Fragments of domination theory in graphs: invariants, interrelations, and algorithmic aspects (II)

Martin Milanič, University of Primorska
martin.milanic@upr.si

CIMPA Research School on Graph Structure and Complex Network Analysis
Nesin Mathematical Village, Turkey

June 12, 2023

Outline of the lectures:

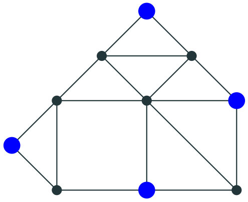
1. Motivations and background ✓
2. Efficient algorithms for special cases – three examples ✓
3. Approximating the minimum dominating set problem via Set Cover
4. Interrelations between graph domination parameters
5. Approximating the vector dominating set via Submodular Cover

Approximation algorithms

A **dominating set** in a graph $G = (V, E)$:

a set $S \subseteq V$ such that each vertex not in S has at least one neighbor in S.

Example:



The associated optimization problem is as follows:

MINIMUM DOMINATING SET	
Input:	A graph G.
Task:	Find a smallest dominating set in G.

The problem also has a **weighted version**:

each vertex has its price; we are looking for a cheapest dominating set.

Approximately solving MINIMUM DOMINATING SET

Approximation algorithms – basic definitions

An **instance** of an optimization problem is an ordered triple $(\mathcal{S}, f, opt)$, where:

- $\mathcal{S}$ is an (implicitly given) set of feasible solutions
- $f : \mathcal{S} \to \mathbb{R}$ is the objective function
- $opt \in \{\min, \max\}$ type of problem: *minimization* or *maximization*

We are looking for

$$\mathrm{OPT} := opt\{f(x) \mid x \in \mathcal{S}\}\,.$$

Example:

MINIMUM DOMINATING SET:

Input: a graph $G = (V, E)$

$\mathcal{S} = \{S \subseteq V : S \text{ is a dominating set in } G\}$

$f(S) = |S|$

$opt = \min$ ▲

Let Π be an optimization problem such that for every instance of the problem and every feasible solution $x \in \mathcal{S}$, the objective function value takes **positive value** $(f(x) > 0)$.

A ρ-**approximation algorithm**:

- An algorithm A for an optimization problem Π that runs in polynomial time and such that:
- For every instance of Π,
  A outputs a feasible solution with objective function value within a factor of ρ of true optimum for that instance.

ρ = approximation ratio (or: approximation factor)

More specifically:

- for **minimization** problems:
  for every instance I, we have

$$f_A(I) \leq \rho \cdot \mathrm{OPT}(I)\,,$$

  where
  $f_A(I)$ is the value of the solution returned by the algorithm, and
  $\mathrm{OPT}(I)$ is the optimal solution value.

- for **maximization** problems: $f_A(I) \geq \mathrm{OPT}(I)/\rho\,.$

A detour: Approximating the Set Cover problem

Set Cover	
Input:	A ground set $U = \{u_1, \ldots, u_n\}$,
	a family $\mathcal{F} = \{S_1, \ldots, S_m\}$ of subsets of U,
	(we assume $S_1 \cup \cdots \cup S_m = U$)
	positive costs of subsets $c(S_1), \ldots, c(S_m)$.
Task:	Find a cheapest covering subfamily $\mathcal{F}' \subseteq \mathcal{F}$.

A subfamily $\mathcal{F}' = \{S_{i_1}, \ldots, S_{i_k}\}$ is said to be **covering**

(or: a **cover**) if $S_{i_1} \cup \cdots \cup S_{i_k} = U$.

Example:

$U = \{1, 2, 3, 4, 5, 6\}$

$S_1 = \{1, 2, 3, 4\}$, $c(S_1) = 9$

$S_2 = \{1, 2, 5\}$, $c(S_2) = 5$

$S_3 = \{2, 3, 4\}$, $c(S_3) = 3$

$S_4 = \{2, 3, 6\}$, $c(S_3) = 4$

$S_5 = \{5, 6\}$, $c(S_3) = 2$

Cheapest cover: $\{S_2, S_3, S_5\}$

Cost of the cover: $c(S_2) + c(S_3) + c(S_5) = 10$.

