A Problem-Solving Methodology using the Extremality Principle and its Application to CS Education

Jagadish M.
IIT Bombay

29 June 2015

(Advisor: Prof. Sridhar Iyer)

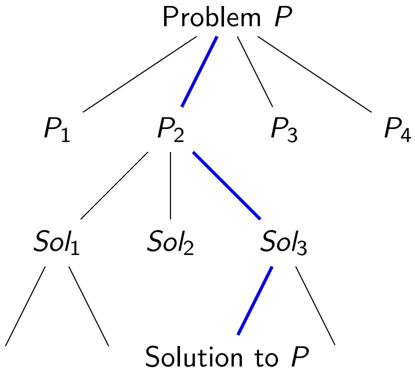
Contribution in a nutshell

We have identified a few domains in theoretical computer science and devised problem-solving techniques for problems in each domain.

Problem-solving as a search process

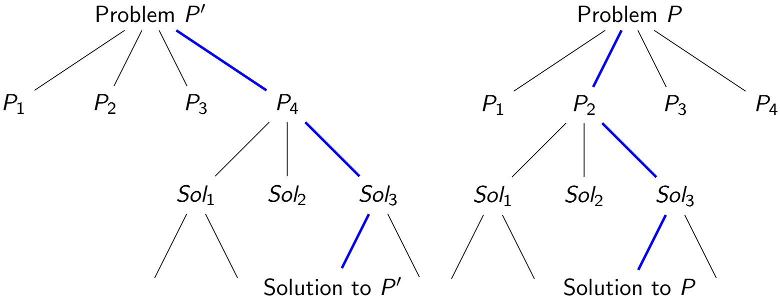
'If you can't solve a problem, then there is an easier problem you can solve: find it.'

Problem-solving as a search process



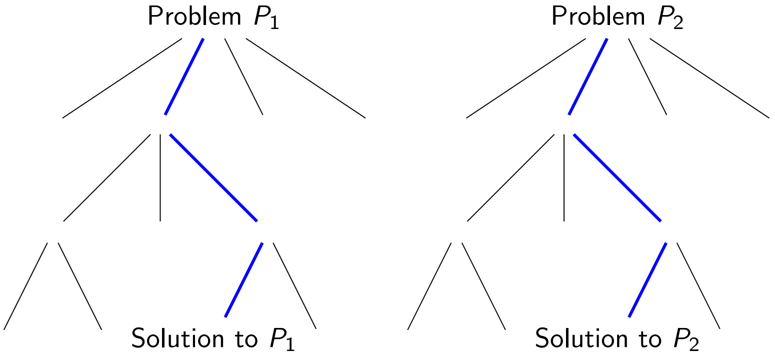
Weaken-Identify-Solve-Extend (WISE).

Too general to be useful



How to give precision?

$P_1 \approx P_2$



How to give precision?

1. Restrict the domain

2. Extremality principle

Looking at objects that maximize or minimize some properties.

Generate extremal instances

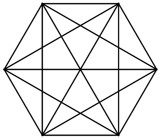
Step 1: Identify the properties of instances

Step 2: Define max/min functions on properties.

Generate extremal graphs

Step 1: Property: Number of edges

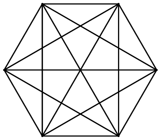
Step 2: Function: Maximum number of edges

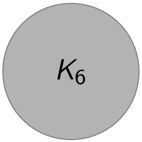


Generate extremal graphs

Step 1: Property: Number of edges

Step 2: Function: Maximum number of edges





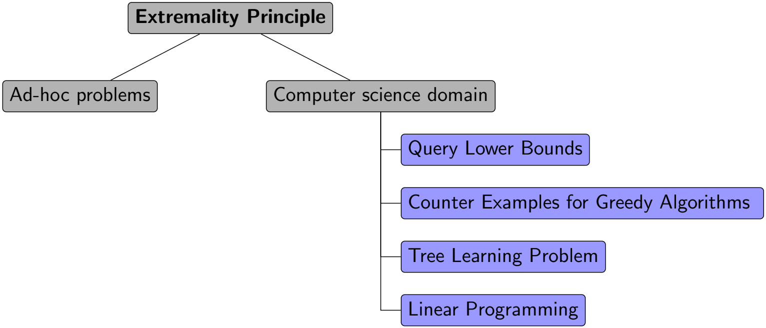
Two types of extremal instances

Domain-specific.

