Graph of girth (lenght of shortest cycle) at least 5
  - COLORING is para-NP-hard [LK07]. In contrast, CD-COLORING is FPT
  - Admits an algorithm running in $\mathcal{O}(2^{\mathcal{O}(q^3)} q^{12} \log q^3)$ time and an $\mathcal{O}(q^3)$ sized vertex kernel on graphs with girth at least 5.

FPT algorithms on class of graphs

What is the class of graphs on which it is FPT when parameterized by #colors (only)?

Theorem

CD-Coloring parameterized by the number of colors is FPT *on chordal graphs and on graphs with girth at least* 5*.*

- Chordal Graphs : Every cycle on 4 or more vertices has a chord
  - Coloring is poly-time solvable but CD-Coloring is NP-Complete
  - Existance of FPT algorithm on Chordal graphs
- Graph of girth (lenght of shortest cycle) at least 5
  - Coloring is para-NP-hard [LK07]. In contrast, CD-Coloring is FPT
  - Admits an algorithm running in $\mathcal{O}(2^{\mathcal{O}(q^3)} q^{12} \log q^3)$ time and an $\mathcal{O}(q^3)$ sized vertex kernel on graphs with girth at least 5.

FPT algorithms on class of graphs

What is the class of graphs on which it is FPT when parameterized by #colors (only)?

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs and on graphs with girth at least* 5*.*

- Chordal Graphs : Every cycle on 4 or more vertices has a chord
  - COLORING is poly-time solvable but CD-COLORING is NP-Complete
  - Existance of FPT algorithm on Chordal graphs
- Graph of girth (lenght of shortest cycle) at least 5
  - COLORING is para-NP-hard [LK07]. In contrast, CD-COLORING is FPT
  - Admits an algorithm running in $\mathcal{O}(2^{\mathcal{O}(q^3)} q^{12} \log q^3)$ time and an $\mathcal{O}(q^3)$ sized vertex kernel on graphs with girth at least 5.

FPT algorithms on class of graphs

What is the class of graphs on which it is FPT when parameterized by #colors (only)?

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs and on graphs with girth at least* 5.

- Chordal Graphs : Every cycle on 4 or more vertices has a chord
  - COLORING is poly-time solvable but CD-COLORING is NP-Complete
  - Existance of FPT algorithm on Chordal graphs
- Graph of girth (lenght of shortest cycle) at least 5
  - COLORING is para-NP-hard [LK07]. In contrast, CD-COLORING is FPT
  - Admits an algorithm running in $\mathcal{O}(2^{\mathcal{O}(q^3)}q^{12}\log q^3)$ time and an $\mathcal{O}(q^3)$ sized vertex kernel on graphs with girth at least 5.

FPT algorithms on class of graphs

What is the class of graphs on which it is FPT when parameterized by #colors (only)?

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs and on graphs with girth at least* 5.

- Chordal Graphs : Every cycle on 4 or more vertices has a chord
  - COLORING is poly-time solvable but CD-COLORING is NP-Complete
  - Existance of FPT algorithm on Chordal graphs
- Graph of girth (lenght of shortest cycle) at least 5
  - COLORING is para-NP-hard [LK07]. In contrast, CD-COLORING is FPT
  - Admits an algorithm running in $\mathcal{O}(2^{\mathcal{O}(q^3)} q^{12} \log q^3)$ time and an $\mathcal{O}(q^3)$ sized vertex kernel on graphs with girth at least 5.

FPT algorithms on class of graphs

What is the class of graphs on which it is FPT when parameterized by #colors (only)?

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs and on graphs with girth at least* 5.

- Chordal Graphs : Every cycle on 4 or more vertices has a chord
  - COLORING is poly-time solvable but CD-COLORING is NP-Complete
  - Existance of FPT algorithm on Chordal graphs
- Graph of girth (lenght of shortest cycle) at least 5
  - COLORING is para-NP-hard [LK07]. In contrast, CD-COLORING is FPT
  - Admits an algorithm running in $\mathcal{O}(2^{\mathcal{O}(q^3)} q^{12} \log q^3)$ time and an $\mathcal{O}(q^3)$ sized vertex kernel on graphs with girth at least 5.

FPT algorithms on chordal graphs

- Let $\omega(G)$ be the size of a maximum clique in G
- For a graph G, $tw(G) \geq \omega(G) - 1$
- For a chordal graph G, $tw(G) = \omega(G) - 1$
- Finding $w(G)$ is poly-time in chordal graphs
- No two vertices in a clique can be in the same color class of cd-coloring.
  - if $\omega(G) \geq q$ then (G, q) is NO instance of CD-COLORING
  - if $\omega(G) \leq q$ then $tw(G) \leq q$

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs*

FPT algorithms on chordal graphs

- Let $\omega(G)$ be the size of a maximum clique in G
- For a graph G, $tw(G) \geq \omega(G) - 1$
- For a chordal graph G, $tw(G) = \omega(G) - 1$
- Finding $w(G)$ is poly-time in chordal graphs
- No two vertices in a clique can be in the same color class of cd-coloring.
  - if $\omega(G) \geq q$ then (G, q) is NO instance of CD-COLORING
  - if $\omega(G) \leq q$ then $tw(G) \leq q$

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs*

FPT algorithms on chordal graphs

- Let $\omega(G)$ be the size of a maximum clique in G
- For a graph G, $tw(G) \geq \omega(G) - 1$
- For a chordal graph G, $tw(G) = \omega(G) - 1$
- Finding $w(G)$ is poly-time in chordal graphs
- No two vertices in a clique can be in the same color class of cd-coloring.
  - if $\omega(G) \geq q$ then (G, q) is NO instance of CD-COLORING
  - if $\omega(G) \leq q$ then $tw(G) \leq q$

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs*

FPT algorithms on chordal graphs

- Let $\omega(G)$ be the size of a maximum clique in G
- For a graph G, $tw(G) \geq \omega(G) - 1$
- For a chordal graph G, $tw(G) = \omega(G) - 1$
- Finding $w(G)$ is poly-time in chordal graphs
- No two vertices in a clique can be in the same color class of cd-coloring.
  - if $\omega(G) \geq q$ then (G, q) is NO instance of CD-COLORING
  - if $\omega(G) \leq q$ then $tw(G) \leq q$

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs*

FPT algorithms on chordal graphs

- Let $\omega(G)$ be the size of a maximum clique in G
- For a graph G, $tw(G) \geq \omega(G) - 1$
- For a chordal graph G, $tw(G) = \omega(G) - 1$
- Finding $w(G)$ is poly-time in chordal graphs
- No two vertices in a clique can be in the same color class of cd-coloring.
  - if $\omega(G) \geq q$ then (G, q) is NO instance of CD-COLORING
  - if $\omega(G) \leq q$ then $tw(G) \leq q$

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs*

FPT algorithms on chordal graphs

- Let $\omega(G)$ be the size of a maximum clique in G
- For a graph G, $tw(G) \geq \omega(G) - 1$
- For a chordal graph G, $tw(G) = \omega(G) - 1$
- Finding $w(G)$ is poly-time in chordal graphs
- No two vertices in a clique can be in the same color class of cd-coloring.
  - if $\omega(G) \geq q$ then (G, q) is NO instance of CD-COLORING
  - if $\omega(G) \leq q$ then $tw(G) \leq q$

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs*

FPT algorithms on chordal graphs

- Let $\omega(G)$ be the size of a maximum clique in G
- For a graph G, $tw(G) \geq \omega(G) - 1$
- For a chordal graph G, $tw(G) = \omega(G) - 1$
- Finding $w(G)$ is poly-time in chordal graphs
- No two vertices in a clique can be in the same color class of cd-coloring.
  - if $\omega(G) \geq q$ then (G, q) is NO instance of CD-COLORING
  - if $\omega(G) \leq q$ then $tw(G) \leq q$