Greedy method for Set Cover

C: set of already covered elements of U

$\overline{C} = U \setminus C$: set of not yet covered elements of U

effective cost of a set $S := c(S)/|S \cap \overline{C}|$

Greedy-Cover$(U, S_1, \ldots, S_m, c)$:
$C \leftarrow \emptyset$, $F \leftarrow \emptyset$
while $C \neq U$ **do**
    $S \leftarrow$ set with minimum effective cost
    $F \leftarrow F \cup \{S\}$, $C \leftarrow C \cup S$
end while
return F

C: set of already covered elements of U

$\overline{C} = U \setminus C$: set of not yet covered elements of U

effective cost of a set $S := c(S)/|S \cap \overline{C}|$

For the purpose of the analysis, we introduce a cost for each newly covered element:

```
Greedy-Cover(U, S_1, ..., S_m, c):
C ← ∅, F ← ∅
while C ≠ U do
    S ← set with minimum effective cost
    α ← c(S)/|S ∩ C̄|
    for each u ∈ S ∩ C̄ do cost(u) = α
    F ← F ∪ {S}, C ← C ∪ S
end while
return F
```

The analysis

We may assume that the algorithm covers elements $u_1, \dots, u_n$ in this order.

Claim

For all $k = 1, \dots, n$ we have:

$$cost(u_k) \leq \frac{\mathrm{OPT}}{n - k + 1} .$$

Proof:

Let $\overline{C}$ be the set of uncovered elements just before element u_k gets covered.

Elements in $\overline{C}$ can be covered with at most $|\overline{C}|$ sets of total cost $\leq \mathrm{OPT}$.

Hence, there exists a set S with effective cost $\leq \frac{\mathrm{OPT}}{|\overline{C}|}$.

Indeed: suppose that for all sets S from the cover of set $\overline{C}$ (as above) we have

$$\frac{\mathrm{OPT}}{|\overline{C}|} < \frac{c(S)}{|S \cap \overline{C}|} \,.$$

Then we would have

$$|S \cap \overline{C}| \cdot \mathrm{OPT} < c(S) \cdot |\overline{C}|$$

and consequently, by summing up over all S from the cover of $\overline{C}$ (as above) and using the inequalities $|\overline{C}| \leq \sum_S |S \cap \overline{C}|$ and $\sum_S c(S) \leq \mathrm{OPT}$ we would derive a contradiction $|\overline{C}| \cdot \mathrm{OPT} < |\overline{C}| \cdot \mathrm{OPT}$.

Consequently:

$$cost(u_k) \leq \frac{\mathrm{OPT}}{|\overline{C}|} \leq \frac{\mathrm{OPT}}{n-k+1} \,.$$

$\square$

Proposition

Greedy-Cover *is an* $H(n)$*-approximation algorithm for* Set Cover, *where*

$$H(n) = 1 + 1/2 + 1/3 + \ldots + 1/n \leq \log n + 1\,.$$

Proof:

$$c(F) = \sum_{k=1}^{n} cost(u_k) \leq \sum_{k=1}^{n} \frac{\mathrm{OPT}}{n-k+1} = \mathrm{OPT} \cdot \left(\sum_{k=1}^{n} \frac{1}{k} \right)\,.$$

Most likely, this is best possible.

Theorem (Dinur-Steurer 2014)

For any $\epsilon > 0$*, if there exists an approximation algorithm for* Set Cover *with approximation ratio* $(1-\epsilon)\log n$*, then* $\mathrm{P} = \mathrm{NP}$*.*

The result holds even for unit costs.

Back to Domination

MINIMUM DOMINATING SET	
Input:	A graph $G = (V, E)$.
Task:	Compute a dominating set of minimum cardinality.

We have seen that MINIMUM DOMINATING SET is NP-hard.

How well can it be approximated?

We can model MINIMUM DOMINATING SET as a special case of SET COVER.

Let us say that a vertex v is **dominated** by a set S if

either v is in S or v has a neighbor in S.

Then, placing a vertex x in S dominates all elements of its closed neighborhood, defined as

$$N[x] = \{x\} \cup N(x) .$$

So we can take:

- the ground set $U = V$,
- the set family $\mathcal{F} = \{S_v : v \in V\}$ where $S_v = N[v]$,
- the cost function $c(S_v) = 1$ for all $v \in V$.

Indeed, we then have:

A set $S \subseteq V$ is a dominating set in G if and only if the set $\{S_v : v \in S\}$ is a covering subfamily of $\mathcal{F}$. And conversely, every covering subfamily arises this way.

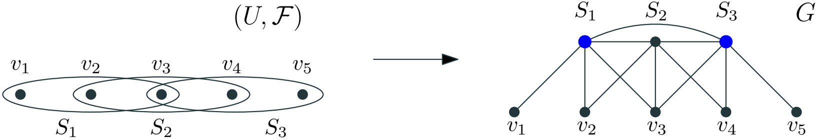
Corollary

The MINIMUM DOMINATING SET *problem can be approximated to within a factor of* $\log n + 1$ *on n-vertex graphs.*

Again, this result is essentially **best possible**.

- The approach leading to the inapproximability result of Dinur and Steurer for the SET COVER problem can be modified to obtain a similar result for MINIMUM DOMINATING SET.