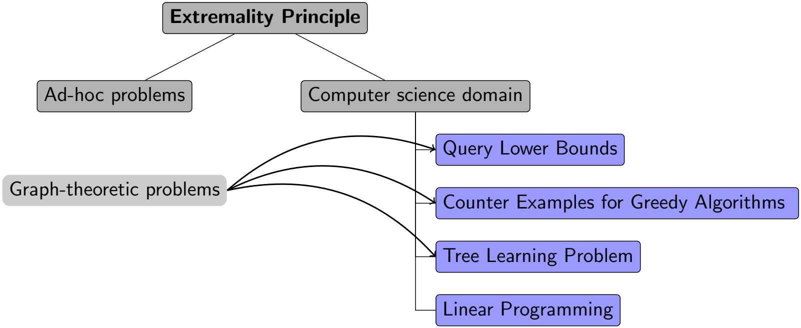
Examples: Complete graph, path, binary tree, bipartite graph, etc.

Problem-specific.

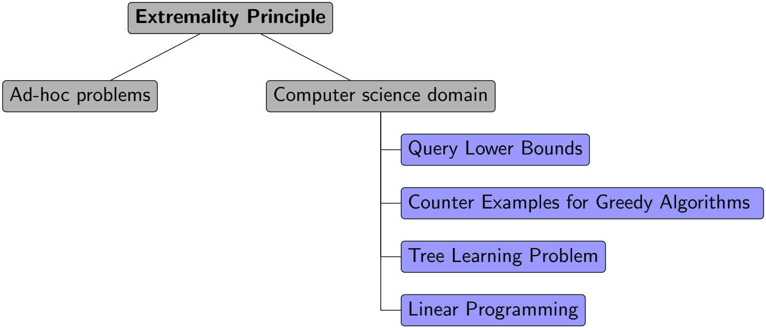
Overview


Extremality Principle
Ad-hoc problems
Computer science domain
Query Lower Bounds
Counter Examples for Greedy Algorithms
Tree Learning Problem
Linear Programming

Overview

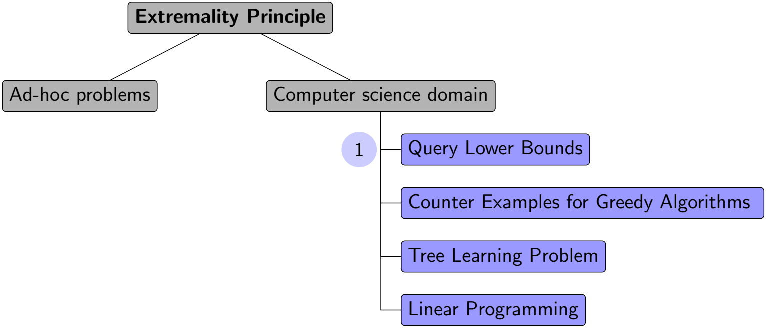

Extremality Principle
Ad-hoc problems
Computer science domain
Graph-theoretic problems
Query Lower Bounds
Counter Examples for Greedy Algorithms
Tree Learning Problem
Linear Programming

Overview



Extremality Principle
Ad-hoc problems
Computer science domain
Query Lower Bounds
Counter Examples for Greedy Algorithms
Tree Learning Problem
Linear Programming

Overview

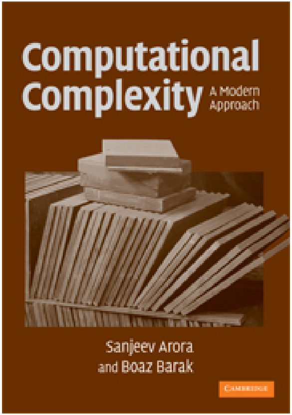


Extremality Principle
Ad-hoc problems
Computer science domain
1
Query Lower Bounds
Counter Examples for Greedy Algorithms
Tree Learning Problem
Linear Programming

Connectivity: An example

Problem We are given an adjacency matrix of a graph $G = (V, E)$. How many entries do we have to probe to check if G is connected.