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs*

FPT algorithms on chordal graphs

- Let $\omega(G)$ be the size of a maximum clique in G
- For a graph G, $tw(G) \geq \omega(G) - 1$
- For a chordal graph G, $tw(G) = \omega(G) - 1$
- Finding $w(G)$ is poly-time in chordal graphs
- No two vertices in a clique can be in the same color class of cd-coloring.
  - if $\omega(G) \geq q$ then (G, q) is NO instance of CD-COLORING
  - if $\omega(G) \leq q$ then $tw(G) \leq q$

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs*

FPT algorithms on chordal graphs

- Let $\omega(G)$ be the size of a maximum clique in G
- For a graph G, $tw(G) \geq \omega(G) - 1$
- For a chordal graph G, $tw(G) = \omega(G) - 1$
- Finding $w(G)$ is poly-time in chordal graphs
- No two vertices in a clique can be in the same color class of cd-coloring.
  - if $\omega(G) \geq q$ then (G, q) is NO instance of CD-COLORING
  - if $\omega(G) \leq q$ then $tw(G) \leq q$

Theorem

CD-COLORING *parameterized by the number of colors is* FPT *on chordal graphs*

- If $G_1, \ldots, G_l$ are the connected components of G, then $\chi_{cd}(G) = \sum_{i=1}^{l} \chi_{cd}(G_i)$.
- wlog assume that input graph is connected
- Let $\chi_{cd}(G)$ be the minimum number of colors in any cd-coloring of graph G
- Dominating Set : $S \subseteq V(G)$ s.t. $V(G) = \bigcup_{v \in S} N[v]$
- Total Dominating Set : S s.t. $V(G) = \bigcup_{v \in S} N(v)$

- If $G_1, \ldots, G_l$ are the connected components of G, then $\chi_{cd}(G) = \sum_{i=1}^{l} \chi_{cd}(G_i)$.
- wlog assume that input graph is connected
- Let $\chi_{cd}(G)$ be the minimum number of colors in any cd-coloring of graph G
- Dominating Set : $S \subseteq V(G)$ s.t. $V(G) = \bigcup_{v \in S} N[v]$
- Total Dominating Set : S s.t. $V(G) = \bigcup_{v \in S} N(v)$

- If $G_1, \ldots, G_l$ are the connected components of G, then $\chi_{cd}(G) = \sum_{i=1}^{l} \chi_{cd}(G_i)$.
- wlog assume that input graph is connected
- Let $\chi_{cd}(G)$ be the minimum number of colors in any cd-coloring of graph G
- Dominating Set : $S \subseteq V(G)$ s.t. $V(G) = \bigcup_{v \in S} N[v]$
- Total Dominating Set : S s.t. $V(G) = \bigcup_{v \in S} N(v)$

- If $G_1, \ldots, G_l$ are the connected components of G, then $\chi_{cd}(G) = \sum_{i=1}^{l} \chi_{cd}(G_i)$.
- wlog assume that input graph is connected
- Let $\chi_{cd}(G)$ be the minimum number of colors in any cd-coloring of graph G
- Dominating Set : $S \subseteq V(G)$ s.t. $V(G) = \bigcup_{v \in S} N[v]$
- Total Dominating Set : S s.t. $V(G) = \bigcup_{v \in S} N(v)$

- If $G_1, \ldots, G_l$ are the connected components of G, then $\chi_{cd}(G) = \sum_{i=1}^{l} \chi_{cd}(G_i)$.
- wlog assume that input graph is connected
- Let $\chi_{cd}(G)$ be the minimum number of colors in any cd-coloring of graph G
- Dominating Set : $S \subseteq V(G)$ s.t. $V(G) = \bigcup_{v \in S} N[v]$
- Total Dominating Set : S s.t. $V(G) = \bigcup_{v \in S} N(v)$

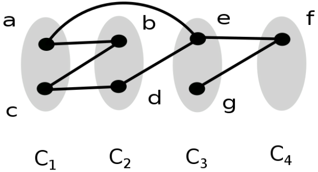
- If $G_1, \ldots, G_l$ are the connected components of G, then $\chi_{cd}(G) = \sum_{i=1}^{l} \chi_{cd}(G_i)$.
- wlog assume that input graph is connected
- Let $\chi_{cd}(G)$ be the minimum number of colors in any cd-coloring of graph G
- Dominating Set : $S \subseteq V(G)$ s.t. $V(G) = \bigcup_{v \in S} N[v]$
- Total Dominating Set : S s.t. $V(G) = \bigcup_{v \in S} N(v)$

cd-Coloring **Parameter:** q
Input: A graph G, integer q
Question: Can graph G be partitioned into q independent sets such that every independent set is contained in closed neighbourhood of some vertex?

- If $G_1, \ldots, G_l$ are the connected components of G, then $\chi_{cd}(G) = \sum_{i=1}^{l} \chi_{cd}(G_i)$.
- wlog assume that input graph is connected
- Let $\chi_{cd}(G)$ be the minimum number of colors in any cd-coloring of graph G
- Dominating Set : $S \subseteq V(G)$ s.t. $V(G) = \bigcup_{v \in S} N[v]$
- Total Dominating Set : S s.t. $V(G) = \bigcup_{v \in S} N(v)$

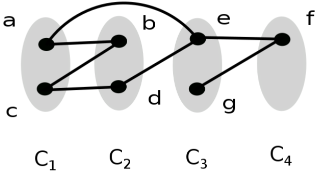
CD-COLORING **Parameter:** q
Input: A connected graph G, integer q
Question: Can graph G be partitioned into q independent sets such that every independent set is contained in open neighbourhood of some vertex?

FPT algorithms and Kernalization



Dominating Set $= \{b, c, f\}$
Total Dominating Set $= \{b, c, f, e\}$
which dominates color classes C_1, C_2, C_3, C_4 resp.

FPT algorithms and Kernalization



Dominating Set $= \{b, c, f\}$
Total Dominating Set $= \{b, c, f, e\}$
which dominates color classes C_1, C_2, C_3, C_4 resp.

FPT algorithm for $\mathcal{G}_5$

- $\mathcal{G}_5 = \{G \mid \text{girth of } G \geq 5\}$
- For a graph $G \in \mathcal{G}_5$, for any u, v: $|N(v) \cap N(u)| \leq 1$.
- In any cd-coloring of G, every color class has unique vertex which dominates it
- For general graphs : $minTDS(G) \leq \chi_{cd}(G)$
- For graph $G \in \mathcal{G}_5$: $minTDS(G) = \chi_{cd}(G)$
- CD-COLORING(G, q) $\Leftrightarrow$ TOTAL DOMINATING SET(G, q) for $G \in \mathcal{G}_5$

FPT algorithm for $\mathcal{G}_5$

- $\mathcal{G}_5 = \{G |$ girth of $G \geq 5\}$
- For a graph $G \in \mathcal{G}_5$, for any u, v: $|N(v) \cap N(u)| \leq 1$.
- In any cd-coloring of G, every color class has unique vertex which dominates it
- For general graphs : $minTDS(G) \leq \chi_{cd}(G)$
- For graph $G \in \mathcal{G}_5$: $minTDS(G) = \chi_{cd}(G)$
- CD-COLORING$(G, q) \Leftrightarrow$ TOTAL DOMINATING SET(G, q) for $G \in \mathcal{G}_5$

FPT algorithm for $\mathcal{G}_5$

- $\mathcal{G}_5 = \{G \mid \text{girth of } G \geq 5\}$
- For a graph $G \in \mathcal{G}_5$, for any u, v: $|N(v) \cap N(u)| \leq 1$.
- In any cd-coloring of G, every color class has unique vertex which dominates it
- For general graphs : $minTDS(G) \leq \chi_{cd}(G)$
- For graph $G \in \mathcal{G}_5$: $minTDS(G) = \chi_{cd}(G)$
- CD-COLORING$(G, q) \Leftrightarrow$ TOTAL DOMINATING SET(G, q) for $G \in \mathcal{G}_5$