The idea is similar as for proving NP-hardness, with a reduction from SET COVER and the roles of clique and independent set interchanged:



Quiz

Consider the following algorithm for the MINIMUM DOMINATING SET problem:

Greedy Domination

$H = G$; $S = \emptyset$;

while $(V(H) \neq \emptyset)$

Let v be a vertex of maximum degree in the graph H;

$S = S \cup \{v\}$;

$H = H - N[v]$;

end while

return S;

It is not difficult to make sure that the algorithm runs in polynomial time and that the computed set S is indeed a dominating set in the graph G.

However, the algorithm can compute very bad solutions even in the case when the input graph is a tree.

Show that for any constant $\rho \geq 1$ the algorithm **Greedy Domination** is not a ρ-approximation for the Minimum Dominating Set problem in the class of trees.

Solution: For $n \geq 2$, let T_n be a tree where each vertex has degree n or 1, obtained by taking a star $K_{1,n}$ and identifying each leaf of it with the center of a star $K_{1,n-1}$.

Formally, the vertex set of the tree T_n is $\{v_0\} \cup \{v_1, \ldots, v_n\} \cup V_1 \cup \ldots \cup V_n$ (disjoint union), where $|V_i| = n - 1$ for all $i \in \{1, \ldots, n\}$, and the edge set is $\cup_{i=1}^{n} \{v_i w \mid w \in \{v_0\} \cup V_i\}$.

The algorithm can then first place the vertex v_0 in the set S.

It then computes $H = H - N[v_0]$, which is a graph with $n(n-1)$ isolated vertices.

In this case, the final set computed by the algorithm is $\{v_0\} \cup \bigcup_{i=1}^{n} V_i$.

The cardinality of this set is $n(n-1) + 1 = n^2 - n + 1$.

However, the tree T_n has a dominating set with cardinality n, namely the set $\{v_1, \ldots, v_n\}$.

The ratio between the quality of the solution computed by the algorithm and the quality of the optimal solution is therefore $n - 1 + 1/n$.

The value $n - 1 + 1/n$ is strictly greater than ρ for all $n \geq \rho + 1$.

Variants of Domination

The same approach can be used to model
many other variants of domination, for example:

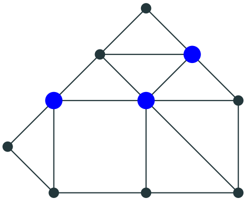
- **total domination**: every vertex has a neighbor in the set
- **distance-k domination**: every vertex is at distance at most k from a vertex in the set
- **vertex cover** (here, one can do better with a different approach: there is a 2-approximation)

Multiple domination

***k*-dominating set**: a set $S \subseteq V$ such that each vertex not in S has at least k neighbors in S.

1-dominating set = dominating set

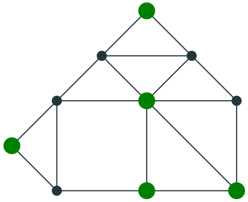
Example: the set of blue vertices is not a 2-dominating set



k-dominating set: a set $S \subseteq V$ such that each vertex not in S has at least k neighbors in S.

1-dominating set = dominating set

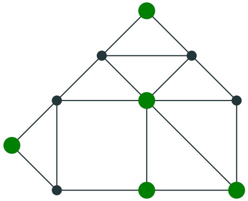
Example: but the following set is:



The ***k*-domination number** of a graph G is denoted by $\gamma_k(G)$ and defined as the minimum cardinality of a k-dominating set in G.

Example:

A graph G with $\gamma_2(G) \leq 5$:



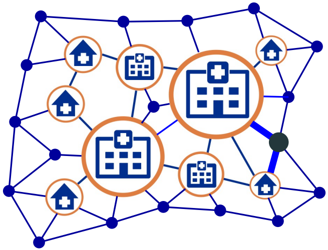
Exercise: show that $\gamma_2(G) = 5$.

Example application: ensuring fault-tolerance of networks

- Even if in the network some of the connections ($< k$) from a vertex outside S fail, the communication will still be possible.

A concrete example (for $k = 2$):

- Suppose that a non-hospital city is directly linked to two hospitals.
- In that case, even if one of the two roads is closed, residents of the city will be able to quickly reach a hospital.

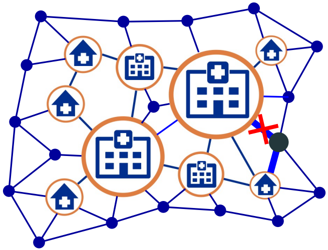


Example application: ensuring fault-tolerance of networks

- Even if in the network some of the connections ($< k$) from a vertex outside S fail, the communication will still be possible.

A concrete example (for $k = 2$):

- Suppose that a non-hospital city is directly linked to two hospitals.
- In that case, even if one of the two roads is closed, residents of the city will be able to quickly reach a hospital.