Our proof vs textbook proof



Connectivity: Textbook proof

Current textbook-proofs of this problem goes through invariant analysis.


```
HIDECONNECTEDNESS(e):
  if M \ {e} is connected
      remove (i, j) from M
      return 0
  else
      add e to Y
      return 1
```

Notice that the graphs Y and M are both consistent with the adversary's answers at all times. The adversary strategy maintains a few other simple invariants.

- ***Y* is a subgraph of *M*.** This is obvious.
- ***M* is connected.** This is also obvious.
- **If *M* has a cycle, none of its edges are in *Y*.** If M has a cycle, then deleting any edge in that cycle leaves M connected.
- ***Y* is acyclic.** This follows directly from the previous invariant.
- **If $Y \neq M$, then *Y* is disconnected.** The only connected acyclic graph is a tree. Suppose Y is a tree and some edge e is in M but not in Y. Then there is a cycle in M that contains e, all of whose other edges are in Y. This violated our third invariant.

We can also think about the adversary strategy in terms of minimum spanning trees. Recall the anti-Kruskal algorithm for computing the *maximum* spanning tree of a graph: Consider the edges one

Drawback. Not easily generalizable to other properties.

Our proof: Main idea

Problem of proving a lower bound.

$\Downarrow$ reduced to

Constructing a certain extremal graph.

Our proof: Main idea

Problem of proving a lower bound.

$\Downarrow$ reduced to

Constructing a certain extremal graph.

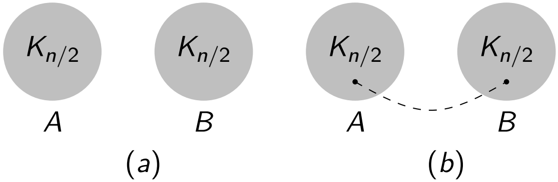
- More generalizable than the invariant proof.
- Students find the second task easy (Pilot experiment).

Critical Graph

A graph G is called a critical graph with respect to property P if

- Graph G does not have the property P.
- But replacing any non-edge with an edge endows G with the property.

Connectivity



- G is not connected.
- But replacing any non-edge with an edge makes it so.

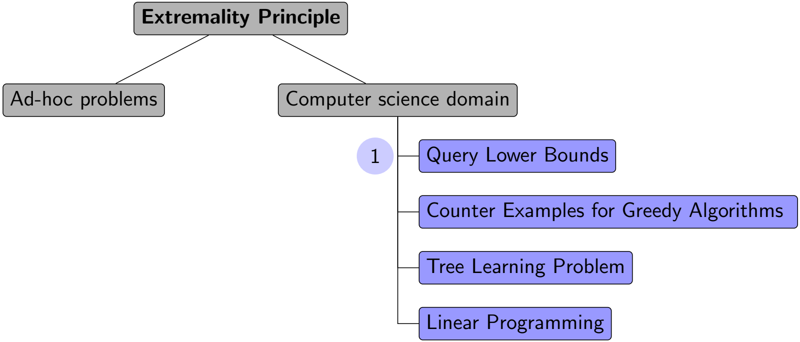
Advantage of our proof

Critical graphs for other topological graph properties are easy to construct. Hence, proving lower bounds becomes easy.

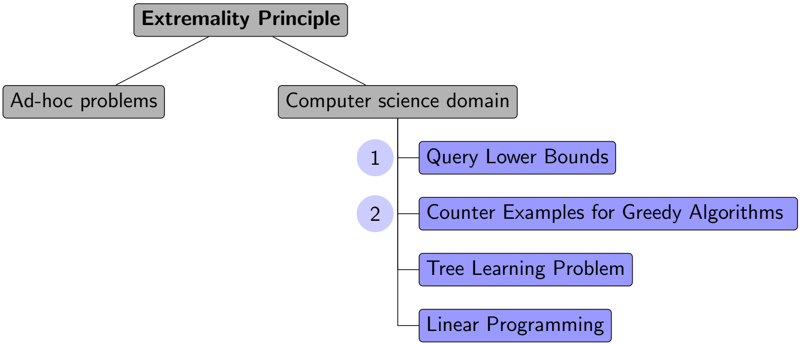
Applicability

Property	**Extremal critical graph**
Connectivity	$2K_n$s
Triangle-freeness	Star
Hamiltonicity	$2K_n$s
Perfect matching	$2K_n$s
Bipartiteness	$K_{n,n}$
Cyclic	Path
Degree-three node	Cycle
Planarity	Triangulated graph
Eulerian	Cycle