FPT algorithm for $\mathcal{G}_5$

- $\mathcal{G}_5 = \{G \mid \text{girth of } G \geq 5\}$
- For a graph $G \in \mathcal{G}_5$, for any u, v: $|N(v) \cap N(u)| \leq 1$.
- In any cd-coloring of G, every color class has unique vertex which dominates it
- For general graphs : $minTDS(G) \leq \chi_{cd}(G)$
- For graph $G \in \mathcal{G}_5$: $minTDS(G) = \chi_{cd}(G)$
- CD-COLORING$(G, q) \Leftrightarrow$ TOTAL DOMINATING SET(G, q) for $G \in \mathcal{G}_5$

FPT algorithm for $\mathcal{G}_5$

- $\mathcal{G}_5 = \{G \mid \text{girth of } G \geq 5\}$
- For a graph $G \in \mathcal{G}_5$, for any u, v: $|N(v) \cap N(u)| \leq 1$.
- In any cd-coloring of G, every color class has unique vertex which dominates it
- For general graphs : $minTDS(G) \leq \chi_{cd}(G)$
- For graph $G \in \mathcal{G}_5$: $minTDS(G) = \chi_{cd}(G)$
- CD-COLORING(G, q) $\Leftrightarrow$ TOTAL DOMINATING SET(G, q) for $G \in \mathcal{G}_5$

FPT algorithm for $\mathcal{G}_5$

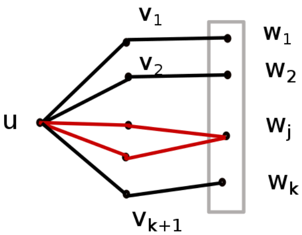
- $\mathcal{G}_5 = \{G \mid \text{girth of } G \geq 5\}$
- For a graph $G \in \mathcal{G}_5$, for any u, v: $|N(v) \cap N(u)| \leq 1$.
- In any cd-coloring of G, every color class has unique vertex which dominates it
- For general graphs : $minTDS(G) \leq \chi_{cd}(G)$
- For graph $G \in \mathcal{G}_5$: $minTDS(G) = \chi_{cd}(G)$
- $\textsc{cd-Coloring}(G, q) \Leftrightarrow \textsc{Total Dominating Set}(G, q)$ for $G \in \mathcal{G}_5$

Kernalization for Total-Dom-Set

Claim

For $G \in \mathcal{G}_5$, if $deg(u) \geq k+1$, then any total dominating set of size at most k contains u.

- Consider Total Dominating Set $\{w_1, w_2, \ldots, w_k\}$ which doesn't contain u
- Note : v_i may be equal to w_j
- By Pigeon-hole principle, there exists some w_j which is adjacent to two vertices in $N(u)$
- This contradicts the fact that $G \in \mathcal{G}_5$

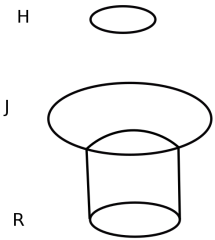


Kernalization for Total-Dom-Set

Partition graph into 3 parts: High degree vertices which will be part of any solution (H), vertices which have been dominated by partial solution (J) and rest of the graph(R)

Kernalization for Total-Dom-Set

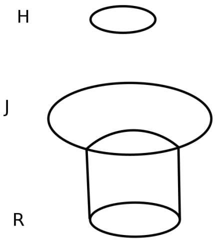
Partition graph into 3 parts: High degree vertices which will be part of any solution (H), vertices which have been dominated by partial solution (J) and rest of the graph(R)



- $H = \{u | \ deg(u) \geq k + 1\}$
  $J = N[H] \setminus H$
  $R = V(G) \setminus (H \cup J)$

Kernalization for Total-Dom-Set

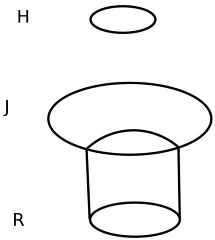
Partition graph into 3 parts: High degree vertices which will be part of any solution (H), vertices which have been dominated by partial solution (J) and rest of the graph(R)



- $H = \{u | \; deg(u) \geq k+1\}$
  $J = N[H] \setminus H$
  $R = V(G) \setminus (H \cup J)$
- H is contained in solution
  $\Rightarrow |H| \leq k$

Kernalization for Total-Dom-Set

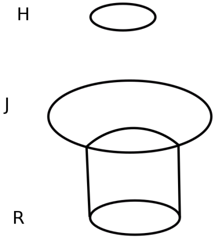
Partition graph into 3 parts: High degree vertices which will be part of any solution (H), vertices which have been dominated by partial solution (J) and rest of the graph(R)



- $H = \{u | \ deg(u) \geq k + 1\}$
  $J = N[H] \setminus H$
  $R = V(G) \setminus (H \cup J)$
- H is contained in solution
  $\Rightarrow |H| \leq k$
- Vertices in R can't be dominated by vertices in H and every vertex in $J \cup R$ has degree at most k,
  $\Rightarrow |R| \leq \mathcal{O}(k^2)$

Kernalization for Total-Dom-Set

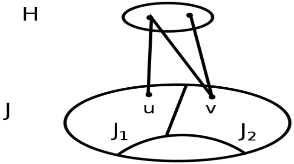
Partition graph into 3 parts: High degree vertices which will be part of any solution (H), vertices which have been dominated by partial solution (J) and rest of the graph(R)



- $H = \{u \mid deg(u) \geq k+1\}$
  $J = N[H] \setminus H$
  $R = V(G) \setminus (H \cup J)$
- H is contained in solution
  $\Rightarrow |H| \leq k$
- Vertices in R can't be dominated by vertices in H and every vertex in $J \cup R$ has degree at most k,
  $\Rightarrow |R| \leq \mathcal{O}(k^2)$
- $|J \cup R| \leq |R| * k \leq \mathcal{O}(k^3)$

Reduction Rule

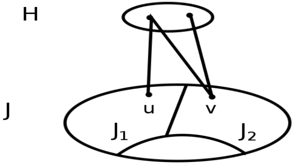
For $u, v \in J \setminus N(R)$, if $N(u) \cap H \subseteq N(v) \cap H$ then delete u.



Reduction Rule

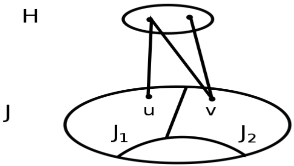
For $u, v \in J \setminus N(R)$, if $N(u) \cap H \subseteq N(v) \cap H$ then delete u.

Apply reduction rule exhaustively



Reduction Rule

For $u, v \in J \setminus N(R)$, if $N(u) \cap H \subseteq N(v) \cap H$ then delete u.



Apply reduction rule exhaustively

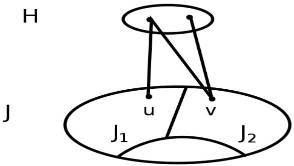
Partition $J \setminus N(R)$ into J_1 and J_2

$J_1 = \{u |$ s.t. $|N(u) \cap H| = 1\}$

$J_2 = \{u |$ s.t. $|N(u) \cap H| \geq 2\}$

Reduction Rule

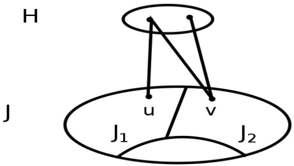
For $u, v \in J \setminus N(R)$, if $N(u) \cap H \subseteq N(v) \cap H$ then delete u.