All of the above problems also have a **weighted version**:

each vertex has its price; we are looking for a cheapest dominating set of the appropriate type.

- In the book *Haynes, Hedetniemi, Slater, Fundamentals of domination in graphs (1998)* the authors list over 75 types of domination problems.

A question for thought

- Think of a realistic scenario that can be modeled with some variant of domination.
- If necessary, you may "invent" a new variant of domination.

Interrelations between graph domination parameters

Suppose that vertices must be **dominated twice or using the sum of weights that equals at least** 2. We obtain 6 graph domination parameters according to the following **two criteria**:

1. The set of **weights** that are allowed to be assigned to vertices: either $\{0, 1\}$ or $\{0, 1, 2\}$.

2. Three possibilities regarding what is considered as **domination**:

- **outer domination**: only vertices with weight 0 need to be dominated,
- **closed domination**: all vertices need to be dominated and vertices with a positive weight dominate their closed neighborhoods,
- **open domination**: all vertices need to be dominated and only open neighborhoods are dominated by vertices with positive weight.

	$\{0,1\}$	$\{0,1,2\}$
outer	2-domination (γ_2)	weak 2-domination (γ_{w2})
closed	double domination ($\gamma_{\times 2}$)	$\{2\}$-domination ($\gamma_{\{2\}}$)
open	total double domination ($\gamma_{t\times 2}$)	total $\{2\}$-domination ($\gamma_{t\{2\}}$)

Reference: Bonomo, Brešar, Grippo, M, Safe. Domination parameters with number 2: interrelations and algorithmic consequences. *Discrete Applied Math.* 235 (2018) 23–50.

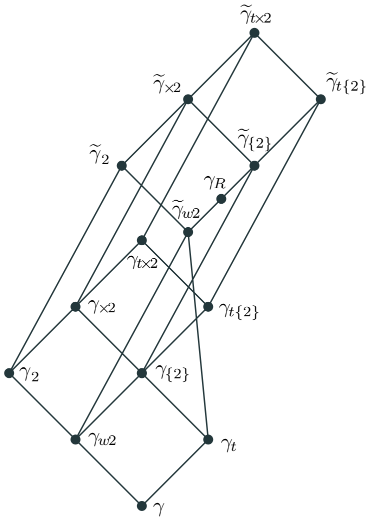
We also consider a third criterion, yielding **"rainbow versions"** of the six parameters:

- weight 0 is replaced by the label $\emptyset$, weight 1 by labels $\{a\}$ and $\{b\}$, and weight 2 by the label $\{a, b\}$,
- the conditions imposed by each parameter are meaningfully adjusted to the rainbow version.

The main difference is that instead of the sum of values of weights, in a rainbow version one considers the **union of labels**, and also the condition of having weight 2 in a neighborhood corresponds to having label $\{a, b\}$.

Together with the extensively studied **Roman domination**, $\gamma_R(G)$, and two classical parameters, the domination number, $\gamma(G)$, and the total domination number, $\gamma_t(G)$, we considered 15 domination parameters in graphs.

Hasse diagram of the relation $\leq$ among the parameters:



Suppose that we have two parameters ρ and ρ' such that

$$\rho(G) \leq \rho'(G) \text{ for all graphs } G\,.$$

Then, the following questions are of interest:

- Is there a function f such that $\rho'(G) \leq f(\rho(G))$ for all graphs G?
- If so, is there a constant $c \geq 1$ such that $\rho'(G) \leq c \cdot \rho(G)$ for all graphs G?

It turns out that for the 15 considered parameters, the two conditions are **equivalent**.

An example

Recall: a **2-dominating set** in $G$is a set $S \subseteq V(G)$ such that every vertex not in S has at least two neighbors in S

$\gamma_2(G)$ = the 2-**domination number** of G = the minimum cardinality of a 2-dominating set in G

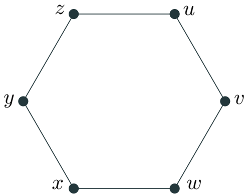
A **double dominating set** in G is a set $S \subseteq V(G)$ such that for every vertex $v \in V(G)$ we have $|S \cap N[v]| \geq 2$

In other words, this is a 2-dominating set S such that every vertex in S has a neighbor in S

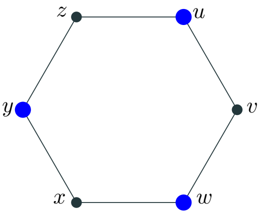
$\gamma_{\times 2}(G)$ = the **double domination number** of G = the minimum cardinality of a double dominating set

Quiz

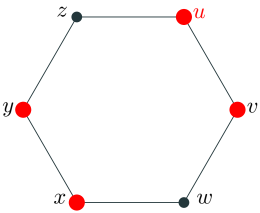
Compute the 2-domination number γ_2 and the double domination number $\gamma_{\times 2}$ of the following graph:



Solution:

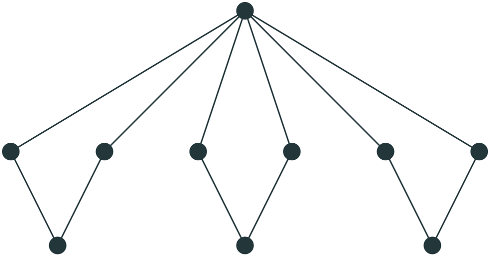


$\gamma_2(C_6) = 3$

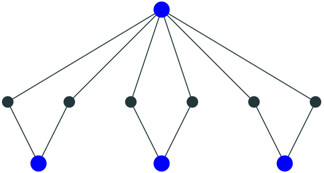


$\gamma_{\times 2}(C_6) = 4$

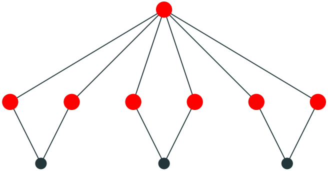
Compute the 2-domination number γ_2 and the double domination number $\gamma_{\times 2}$ of the following graph:



Solution:



$\gamma_2 = 4$



$\gamma_{\times 2} = 7$

Proposition

For every graph G without isolated vertices, it holds that

$$\gamma_2(G) \leq \gamma_{\times 2}(G) \leq 2\,\gamma_2(G) - 1\,.$$

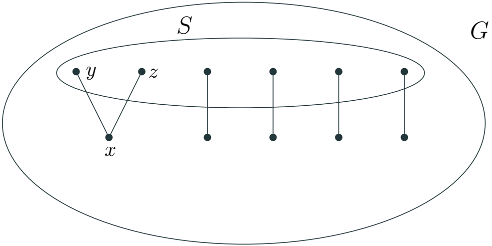
Proof: The inequality $\gamma_2(G) \leq \gamma_{\times 2}(G)$ holds, since every double dominating set is a 2-dominating set.

Let us show the inequality $\gamma_{\times 2}(G) \leq 2\,\gamma_2(G) - 1$. Let S be a minimum 2-dominating set in G.

For all $x \in V \setminus S$, we have $|S \cap N[x]| \geq 2$, hence the condition imposed on a double dominating set is already fulfilled for these vertices.

Let $y, z \in S$ be two neighbors of x.

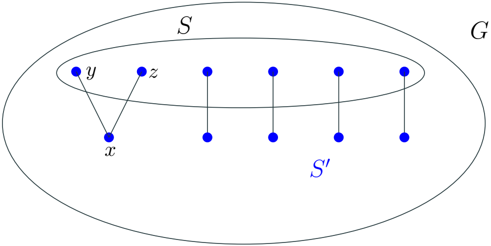
Let S' be a superset of S obtained from S by adding to it vertex x and for each vertex u from $S \setminus \{y, z\}$, adding an arbitrary vertex $v \in N(u)$.



$\square$

Let $y, z \in S$ be two neighbors of x.

Let S' be a superset of S obtained from S by adding to it vertex x and for each vertex u from $S \setminus \{y, z\}$, adding an arbitrary vertex $v \in N(u)$.



Clearly, for any $u \in S$, we have $|N[u] \cap S'| \geq 2$, where one of the vertices from $S' \cap N[u]$ is u itself.

Thus, S' is a double dominating set in G and its cardinality is

$$|S'| \leq |S| + 1 + (|S| - 2) = 2|S| - 1 = 2\,\gamma_2(G) - 1\,.$$

Let us write that $\rho \preceq \rho'$ if there exists a function f such that $\rho(G) \leq f(\rho'(G))$ for all graphs G for which the two parameters are well defined.

Then, we obtain the following Hasse diagram representing the preorder $\preceq$ on the 15 domination parameters:

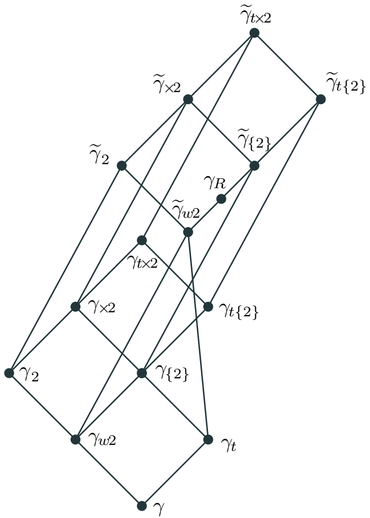
$$\boxed{\widetilde{\gamma}_{t\times 2}}$$