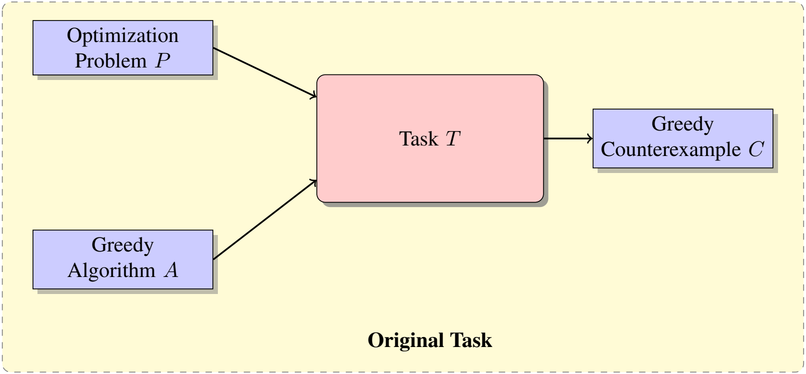
Overview: Counterexamples



Overview: Counterexamples

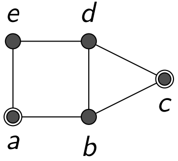


Scope of problems



Max. Independent Set Problem (MIS)

A set of vertices S is said to be *independent* if no two vertices in S have an edge between them.

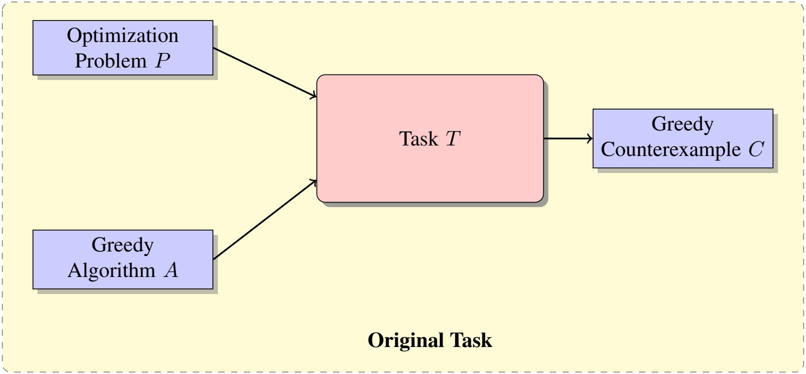


Given a simple graph G output the largest independent set.

Greedy strategy for MIS

1. Pick the vertex with the smallest degree (say v).
2. Delete v and its neighbours from G.
3. Recurse on the remaining graph.

Scope of problems



Definitions

Maximal set

An independent set is said to be *maximal* if we cannot extend it.

Discrepancy of G

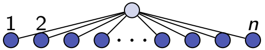
The difference between the largest and the smallest maximal sets in G.

Extremal graphs of interest

Graphs with high discrepancy.

Graphs with low discrepancy.

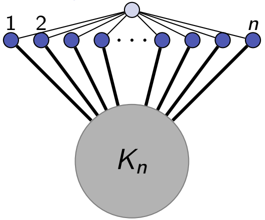
MIS counterexample



Maximum discrepancy graph

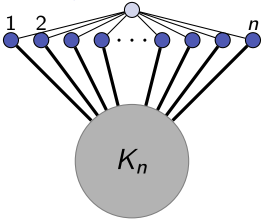
MIS counterexample

Greedy counterexample



MIS counterexample

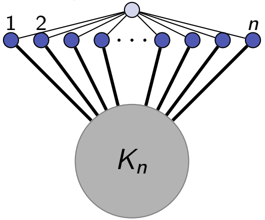
Greedy counterexample



Minimum discrepancy graph

MIS counterexample

Greedy counterexample

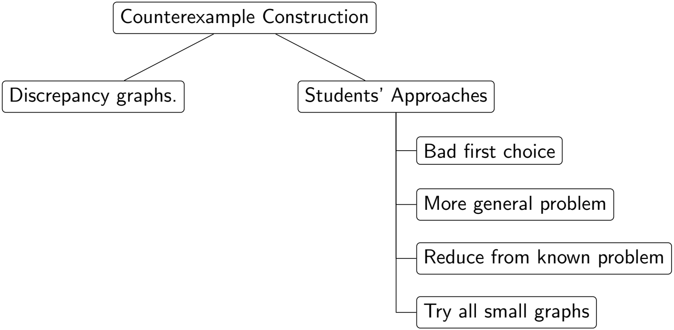


Best Counterexample= Max. + Min. discrepancy graphs!