Apply reduction rule exhaustively
Partition $J \setminus N(R)$ into J_1 and J_2
$J_1 = \{u | \text{ s.t. } |N(u) \cap H| = 1\}$
$J_2 = \{u | \text{ s.t. } |N(u) \cap H| \geq 2\}$
$|J_1| \leq |H| \leq k$
(Otherwise reduction rule will be applicable)

Reduction Rule

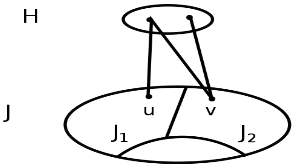
For $u, v \in J \setminus N(R)$, if $N(u) \cap H \subseteq N(v) \cap H$ then delete u.



Apply reduction rule exhaustively
Partition $J \setminus N(R)$ into J_1 and J_2
$J_1 = \{u| \text{ s.t. } |N(u) \cap H| = 1\}$
$J_2 = \{u| \text{ s.t. } |N(u) \cap H| \geq 2\}$
$|J_1| \leq |H| \leq k$
(Otherwise reduction rule will be applicable)
$|J_2| \leq \binom{k}{2} \leq \mathcal{O}(k^2)$
(If $|J_2|$ is larger we will find cycle of lenght 4)

Reduction Rule

For $u, v \in J \setminus N(R)$, if $N(u) \cap H \subseteq N(v) \cap H$ then delete u.



Apply reduction rule exhaustively
Partition $J \setminus N(R)$ into J_1 and J_2
$J_1 = \{u |$ s.t. $|N(u) \cap H| = 1\}$
$J_2 = \{u |$ s.t. $|N(u) \cap H| \geq 2\}$
$|J_1| \leq |H| \leq k$
(Otherwise reduction rule will be applicable)
$|J_2| \leq \binom{k}{2} \leq \mathcal{O}(k^2)$
(If $|J_2|$ is larger we will find cycle of lenght 4)

Lemma

Total Dominating Set *admits a kernel of $\mathcal{O}(k^3)$ vertices on $\mathcal{G}_5$.*

Exact algorithm to compute cd-chromatic number

Let $\mathcal{I}$ be the set containing all independent sets in G

$$\mathcal{I}^1 = \{X \mid X \in \mathcal{I} \text{ and } \exists u \in V(G) \text{ s.t. } X \subseteq N(u)\}$$

$\mathcal{I}^1$: Possible candidates for color classes in cd-coloring of graph

$$\mathcal{I}^2 = \{Y \mid \exists X_1, X_2 \in \mathcal{I}^1 \text{ s.t. } X_1 \cup X_2 = Y \text{ and } X_1 \cap X_2 = \emptyset\}$$

$\mathcal{I}^2$: Possible subgraphs which will need 2 color classes in some cd-coloring of graph

$$\mathcal{I}^3 = \{Z \mid \exists X \in \mathcal{I}^1, Y \in \mathcal{I}^2 \text{ s.t } X \cup Y = Z \text{ and } X \cap Y = \emptyset\}$$

$\mathcal{I}^3$: Possible subgraphs which will need 3 color classes in some cd-coloring of graph

To compute $\chi_{cd}(G)$, we need to find minimum q such that $V(G) \in \mathcal{I}^q$

Exact algorithm to compute cd-chromatic number

Let $\mathcal{I}$ be the set containing all independent sets in G

$$\mathcal{I}^1 = \{X \mid X \in \mathcal{I} \text{ and } \exists u \in V(G) \text{ s.t. } X \subseteq N(u)\}$$

$\mathcal{I}^1$: Possible candidates for color classes in cd-coloring of graph

$$\mathcal{I}^2 = \{Y \mid \exists X_1, X_2 \in \mathcal{I}^1 \text{ s.t. } X_1 \cup X_2 = Y \text{ and } X_1 \cap X_2 = \emptyset\}$$

$\mathcal{I}^2$: Possible subgraphs which will need 2 color classes in some cd-coloring of graph

$$\mathcal{I}^3 = \{Z \mid \exists X \in \mathcal{I}^1, Y \in \mathcal{I}^2 \text{ s.t } X \cup Y = Z \text{ and } X \cap Y = \emptyset\}$$

$\mathcal{I}^3$: Possible subgraphs which will need 3 color classes in some cd-coloring of graph

To compute $\chi_{cd}(G)$, we need to find minimum q such that $V(G) \in \mathcal{I}^q$

Exact algorithm to compute cd-chromatic number

Let $\mathcal{I}$ be the set containing all independent sets in G

$$\mathcal{I}^1 = \{X \mid X \in \mathcal{I} \text{ and } \exists u \in V(G) \text{ s.t. } X \subseteq N(u)\}$$

$\mathcal{I}^1$: Possible candidates for color classes in cd-coloring of graph

$$\mathcal{I}^2 = \{Y \mid \exists X_1, X_2 \in \mathcal{I}^1 \text{ s.t. } X_1 \cup X_2 = Y \text{ and } X_1 \cap X_2 = \emptyset\}$$

$\mathcal{I}^2$: Possible subgraphs which will need 2 color classes in some cd-coloring of graph

$$\mathcal{I}^3 = \{Z \mid \exists X \in \mathcal{I}^1, Y \in \mathcal{I}^2 \text{ s.t } X \cup Y = Z \text{ and } X \cap Y = \emptyset\}$$

$\mathcal{I}^3$: Possible subgraphs which will need 3 color classes in some cd-coloring of graph

To compute $\chi_{cd}(G)$, we need to find minimum q such that $V(G) \in \mathcal{I}^q$

Exact algorithm to compute cd-chromatic number

Let $\mathcal{I}$ be the set containing all independent sets in G

$$\mathcal{I}^1 = \{X \mid X \in \mathcal{I} \text{ and } \exists u \in V(G) \text{ s.t. } X \subseteq N(u)\}$$

$\mathcal{I}^1$: Possible candidates for color classes in cd-coloring of graph

$$\mathcal{I}^2 = \{Y \mid \exists X_1, X_2 \in \mathcal{I}^1 \text{ s.t. } X_1 \cup X_2 = Y \text{ and } X_1 \cap X_2 = \emptyset\}$$

$\mathcal{I}^2$: Possible subgraphs which will need 2 color classes in some cd-coloring of graph

$$\mathcal{I}^3 = \{Z \mid \exists X \in \mathcal{I}^1, Y \in \mathcal{I}^2 \text{ s.t } X \cup Y = Z \text{ and } X \cap Y = \emptyset\}$$

$\mathcal{I}^3$: Possible subgraphs which will need 3 color classes in some cd-coloring of graph

To compute $\chi_{cd}(G)$, we need to find minimum q such that $V(G) \in \mathcal{I}^q$

Exact algorithm to compute cd-chromatic number

Let $\mathcal{I}$ be the set containing all independent sets in G

$$\mathcal{I}^1 = \{X \mid X \in \mathcal{I} \text{ and } \exists u \in V(G) \text{ s.t. } X \subseteq N(u)\}$$

$\mathcal{I}^1$: Possible candidates for color classes in cd-coloring of graph

$$\mathcal{I}^2 = \{Y \mid \exists X_1, X_2 \in \mathcal{I}^1 \text{ s.t. } X_1 \cup X_2 = Y \text{ and } X_1 \cap X_2 = \emptyset\}$$

$\mathcal{I}^2$: Possible subgraphs which will need 2 color classes in some cd-coloring of graph

$$\mathcal{I}^3 = \{Z \mid \exists X \in \mathcal{I}^1, Y \in \mathcal{I}^2 \text{ s.t } X \cup Y = Z \text{ and } X \cap Y = \emptyset\}$$

$\mathcal{I}^3$: Possible subgraphs which will need 3 color classes in some cd-coloring of graph

To compute $\chi_{cd}(G)$, we need to find minimum q such that $V(G) \in \mathcal{I}^q$

Exact algorithm to compute cd-chromatic number

Let $\mathcal{I}$ be the set containing all independent sets in G

$$\mathcal{I}^1 = \{X \mid X \in \mathcal{I} \text{ and } \exists u \in V(G) \text{ s.t. } X \subseteq N(u)\}$$