$$\boxed{\widetilde{\gamma}_2 \quad \widetilde{\gamma}_{\times 2}}$$

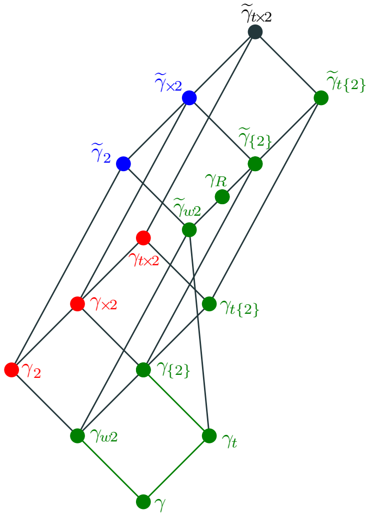
$$\boxed{\gamma_2 \quad \gamma_{\times 2} \quad \gamma_{t\times 2}}$$

$$\boxed{\begin{array}{c} \gamma \quad \gamma_t \quad \gamma_{w2} \quad \gamma_{\{2\}} \quad \gamma_{t\{2\}} \\ \widetilde{\gamma}_{w2} \quad \gamma_R \quad \widetilde{\gamma}_{\{2\}} \quad \widetilde{\gamma}_{t\{2\}} \end{array}}$$

Hasse diagram of the relation $\leq$ among the parameters:



Hasse diagram of the relation $\leq$ among the parameters:



Appplication for approximation algorithms

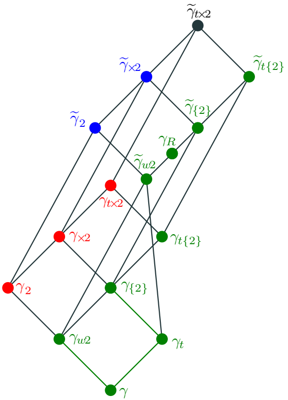
Proposition

Let ρ and ρ' be two **graph minimization parameters** *such that there exist constants $c_1, c_2 > 0$ such that $c_1 \cdot \rho(G) \leq \rho'(G) \leq c_2 \cdot \rho(G)$.*

Suppose also that there exists a polynomial-time algorithm that for a given graph G and a solution I to ρ, computes a solution J to ρ' with $|J| \leq c_2 \cdot |I|$.

Then, for every $c \geq 1$, if there is a c-approximation algorithm for ρ, then there is (cc_2/c_1)-approximation algorithm for ρ'.

Using this result, the $\mathcal{O}(\log n)$-approximation and the $\Omega(\log n)$-inapproximability results for MINIMUM DOMINATING SET imply similar results for all the graph parameters denoted by green in the figure.



Similar results also hold for all the remaining parameters, except for the topmost one, in which case no approximation is possible.

A further generalization

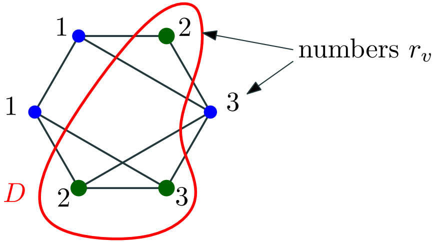
Minimum Vector Dominating Set

Input: A graph $G = (V, E)$; for every vertex v, an integer $r(v)$.

Task: Find a smallest vector dominating set.

vector dominating set: a set $S \subseteq V$ such that each vertex v not in S has at least $r(v)$ neighbors in S

Example: S is a vector dominating set

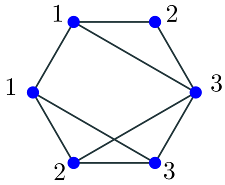


Example application:

Suppose you have a company that wants to place a new product on the market. The product is called **CoolHat**.

- Consider a set of potential buyers, each of whom is friends with a few others.
- For each of the potential buyers you know who her friends are.

  Based on a preliminary analysis of shopping habits, you have an estimate whether the fact that a number of her friends already have a CoolHat, means that she too would buy a CoolHat.



Of course, in the company, you want all potential customers to have the product.

So you want to answer the following question:

to how many customers we have to give a CoolHat for free

so that a CoolHat will also be bought by all other potential buyers based on the opinions of their friends?

So you're looking for a smallest **vector dominating set**

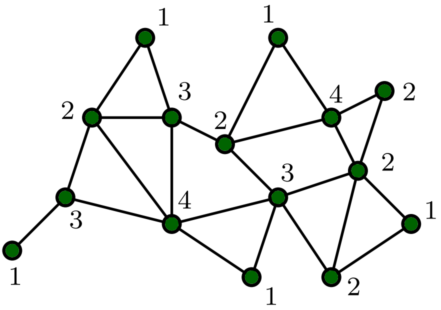
in the friendship graph where for each person $v \in V$

you put $r(v) =$ the minimum number of friends of v that must already have a CoolHat so that v will also buy it.