Applicability

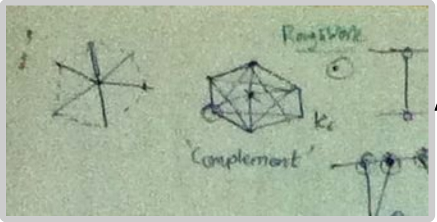
Problem	**St. I**	**St. II**
Independent Set	Star	K_n
Vertex Cover	Star	Centipede
Matching	Paths	$K_{n,n}$
Maxleaf	Path	Binary tree
Maxcut	$K_{n,n}$	$K_{n,n}$
Network Flow	-	Paths
Triangle-Free	-	K_ns
Dominating Set	Paths	Paths

Pilot Experiments



Pilot Experiments

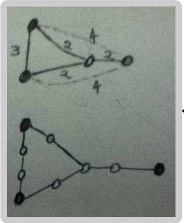
- Counterexample Construction
  - Discrepancy graphs.
  - Students' Approaches
    - Bad first choice
    - More general problem
    - Reduce from known problem
    - Try all small graphs



Pilot Experiments

Counterexample Construction

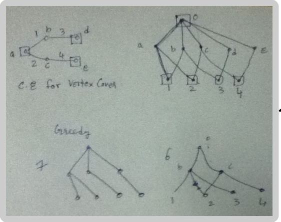
- Discrepancy graphs.



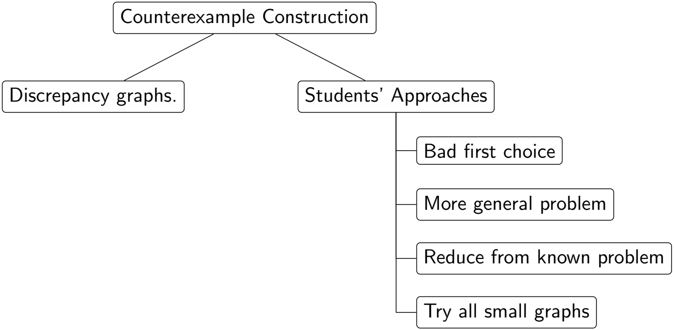
- Students' Approaches
  - Bad first choice
  - More general problem
  - Reduce from known problem
  - Try all small graphs

Pilot Experiments

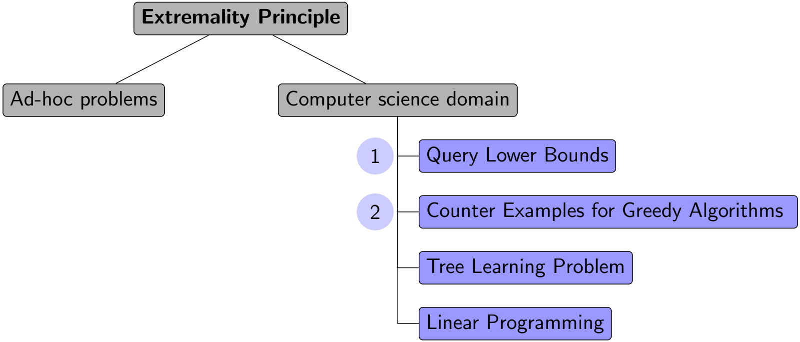
- Counterexample Construction
  - Discrepancy graphs.
  - Students' Approaches
    - Bad first choice
    - More general problem
    - Reduce from known problem
    - Try all small graphs