$\mathcal{I}^1$: Possible candidates for color classes in cd-coloring of graph

$$\mathcal{I}^2 = \{Y \mid \exists X_1, X_2 \in \mathcal{I}^1 \text{ s.t. } X_1 \cup X_2 = Y \text{ and } X_1 \cap X_2 = \emptyset\}$$

$\mathcal{I}^2$: Possible subgraphs which will need 2 color classes in some cd-coloring of graph

$$\mathcal{I}^3 = \{Z \mid \exists X \in \mathcal{I}^1, Y \in \mathcal{I}^2 \text{ s.t } X \cup Y = Z \text{ and } X \cap Y = \emptyset\}$$

$\mathcal{I}^3$: Possible subgraphs which will need 3 color classes in some cd-coloring of graph

To compute $\chi_{cd}(G)$, we need to find minimum q such that $V(G) \in \mathcal{I}^q$

Exact algorithm to compute cd-chromatic number

Let $\mathcal{I}$ be the set containing all independent sets in G

$$\mathcal{I}^1 = \{X \mid X \in \mathcal{I} \text{ and } \exists u \in V(G) \text{ s.t. } X \subseteq N(u)\}$$

$\mathcal{I}^1$: Possible candidates for color classes in cd-coloring of graph

$$\mathcal{I}^2 = \{Y \mid \exists X_1, X_2 \in \mathcal{I}^1 \text{ s.t. } X_1 \cup X_2 = Y \text{ and } X_1 \cap X_2 = \emptyset\}$$

$\mathcal{I}^2$: Possible subgraphs which will need 2 color classes in some cd-coloring of graph

$$\mathcal{I}^3 = \{Z \mid \exists X \in \mathcal{I}^1, Y \in \mathcal{I}^2 \text{ s.t } X \cup Y = Z \text{ and } X \cap Y = \emptyset\}$$

$\mathcal{I}^3$: Possible subgraphs which will need 3 color classes in some cd-coloring of graph

To compute $\chi_{cd}(G)$, we need to find minimum q such that $V(G) \in \mathcal{I}^q$

Exact algorithm to compute cd-chromatic number

Let $\mathcal{I}$ be the set containing all independent sets in G

$$\mathcal{I}^1 = \{X \mid X \in \mathcal{I} \text{ and } \exists u \in V(G) \text{ s.t. } X \subseteq N(u)\}$$

$\mathcal{I}^1$: Possible candidates for color classes in cd-coloring of graph

$$\mathcal{I}^2 = \{Y \mid \exists X_1, X_2 \in \mathcal{I}^1 \text{ s.t. } X_1 \cup X_2 = Y \text{ and } X_1 \cap X_2 = \emptyset\}$$

$\mathcal{I}^2$: Possible subgraphs which will need 2 color classes in some cd-coloring of graph

$$\mathcal{I}^3 = \{Z \mid \exists X \in \mathcal{I}^1, Y \in \mathcal{I}^2 \text{ s.t } X \cup Y = Z \text{ and } X \cap Y = \emptyset\}$$

$\mathcal{I}^3$: Possible subgraphs which will need 3 color classes in some cd-coloring of graph

To compute $\chi_{cd}(G)$, we need to find minimum q such that $V(G) \in \mathcal{I}^q$

Set and its representation

Universe $U = \{u_1, u_2, \ldots, u_n\}$ with fixed ordering on elements.

Characteristic vector of a set $S \subseteq U$ is bit vector $\psi(S)$, of length n s.t. $\psi(S)[j] = 1$ iff $u_j \in S$

Hamming weight of a vector ϕ is the number of 1s in ϕ and its denoted by $\mathcal{H}(\phi)$

val(ϕ) denotes the integer d of which ϕ is the binary representation.

Let z be an indeterminate variable. For $S_1, S_2 \subseteq U$, define modefied multiplication $(\star)$

$$z^{val(\psi(S_1))} \star z^{val(\psi(S_2))} = \begin{cases} z^{val(\psi(S_1))+val(\psi(S_2))} & \text{if } S_1 \cap S_2 = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Objective : To compute $z^{val(\psi(S_1))} \star z^{val(\psi(S_2))}$ without explicitely looking at S_1, S_2

Set and its representation

Universe $U = \{u_1, u_2, \ldots, u_n\}$ with fixed ordering on elements.
Characteristic vector of a set $S \subseteq U$ is bit vector $\psi(S)$, of length n s.t. $\psi(S)[j] = 1$ iff $u_j \in S$
Hamming weight of a vector ϕ is the number of 1s in ϕ and its denoted by $\mathcal{H}(\phi)$
val(ϕ) denotes the integer d of which ϕ is the binary representation.
Let z be an indeterminate variable. For $S_1, S_2 \subseteq U$, define modefied multiplication ($\star$)

$$z^{val(\psi(S_1))} \star z^{val(\psi(S_2))} = \begin{cases} z^{val(\psi(S_1))+val(\psi(S_2))} & \text{if } S_1 \cap S_2 = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Objective : To compute $z^{val(\psi(S_1))} \star z^{val(\psi(S_2))}$ without explicitely looking at S_1, S_2

Set and its representation

Universe $U = \{u_1, u_2, \ldots, u_n\}$ with fixed ordering on elements.
Characteristic vector of a set $S \subseteq U$ is bit vector $\psi(S)$, of length n s.t. $\psi(S)[j] = 1$ iff $u_j \in S$
Hamming weight of a vector ϕ is the number of 1s in ϕ and its denoted by $\mathcal{H}(\phi)$
val(ϕ) denotes the integer d of which ϕ is the binary representation.
Let z be an indeterminate variable. For $S_1, S_2 \subseteq U$, define modefied multiplication $(\star)$

$$z^{val(\psi(S_1))} \star z^{val(\psi(S_2))} = \begin{cases} z^{val(\psi(S_1))+val(\psi(S_2))} & \text{if } S_1 \cap S_2 = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Objective : To compute $z^{val(\psi(S_1))} \star z^{val(\psi(S_2))}$ without explicitely looking at S_1, S_2

Set and its representation

Universe $U = \{u_1, u_2, \ldots, u_n\}$ with fixed ordering on elements.
Characteristic vector of a set $S \subseteq U$ is bit vector $\psi(S)$, of length n s.t. $\psi(S)[j] = 1$ iff $u_j \in S$
Hamming weight of a vector ϕ is the number of 1s in ϕ and its denoted by $\mathcal{H}(\phi)$
val(ϕ) denotes the integer d of which ϕ is the binary representation.
Let z be an indeterminate variable. For $S_1, S_2 \subseteq U$, define modefied multiplication $(\star)$

$$z^{val(\psi(S_1))} \star z^{val(\psi(S_2))} = \begin{cases} z^{val(\psi(S_1))+val(\psi(S_2))} & \text{if } S_1 \cap S_2 = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Objective : To compute $z^{val(\psi(S_1))} \star z^{val(\psi(S_2))}$ without explicitely looking at S_1, S_2

Set and its representation

Universe $U = \{u_1, u_2, \ldots, u_n\}$ with fixed ordering on elements.
Characteristic vector of a set $S \subseteq U$ is bit vector $\psi(S)$, of length n s.t. $\psi(S)[j] = 1$ iff $u_j \in S$
Hamming weight of a vector ϕ is the number of 1s in ϕ and its denoted by $\mathcal{H}(\phi)$
val(ϕ) denotes the integer d of which ϕ is the binary representation.
Let z be an indeterminate variable. For $S_1, S_2 \subseteq U$, define modefied multiplication $(\star)$

$$z^{val(\psi(S_1))} \star z^{val(\psi(S_2))} = \begin{cases} z^{val(\psi(S_1))+val(\psi(S_2))} & \text{if } S_1 \cap S_2 = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Objective : To compute $z^{val(\psi(S_1))} \star z^{val(\psi(S_2))}$ without explicitely looking at S_1, S_2