Given: a graph $G = (V, E)$

For every vertex v, an integer $r(v)$

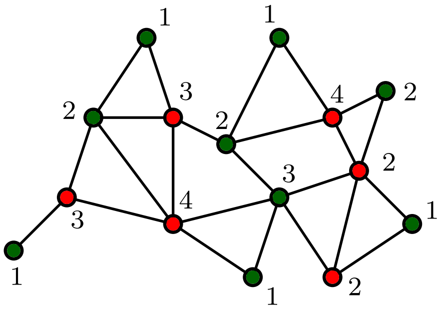
A set $S \subseteq V$ is a vector dominating set for (G, r) if every vertex v in $V \setminus S$ has at least $r(v)$ neighbors in S.



Given: a graph $G = (V, E)$

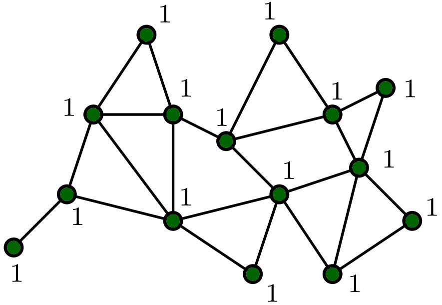
For every vertex v, an integer $r(v)$

A set $S \subseteq V$ is a vector dominating set for (G, r) if every vertex v in $V \setminus S$ has at least $r(v)$ neighbors in S.



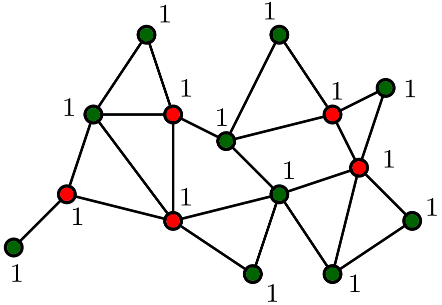
Vector dominating sets are a common generalization of:

- **dominating sets:** $r(v) = 1$ for all v



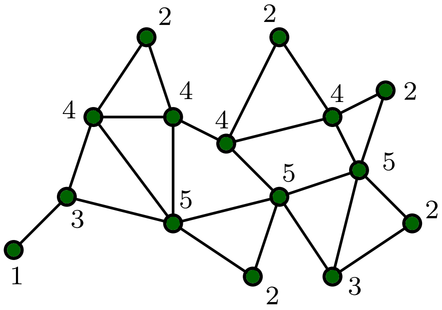
Vector dominating sets are a common generalization of:

- **dominating sets:** $r(v) = 1$ for all v



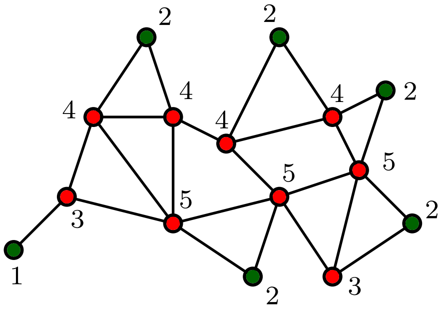
Vector dominating sets are a common generalization of:

- dominating sets: $r(v) = 1$ for all v
- **vertex covers:** $r(v) = d(v)$ for all v



Vector dominating sets are a common generalization of:

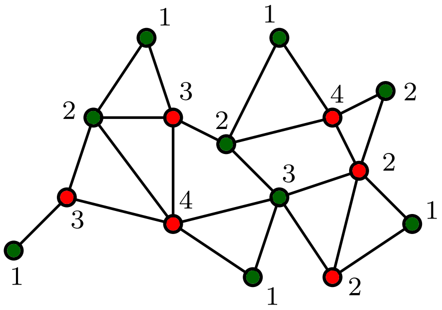
- dominating sets: $r(v) = 1$ for all v
- **vertex covers:** $r(v) = d(v)$ for all v



Given: a graph $G = (V, E)$

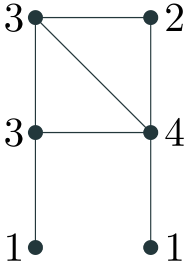
For every vertex v, an integer $r(v)$

A set $S \subseteq V$ is a vector dominating set for (G, r) if every vertex v in $V \setminus S$ has at least $r(v)$ neighbors in S.

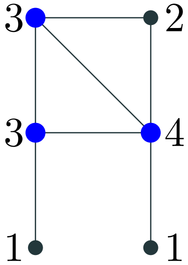


Quiz

Compute a minimum vector dominating set for the following input:



Solution:



A question for thought

How would you compute a minimum vector dominating set if the input graph G is complete?

MINIMUM VECTOR DOMINATING SET	
Input:	A graph $G = (V, E)$, an function $r(v) : V \to \mathbb{Z}^+$
Task:	Compute a minimum vector dominating set for (G, r).