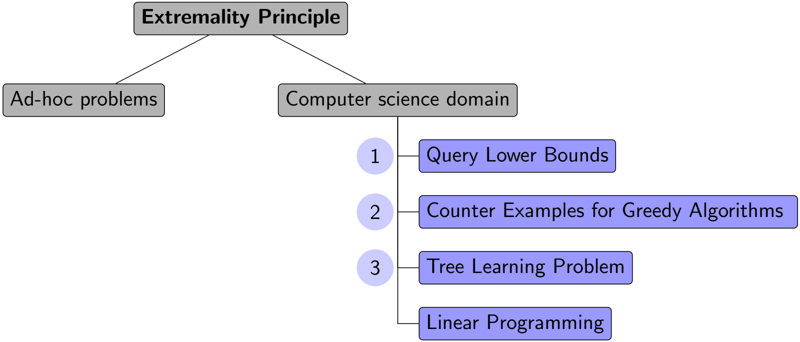
Pilot Experiments



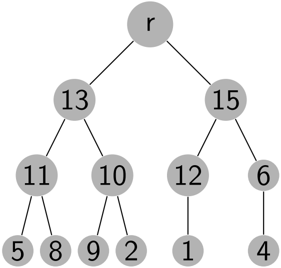
Overview: Tree-learning problem



Overview: Tree-learning problem



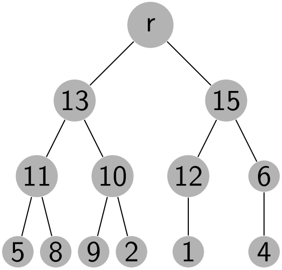
Problem description



Query: 'Is node x a descendant of node y?'

Joint work with Anindya Sen.

Problem description



Result: An $O(n^{1.5} \log n)$ algorithm. Previously, only $O(n^2)$ was known.

Joint work with Anindya Sen.

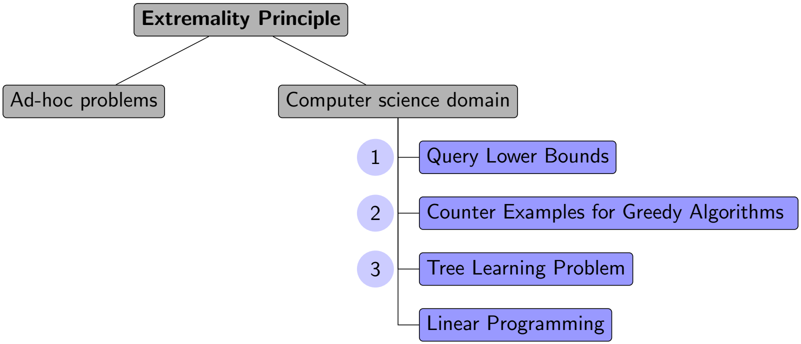
Key ideas via extremal trees

Idea	Extremal Tree	Generalization
I	Path	Bounded-leaves trees
II	Complete binary tree	Short-diameter trees
II	Centipede	Long diameter trees

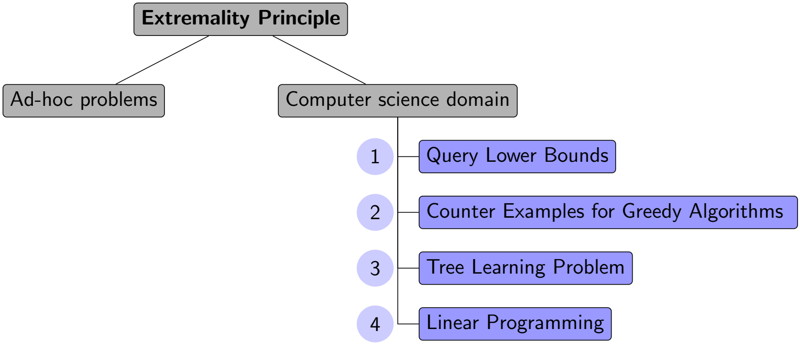
Ideas I+II+II $= O(n^{1.8} \log n)$ algorithm

Joint work with Anindya Sen.

Overview: LPs

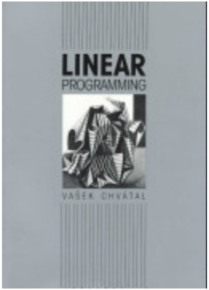


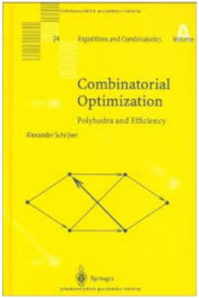
Overview: LPs

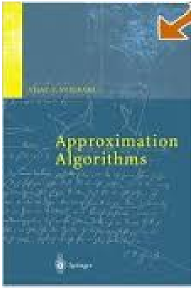


Motivation

Most books on linear programming assume knowledge of linear algebra.