Set and its representation

Universe $U = \{u_1, u_2, \ldots, u_n\}$ with fixed ordering on elements.
Characteristic vector of a set $S \subseteq U$ is bit vector $\psi(S)$, of length n s.t. $\psi(S)[j] = 1$ iff $u_j \in S$
Hamming weight of a vector ϕ is the number of 1s in ϕ and its denoted by $\mathcal{H}(\phi)$
val(ϕ) denotes the integer d of which ϕ is the binary representation.
Let z be an indeterminate variable. For $S_1, S_2 \subseteq U$, define modefied multiplication $(\star)$

$$z^{val(\psi(S_1))} \star z^{val(\psi(S_2))} = \begin{cases} z^{val(\psi(S_1))+val(\psi(S_2))} & \text{if } S_1 \cap S_2 = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Objective : To compute $z^{val(\psi(S_1))} \star z^{val(\psi(S_2))}$ without explicitely looking at S_1, S_2

Set and its representation

Universe $U = \{a, b, c, d\}$

Sets	Char. Vector	Ham-Wt
$S_1 = \{a, c\}$	$B_1 = 1010$	2
$S_2 = \{b\}$	$B_2 = 0100$	1
$S_3 = \{c, d\}$	$B_3 = 0011$	2
$S_1 \cup S_2 = \{a, b, c\}$	$B_{12} = 1110$	3
$S_1 \cup S_3 = \{a, c, d\}$	$B_{13} = 1011$	3

$S_1 \cap S_2 = \emptyset$ and $\mathcal{H}(B_{12}) = \mathcal{H}(B_1) + \mathcal{H}(B_2)$
$S_1 \cap S_3 \neq \emptyset$ and $\mathcal{H}(B_{13}) \neq \mathcal{H}(B_1) + \mathcal{H}(B_2)$

1-bit is *lost* while adding two bit-vectors B and B' if there is an index $i \in [n]$ such that $B[i] = B'[i] = 1$ i.e. while adding two bit-vectors corresponds to sets which are not disjoint

Set and its representation

Universe $U = \{a, b, c, d\}$

Sets	Char. Vector	Ham-Wt
$S_1 = \{a, c\}$	$B_1 = 1010$	2
$S_2 = \{b\}$	$B_2 = 0100$	1
$S_3 = \{c, d\}$	$B_3 = 0011$	2
$S_1 \cup S_2 = \{a, b, c\}$	$B_{12} = 1110$	3
$S_1 \cup S_3 = \{a, c, d\}$	$B_{13} = 1011$	3

$S_1 \cap S_2 = \emptyset$ and $\mathcal{H}(B_{12}) = \mathcal{H}(B_1) + \mathcal{H}(B_2)$
$S_1 \cap S_3 \neq \emptyset$ and $\mathcal{H}(B_{13}) \neq \mathcal{H}(B_1) + \mathcal{H}(B_2)$

1-bit is *lost* while adding two bit-vectors B and B' if there is an index $i \in [n]$ such that $B[i] = B'[i] = 1$ i.e. while adding two bit-vectors corresponds to sets which are not disjoint

Set and its representation

Universe $U = \{a, b, c, d\}$

Sets	Char. Vector	Ham-Wt
$S_1 = \{a, c\}$	$B_1 = 1010$	2
$S_2 = \{b\}$	$B_2 = 0100$	1
$S_3 = \{c, d\}$	$B_3 = 0011$	2
$S_1 \cup S_2 = \{a, b, c\}$	$B_{12} = 1110$	3
$S_1 \cup S_3 = \{a, c, d\}$	$B_{13} = 1011$	3

$S_1 \cap S_2 = \emptyset$ and $\mathcal{H}(B_{12}) = \mathcal{H}(B_1) + \mathcal{H}(B_2)$
$S_1 \cap S_3 \neq \emptyset$ and $\mathcal{H}(B_{13}) \neq \mathcal{H}(B_1) + \mathcal{H}(B_2)$

1-bit is *lost* while adding two bit-vectors B and B' if there is an index $i \in [n]$ such that $B[i] = B'[i] = 1$ i.e. while adding two bit-vectors corresponds to sets which are not disjoint

Set and its representation

Universe $U = \{a, b, c, d\}$

Sets	Char. Vector	Ham-Wt
$S_1 = \{a, c\}$	$B_1 = 1010$	2
$S_2 = \{b\}$	$B_2 = 0100$	1
$S_3 = \{c, d\}$	$B_3 = 0011$	2
$S_1 \cup S_2 = \{a, b, c\}$	$B_{12} = 1110$	3
$S_1 \cup S_3 = \{a, c, d\}$	$B_{13} = 1011$	3

$$S_1 \cap S_2 = \emptyset \text{ and } \mathcal{H}(B_{12}) = \mathcal{H}(B_1) + \mathcal{H}(B_2)$$
$$S_1 \cap S_3 \neq \emptyset \text{ and } \mathcal{H}(B_{13}) \neq \mathcal{H}(B_1) + \mathcal{H}(B_2)$$

1-bit is *lost* while adding two bit-vectors B and B' if there is an index $i \in [n]$ such that $B[i] = B'[i] = 1$ i.e. while adding two bit-vectors corresponds to sets which are not disjoint

Set and its representation

Universe $U = \{a, b, c, d\}$

Sets	Char. Vector	Ham-Wt
$S_1 = \{a, c\}$	$B_1 = 1010$	2
$S_2 = \{b\}$	$B_2 = 0100$	1
$S_3 = \{c, d\}$	$B_3 = 0011$	2
$S_1 \cup S_2 = \{a, b, c\}$	$B_{12} = 1110$	3
$S_1 \cup S_3 = \{a, c, d\}$	$B_{13} = 1011$	3

$$S_1 \cap S_2 = \emptyset \text{ and } \mathcal{H}(B_{12}) = \mathcal{H}(B_1) + \mathcal{H}(B_2)$$
$$S_1 \cap S_3 \neq \emptyset \text{ and } \mathcal{H}(B_{13}) \neq \mathcal{H}(B_1) + \mathcal{H}(B_2)$$

1-bit is *lost* while adding two bit-vectors B and B' if there is an index $i \in [n]$ such that $B[i] = B'[i] = 1$ i.e. while adding two bit-vectors corresponds to sets which are not disjoint

Set and its representation

Universe $U = \{a, b, c, d\}$

Sets	Char. Vector	Ham-Wt
$S_1 = \{a, c\}$	$B_1 = 1010$	2
$S_2 = \{b\}$	$B_2 = 0100$	1
$S_3 = \{c, d\}$	$B_3 = 0011$	2
$S_1 \cup S_2 = \{a, b, c\}$	$B_{12} = 1110$	3
$S_1 \cup S_3 = \{a, c, d\}$	$B_{13} = 1011$	3

$$S_1 \cap S_2 = \emptyset \text{ and } \mathcal{H}(B_{12}) = \mathcal{H}(B_1) + \mathcal{H}(B_2)$$
$$S_1 \cap S_3 \neq \emptyset \text{ and } \mathcal{H}(B_{13}) \neq \mathcal{H}(B_1) + \mathcal{H}(B_2)$$

1-bit is *lost* while adding two bit-vectors B and B' if there is an index $i \in [n]$ such that $B[i] = B'[i] = 1$ i.e. while adding two bit-vectors corresponds to sets which are not disjoint

Algorithm to compute $(\star)$

Algorithm to compute modified multiplication $(\star)$

Input: Two polynomials $p(z), r(z)$
Output: $p(z) \star r(z)$

Initialize polynomial $t(z)$ to 0
for *each ordered pair* (i, j) *such that* $i + j \leq n$ **do**
    $s_i(z) \leftarrow$ monomials in $p(z)$ whose exponent as Ham-Wt i
    $s_j(z) \leftarrow$ monomials in $r(z)$ whose exponent as Ham-Wt j
    Compute $s_{ij}(z) = s_i(z) * s_j(z)$ (standard multiplication)
    $t(z) \leftarrow t(z)+$ monomials in $s_{ij}(z)$ whose exponent has Ham-Wt $i + j$
return $t(z)$