There is an extension of SET COVER called SET MULTICOVER, where each element needs to be covered multiple times.

This problem can also be approximated greedily, with a ratio of $H(\Delta)$ where Δ is the maximum size of a set in the family (Dobson 1982).

Bad news:

It is not clear how to model MINIMUM VECTOR DOMINATING SET in this setting.

Good news:

We can use a different result (also from 1982) to solve this problem!

We will obtain the following:

Theorem

The MINIMUM VECTOR DOMINATING SET *problem can be approximated in polynomial time to within a factor of* $\log(2\Delta(G)) + 1$*, where* $\Delta(G)$ *is the maximum degree of* G*.*

First, note that we may assume that for all $v \in V$, we have $r(v) \leq d(v)$, where $d(v)$ is the degree (the number of neighbors) of v in G:

- If $r(v) > d(v)$, then v must be contained in every vector dominating set.

  Thus, we can set $r(w) \leftarrow r(w) - 1$ for all $w \in N(v)$ and add v to an optimal (or approximate) solution for the reduced problem on $G - v$.

Approximating vector domination

Greedy Strategy

– start with $S = \emptyset$

– if S is not a vector dominating set, keep on adding to S a vertex $v \in V \setminus S$ maximizing $f(S \cup \{v\}) - f(S)$

$\mathrm{argmax}_{v \in V} \left(f(S \cup \{v\}) - f(S)\right)$

What is f?

$$f(X) = \textstyle\sum_{v \in V} f_v(X), \text{ for all } X \subseteq V, \text{ and}$$

$$f_v(X) = \begin{cases} \min\{|X \cap N(v)|, r(v)\} & \text{if } v \notin X; \\ r(v) & \text{if } v \in X. \end{cases}$$

$|X \cap N(v)|$ = the number of already chosen neighbors of v

$$f(X) = \sum_{v \in V} f_v(X)\,, \text{ for all } X \subseteq V, \text{ and}$$

$$f_v(X) = \begin{cases} \min\{|X \cap N(v)|, r(v)\} & \text{if } v \notin X; \\ r(v) & \text{if } v \in X. \end{cases}$$

Note that:

- $f(V) = \sum_{v \in V} r(v)$
- $f(X) = f(V)$ if and only if $X \subseteq V$ is a vector dominating set for (G, r).
- Hence, the Minimum Vector Dominating Set problem asks for a **smallest set** $X \subseteq V(G)$ with $f(X) = f(V)$.

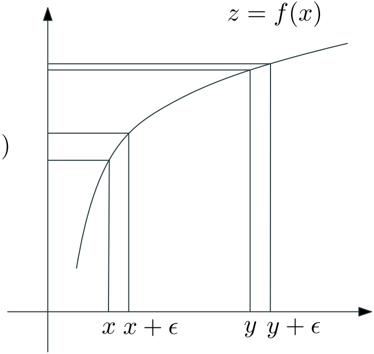
It can be shown that f is a (non-decreasing, integer-valued) **submodular set function**.

f is *non-decreasing* if $X \subseteq Y \Rightarrow f(X) \leq f(Y)$

Submodularity is a discrete analog of concavity:

$$X \subseteq Y \quad \Rightarrow \quad f(X \cup \{v\}) - f(X) \geq f(Y \cup \{v\}) - f(Y)\,.$$

$$f(x+\epsilon) - f(x) \geq f(y+\epsilon) - f(y)$$



Hence, Minimum Vector Dominating Set is a special case of the following problem:

Minimum Submodular Cover	
Input:	A finite set V and an integer-valued non-decreasing submodular set function f on subsets of V (given by an oracle – a black box which outputs $f(X)$ for a given $X \subseteq V$).
Task:	Find a smallest set $X \subseteq V$ such that $f(X) = f(V)$.

By a result of [Wolsey, 1982] on minimum submodular cover,

the greedy strategy approximates OPT by a factor of at most $H(\max f(\{y\}))$.

For the MINIMUM VECTOR DOMINATING SET problem, we have, for every $y \in V$:

$$f(\{y\}) = \sum_{v \in V \setminus \{y\}} f_v(\{y\}) + f_y(\{y\}) \leq d(y) + r(y) \leq 2d(y)\,.$$

Hence $\max_{y \in V} f(\{y\}) \leq 2\Delta(G)$ and the greedy strategy approximates OPT by a factor of at most

$$H(2\Delta(G)) \leq \log(2\Delta(G)) + 1\,,$$

as claimed. $\square$

For further details, see:

Cicalese, M, Vaccaro. On the approximability and exact algorithms for vector domination and related problems in graphs, *Discrete Applied Math.* 161 (2013) 750–767.

Thank you!

Questions?