Motivation

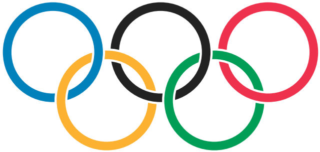
Introduce linear programming to a younger audience using weak duality.

Scope of problems

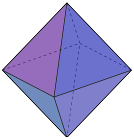
Math puzzles that can be formulated as a linear programs.

Scope of problems

Mathematical Olympiad



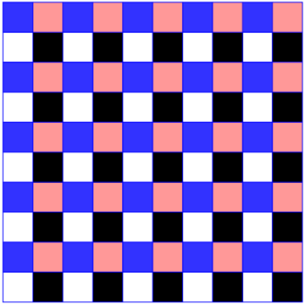
Simple Polytopes



Main Idea: Certificates are easy to find.

Tiling

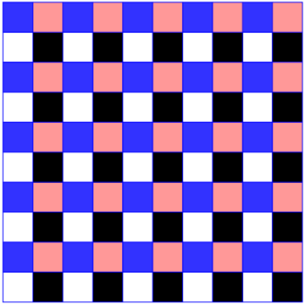
Prove that every 10×10 board cannot be tiled using straight tetraminoes.[1]



[1] F. Ardilla and R.P. Stanley. *Tilings*. Mathematical Intelligencer 2010

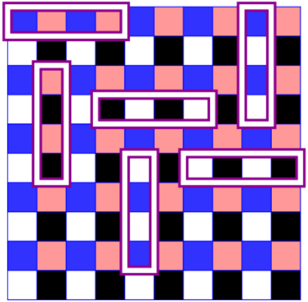
Tiling

Every color appear appears 25 times (odd number).



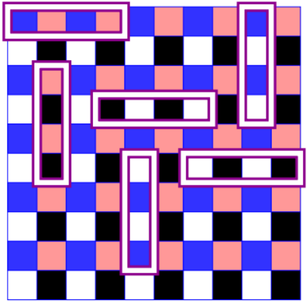
Tiling

Every tile covers even number of colors.



Tiling

Every tile covers even number of colors.



Tiling via LP

What is maximum number of non-overlapping tiles we can place?

LP-formulation. We ensure this by saying that among all the tiles that cover a cell c, at most one should be picked.

$$\text{max: } \sum_{t \in T} t$$

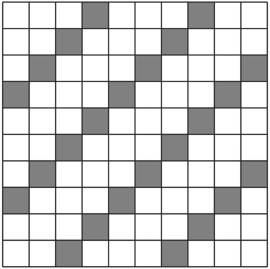
$$\sum_{t \in T_c} t \leq 1 \quad \forall c \in C$$

Tiling - Dual

The dual will have one variable for each cell.

- Minimize the sum of cell-variables.
- Constraint : Sum of every four adjacent cells $>= 1$.

Tiling: Certificate

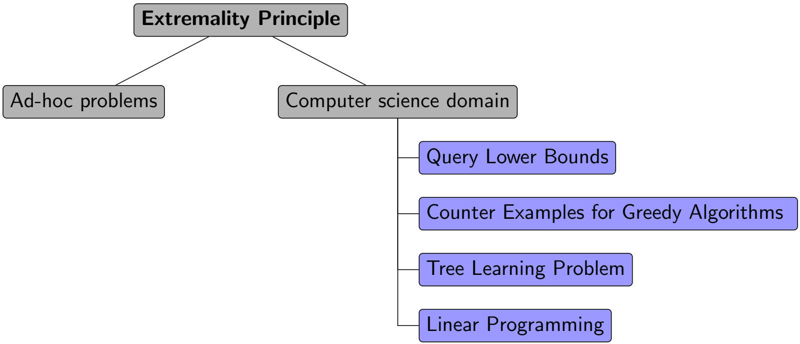


Every tile must lie on exactly one dark cell. But there are only 24 dark cells.