Running time : $\mathcal{O}(n^2 \times d^2)$ where d is degree of polynomial

Algorithm to compute $(\star)$

Algorithm to compute modified multiplication $(\star)$

Input: Two polynomials $p(z), r(z)$
Output: $p(z) \star r(z)$

Initialize polynomial $t(z)$ to 0
for *each ordered pair* (i, j) *such that* $i + j \leq n$ **do**
    $s_i(z) \leftarrow$ monomials in $p(z)$ whose exponent as Ham-Wt i
    $s_j(z) \leftarrow$ monomials in $r(z)$ whose exponent as Ham-Wt j
    Compute $s_{ij}(z) = s_i(z) * s_j(z)$ (standard multiplication)
    $t(z) \leftarrow t(z)+$ monomials in $s_{ij}(z)$ whose exponent has Ham-Wt $i + j$
return $t(z)$

Running time : $\mathcal{O}(n^2 \times d^2)$ where d is degree of polynomial

Algorithm to compute ($\star$)

Algorithm to compute modified multiplication ($\star$)

Input: Two polynomials $p(z), r(z)$

Output: $p(z) \star r(z)$

```
Initialize polynomial t(z) to 0
for each ordered pair (i, j) such that i + j ≤ n do
    s_i(z) ← monomials in p(z) whose exponent as Ham-Wt i
    s_j(z) ← monomials in r(z) whose exponent as Ham-Wt j
    Compute s_ij(z) = s_i(z) * s_j(z) (standard multiplication)
    t(z) ← t(z)+ monomials in s_ij(z) whose exponent has Ham-Wt i + j
return t(z)
```

Running time : $\mathcal{O}(n^2 \times d^2)$ where d is degree of polynomial

Algorithm to compute $(\star)$

Algorithm to compute modified multiplication $(\star)$

Input: Two polynomials $p(z), r(z)$
Output: $p(z) \star r(z)$

```
Initialize polynomial t(z) to 0
for each ordered pair (i, j) such that i + j ≤ n do
    s_i(z) ← monomials in p(z) whose exponent as Ham-Wt i
    s_j(z) ← monomials in r(z) whose exponent as Ham-Wt j
    Compute s_ij(z) = s_i(z) * s_j(z) (standard multiplication)
    t(z) ← t(z)+ monomials in s_ij(z) whose exponent has Ham-Wt i + j
return t(z)
```

Running time : $\mathcal{O}(n^2 \times d^2)$ where d is degree of polynomial

Algorithm to compute modified multiplication $(\star)$

Input: Two polynomials $p(z), r(z)$
Output: $p(z) \star r(z)$

Initialize polynomial $t(z)$ to 0
for *each ordered pair* (i, j) *such that* $i + j \leq n$ **do**
    $s_i(z) \leftarrow$ monomials in $p(z)$ whose exponent as Ham-Wt i
    $s_j(z) \leftarrow$ monomials in $r(z)$ whose exponent as Ham-Wt j
    Compute $s_{ij}(z) = s_i(z) * s_j(z)$ (standard multiplication)
    $t(z) \leftarrow t(z)+$ monomials in $s_{ij}(z)$ whose exponent has Ham-Wt $i + j$
return $t(z)$

Running time : $\mathcal{O}(n^2 \times d^2)$ where d is degree of polynomial

Algorithm to compute $(\star)$

Algorithm to compute modified multiplication $(\star)$

Input: Two polynomials $p(z), r(z)$
Output: $p(z) \star r(z)$

```
Initialize polynomial t(z) to 0
for each ordered pair (i, j) such that i + j ≤ n do
    s_i(z) ← monomials in p(z) whose exponent as Ham-Wt i
    s_j(z) ← monomials in r(z) whose exponent as Ham-Wt j
    Compute s_ij(z) = s_i(z) * s_j(z) (standard multiplication)
    t(z) ← t(z)+ monomials in s_ij(z) whose exponent has Ham-Wt i + j
return t(z)
```

Running time : $\mathcal{O}(n^2 \times d^2)$ where d is degree of polynomial

Algorithm to compute $(\star)$

Algorithm to compute modified multiplication $(\star)$

Input: Two polynomials $p(z), r(z)$

Output: $p(z) \star r(z)$

```
Initialize polynomial t(z) to 0
for each ordered pair (i, j) such that i + j ≤ n do
    s_i(z) ← monomials in p(z) whose exponent as Ham-Wt i
    s_j(z) ← monomials in r(z) whose exponent as Ham-Wt j
    Compute s_ij(z) = s_i(z) * s_j(z) (standard multiplication)
    t(z) ← t(z)+ monomials in s_ij(z) whose exponent has Ham-Wt i + j
return t(z)
```

Running time : $\mathcal{O}(n^2 \times d^2)$ where d is degree of polynomial

Algorithm to compute ($\star$)

Algorithm to compute modified multiplication ($\star$)

Input: Two polynomials $p(z), r(z)$
Output: $p(z) \star r(z)$

```
Initialize polynomial t(z) to 0
for each ordered pair (i, j) such that i + j ≤ n do
    s_i(z) ← monomials in p(z) whose exponent as Ham-Wt i
    s_j(z) ← monomials in r(z) whose exponent as Ham-Wt j
    Compute s_ij(z) = s_i(z) * s_j(z) (standard multiplication)
    t(z) ← t(z)+ monomials in s_ij(z) whose exponent has Ham-Wt i + j
return t(z)
```

Running time : $\mathcal{O}(n^2 \times d^2)$ where d is degree of polynomial

Algorithm to compute $(\star)$

Algorithm to compute modified multiplication $(\star)$

Input: Two polynomials $p(z), r(z)$
Output: $p(z) \star r(z)$

Initialize polynomial $t(z)$ to 0
for *each ordered pair* (i, j) *such that* $i + j \leq n$ **do**
    $s_i(z) \leftarrow$ monomials in $p(z)$ whose exponent as Ham-Wt i
    $s_j(z) \leftarrow$ monomials in $r(z)$ whose exponent as Ham-Wt j
    Compute $s_{ij}(z) = s_i(z) * s_j(z)$ (standard multiplication)
    $t(z) \leftarrow t(z) +$ monomials in $s_{ij}(z)$ whose exponent has Ham-Wt $i + j$
return $t(z)$

Running time : $\mathcal{O}(n^2 \times d^2)$ where d is degree of polynomial

Exact Algorithm

Maximum degree of $q(z)$ is 2^n ($\Rightarrow$) running time $\mathcal{O}(n^2 \times 4^n)$

Lemma (Fast Fourier Transform [SS71])

Two polynomials of degree at most d over any commutative ring $\mathcal{R}$ can be multiplied using $\mathcal{O}(d \cdot \log d \cdot \log\log d)$ additions and multiplications in $\mathcal{R}$.

($\Rightarrow$) running time $\mathcal{O}(n^2 \times 2^n \cdot n \cdot \log n)$
Since $\chi_{cd}(G) \leq |V(G)|$, we need to multiply two polynomials at most n times

Theorem

Given a graph G on n vertices, there is an algorithm which finds its cd-chromatic number in $\mathcal{O}(2^n n^4 \log n)$ time.

Exact Algorithm

Maximum degree of $q(z)$ is 2^n ($\Rightarrow$) running time $\mathcal{O}(n^2 \times 4^n)$

Lemma (Fast Fourier Transform [SS71])

Two polynomials of degree at most d over any commutative ring $\mathcal{R}$ can be multiplied using $\mathcal{O}(d \cdot \log d \cdot \log\log d)$ additions and multiplications in $\mathcal{R}$.

($\Rightarrow$) running time $\mathcal{O}(n^2 \times 2^n \cdot n \cdot \log n)$
Since $\chi_{cd}(G) \leq |V(G)|$, we need to multiply two polynomials at most n times

Theorem

Given a graph G on n vertices, there is an algorithm which finds its cd-chromatic number in $\mathcal{O}(2^n n^4 \log n)$ time.

Exact Algorithm

Maximum degree of $q(z)$ is 2^n ($\Rightarrow$) running time $\mathcal{O}(n^2 \times 4^n)$

Lemma (Fast Fourier Transform [SS71])

Two polynomials of degree at most d over any commutative ring $\mathcal{R}$ can be multiplied using $\mathcal{O}(d \cdot \log d \cdot \log\log d)$ additions and multiplications in $\mathcal{R}$.

($\Rightarrow$) running time $\mathcal{O}(n^2 \times 2^n \cdot n \cdot \log n)$

Since $\chi_{cd}(G) \leq |V(G)|$, we need to multiply two polynomials at most n times

Theorem

Given a graph G on n vertices, there is an algorithm which finds its cd-chromatic number in $\mathcal{O}(2^n n^4 \log n)$ time.

Exact Algorithm

Maximum degree of $q(z)$ is 2^n ($\Rightarrow$) running time $\mathcal{O}(n^2 \times 4^n)$

Lemma (Fast Fourier Transform [SS71])

Two polynomials of degree at most d over any commutative ring $\mathcal{R}$ can be multiplied using $\mathcal{O}(d \cdot \log d \cdot \log\log d)$ additions and multiplications in $\mathcal{R}$.

($\Rightarrow$) running time $\mathcal{O}(n^2 \times 2^n \cdot n \cdot \log n)$
Since $\chi_{cd}(G) \leq |V(G)|$, we need to multiply two polynomials at most n times

Theorem

Given a graph G on n vertices, there is an algorithm which finds its cd-chromatic number in $\mathcal{O}(2^n n^4 \log n)$ time.

Exact Algorithm

Maximum degree of $q(z)$ is 2^n $(\Rightarrow)$ running time $\mathcal{O}(n^2 \times 4^n)$

Lemma (Fast Fourier Transform [SS71])

Two polynomials of degree at most d over any commutative ring $\mathcal{R}$ can be multiplied using $\mathcal{O}(d \cdot \log d \cdot \log\log d)$ additions and multiplications in $\mathcal{R}$.

$(\Rightarrow)$ running time $\mathcal{O}(n^2 \times 2^n \cdot n \cdot \log n)$
Since $\chi_{cd}(G) \leq |V(G)|$, we need to multiply two polynomials at most n times

Theorem

Given a graph G on n vertices, there is an algorithm which finds its cd-chromatic number in $\mathcal{O}(2^n n^4 \log n)$ time.

cd-Partization

> CD-PARTIZATION **Parameter:** k, q
> **Input:** Graph G, integers k and q
> **Question:** Does there exist $S \subseteq V(G)$, $|S| \leq k$, such that $\chi_{cd}(G - S) \leq q$?

Theorem

q-CD-PARTIZATION is NP-complete for $q \in \{2, 3\}$.

- Note $\mathcal{G} = \{G \mid \chi_{cd}(G) \leq q\}$ is not a hereditary graph class
- Result of Lewis and Yannakakis [LY80] does not imply NP-hardness.
- Reduction from q-PARTIZATION

cd-Partization

cd-Partization **Parameter:** k, q
Input: Graph G, integers k and q
Question: Does there exist $S \subseteq V(G)$, $|S| \leq k$, such that $\chi_{cd}(G - S) \leq q$?

Theorem

q-cd-Partization is NP-complete for $q \in \{2, 3\}$.

- Note $\mathcal{G} = \{G \mid \chi_{cd}(G) \leq q\}$ is not a hereditary graph class
- Result of Lewis and Yannakakis [LY80] does not imply NP-hardness.
- Reduction from q-Partization

cd-Partization

CD-PARTIZATION **Parameter:** k, q
Input: Graph G, integers k and q
Question: Does there exist $S \subseteq V(G)$, $|S| \leq k$, such that $\chi_{cd}(G - S) \leq q$?

Theorem

q-CD-PARTIZATION is NP-complete for $q \in \{2, 3\}$.

- Note $\mathcal{G} = \{G \mid \chi_{cd}(G) \leq q\}$ is not a hereditary graph class
- Result of Lewis and Yannakakis [LY80] does not imply NP-hardness.
- Reduction from q-PARTIZATION

cd-Partization

CD-PARTIZATION **Parameter:** k, q
Input: Graph G, integers k and q
Question: Does there exist $S \subseteq V(G)$, $|S| \leq k$, such that $\chi_{cd}(G - S) \leq q$?

Theorem

q-CD-PARTIZATION is NP*-complete for $q \in \{2, 3\}$.*

- Note $\mathcal{G} = \{G \mid \chi_{cd}(G) \leq q\}$ is not a hereditary graph class
- Result of Lewis and Yannakakis [LY80] does not imply NP-hardness.
- Reduction from q-PARTITION

cd-Partization on Split Graphs

Split Graph : Vertex set can be partitioned into a clique and an independent set.

cd-Partization on Split Graphs

Split Graph : Vertex set can be partitioned into a clique and an independent set.

Theorem

CD-PARTIZATION *on split graphs is* NP*-hard.*

cd-Partization on Split Graphs

Split Graph : Vertex set can be partitioned into a clique and an independent set.

Theorem

CD-PARTIZATION *on split graphs is* NP*-hard.*

(Parameter preserving) Reduction from SET COVER

cd-Partization on Split Graphs

Split Graph : Vertex set can be partitioned into a clique and an independent set.

Theorem

CD-PARTIZATION *on split graphs is* NP-*hard.*

(Parameter preserving) Reduction from SET COVER

Corollary

CD-PARTIZATION *on split graphs parameterized by q is $W[2]$-hard.*

cd-Partization on Split Graphs

Split Graph : Vertex set can be partitioned into a clique and an independent set.

Theorem

CD-PARTIZATION *on split graphs is* NP*-hard.*

(Parameter preserving) Reduction from SET COVER

Corollary

CD-PARTIZATION *on split graphs parameterized by q is $W[2]$-hard.*

Theorem

CD-PARTIZATION *on split graphs is* FPT *with respect to parameters q and k. Furthermore, the problem does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly*.*

Exact Algorithms for cd-Partization

Theorem

Given a graph G and an integer k, there is an algorithm that determines if there is a set S of size k whose deletion results in a graph H with $\chi_{cd}(H) \leq 3$ in $\mathcal{O}^*(2.3146^k)$ time.

Exact Algorithms for cd-Partization

Theorem

Given a graph G and an integer k, there is an algorithm that determines if there is a set S of size k whose deletion results in a graph H with $\chi_{cd}(H) \leq 3$ in $\mathcal{O}^*(2.3146^k)$ time.

- Complete characterization of class $\mathcal{H} = \{H \mid \chi_{cd}(H) \leq 3\}$
- Use exact algorithms for Vertex Cover and Odd Cycle Transversal as sub-routine

References

Bruno Courcelle.
The Monadic Second-order Logic of Graphs III: Tree-decompositions, Minor and Complexity Issues.
ITA, 26:257–286, 1992.

Vadim V Lozin and Marcin Kaminski.
Coloring Edges and Vertices of Graphs Without Short or Long Cycles.
Contributions to Discrete Mathematics, 2(1):61–66, 2007.

J. M. Lewis and M. Yannakakis.
The Node-Deletion Problem for Hereditary Properties is NP-Complete.
Journal of Computer and System Sciences, 20(2):219–230, 1980.

Doz Dr A Schönhage and Volker Strassen.
Schnelle Multiplikation Grosser Zahlen.

Thank you!