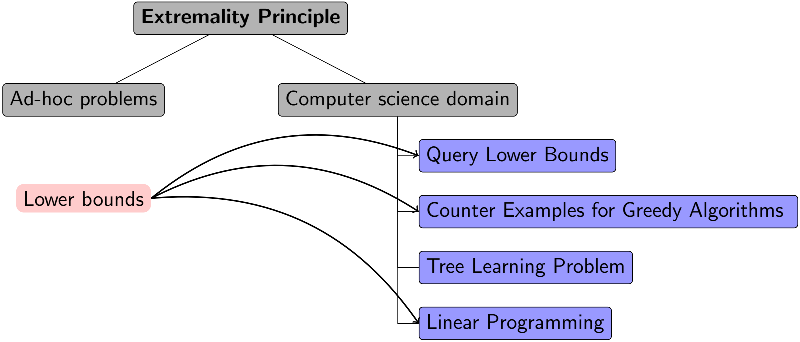
Applicability

Problem	Method
Baltic 2006	Small cases.
Ratio Ineq.	Guess the certificate.
IMO 1965	Guess the tight constraints.
IMO 1977	Guess the certificate.
Engel.	Guess the tight constraints.
Math. Lapok	Small cases.
IMO 2007	Guess the certificate.
IMO 1979	Guess the tight constraints.
Math. Intel.	Small cases.

Common theme



Common theme



Future Work

Counterexamples for online algorithms

Caching problem.

Counterexamples for LP

Instances with large integrality gaps.

Lower bounds in other areas

Number theory: What is a bad instance for Euclid's GCD algorithm?

Thank You

Titu Andreescu

In a set of $2n+1$ numbers the sum of any n numbers is smaller than the sum of the rest. Prove that all numbers are positive.

Titu Andreescu

Arrange the numbers in increasing order $a_1 \leq a_2 \leq \cdots \leq a_{2n+1}$. By hypothesis, we have

$$a_{n+2} + a_{n+3} + \cdots + a_{2n+1} < a_1 + a_2 + \cdots + a_{n+1}.$$

This implies

$$a_1 > (a_{n+2} - a_2) + (a_{n+3} - a_3) + \cdots + (a_{2n+1} - a_{n+1}).$$

Since all the differences in the parentheses are nonnegative, it follows that $a_1 > 0$, and we are done.

(*Középiskolai Matematikai Lapok (Mathematics Gazette for High Schools, Budapest)*)

Duality-based solution

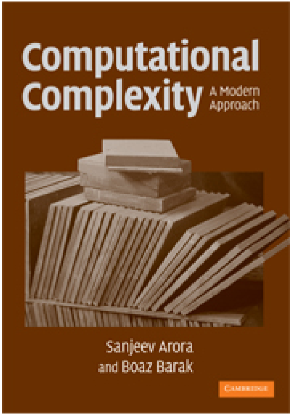
$$x + S > S'$$

$$x + S' > S$$

$$2x + S + S' > S + S'$$

$$x > 0$$

Our Proof vs Textbook Proof



Baseline for comparison.

Experiment

Phase	**Topic**	**Students**				
		S_1	S_2	S_3	S_4	S_5
1*I	Adv.	Yes	Yes	No	No	No
6*III	P1	8m	10m	-	9m	-
	P2	5m	5m	9m	6m	6m
	P3	8m	-	-	10m	9m
	P4	1m	2m	4m	2m	2m
	P5	1m	2m	2m	1m	2m
	P6	1m	1m	1m	1m	1m

Experiment

Phase	Topic	Students				
		S_1	S_2	S_3	S_4	S_5
1*I	Adv.	Yes	Yes	No	No	No
6*III	P1	8m	10m	-	9m	-
	P2	5m	5m	9m	6m	6m
	P3	8m	-	-	10m	9m
	P4	1m	2m	4m	2m	2m
	P5	1m	2m	2m	1m	2m
	P6	1m	1m	1m	1m	1m

Students found our method easier than Arora-Barak ☺

Role of experiments

Pilot Exp. Helped us identify problem areas.

⇓

Refinement of Anchor Method.

Critical Graph vs Invariant Proof

⇓

Pilot Exp. Our method was better than a baseline