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

\( K_6 \)
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

Query Lower Bounds

Counter Examples for Greedy Algorithms

Tree Learning Problem

Linear Programming

Graph-theoretic problems
Overview

Extremality Principle

Ad-hoc problems

Computer science domain

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.

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.

\[\downarrow\text{reduced to}\]

Constructing a certain extremal graph.

Students find the second task easy (Pilot experiment).
Our proof: Main idea

Problem of proving a lower bound.

\[ \text{reduced to} \]

Constructing a certain extremal graph.

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Extremal critical graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connectivity</td>
<td>$2K_n$s</td>
</tr>
<tr>
<td>Triangle-freeness</td>
<td>Star</td>
</tr>
<tr>
<td>Hamiltonicity</td>
<td>$2K_n$s</td>
</tr>
<tr>
<td>Perfect matching</td>
<td>$2K_n$s</td>
</tr>
<tr>
<td>Bipartiteness</td>
<td>$K_{n,n}$</td>
</tr>
<tr>
<td>Cyclic</td>
<td>Path</td>
</tr>
<tr>
<td>Degree-three node</td>
<td>Cycle</td>
</tr>
<tr>
<td>Planarity</td>
<td>Triangulated graph</td>
</tr>
<tr>
<td>Eulerian</td>
<td>Cycle</td>
</tr>
</tbody>
</table>
Overview: Counterexamples

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

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

Task $T$

Optimization Problem $P$

Greedy Algorithm $A$

Greedy Counterexample $C$

Original Task
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

- Optimization Problem $P$
- Greedy Algorithm $A$
- Task $T$
- Greedy Counterexample $C$
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

$K_n$
MIS counterexample

Greedy counterexample

Minimum discrepancy graph
MIS counterexample

Greedy counterexample

Best Counterexample = Max. + Min. discrepancy graphs!
## Applicability

<table>
<thead>
<tr>
<th>Problem</th>
<th>St. I</th>
<th>St. II</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent Set</td>
<td>Star</td>
<td>$K_n$</td>
</tr>
<tr>
<td>Vertex Cover</td>
<td>Star</td>
<td>Centipede</td>
</tr>
<tr>
<td>Matching</td>
<td>Paths</td>
<td>$K_{n,n}$</td>
</tr>
<tr>
<td>Maxleaf</td>
<td>Path</td>
<td>Binary tree</td>
</tr>
<tr>
<td>Maxcut</td>
<td>$K_{n,n}$</td>
<td>$K_{n,n}$</td>
</tr>
<tr>
<td>Network Flow</td>
<td>-</td>
<td>Paths</td>
</tr>
<tr>
<td>Triangle-Free</td>
<td>-</td>
<td>$K_{n,s}$</td>
</tr>
<tr>
<td>Dominating Set</td>
<td>Paths</td>
<td>Paths</td>
</tr>
</tbody>
</table>
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

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
Overview: Tree-learning problem

Extremality Principle

Ad-hoc problems

Computer science domain

1. Query Lower Bounds
2. Counter Examples for Greedy Algorithms
   - Tree Learning Problem
   - Linear Programming
Overview: Tree-learning problem

Extremality Principle

Ad-hoc problems

Computer science domain

1. Query Lower Bounds
2. Counter Examples for Greedy Algorithms
3. Tree Learning Problem
   - Linear Programming
Problem description

Query: ‘Is node $x$ a descendant of node $y$?’

Joint work with Anindya Sen.
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

<table>
<thead>
<tr>
<th>Idea</th>
<th>Extremal Tree</th>
<th>Generalization</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Path</td>
<td>Bounded-leaves trees</td>
</tr>
<tr>
<td>II</td>
<td>Complete binary tree</td>
<td>Short-diameter trees</td>
</tr>
<tr>
<td>II</td>
<td>Centipede</td>
<td>Long diameter trees</td>
</tr>
</tbody>
</table>

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

Joint work with Anindya Sen.
Overview: LPs

- Extremality Principle
  - Ad-hoc problems
  - Computer science domain
    - 1. Query Lower Bounds
    - 2. Counter Examples for Greedy Algorithms
    - 3. Tree Learning Problem
    - Linear Programming
Overview: LPs

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

Main Idea: Certificates are easy to find.
Prove that every $10 \times 10$ board cannot be tiled using straight tetraminoes.\textsuperscript{1}

\textsuperscript{1}F. Ardilla and R.P. Stanley. \textit{Tilings}. Mathematical Intelligencer 2010
Tiling

Every color appear appears 25 times (odd number).
Every tile covers even number of colors.
Tiling

Every tile covers even number of colors.
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.

$$\begin{align*}
\text{max: } & \sum_{t \in T} t \\
\sum_{t \in T_c} t & \leq 1 \quad \forall c \in C
\end{align*}$$
The dual will have one variable for each cell.

- Minimize the sum of cell-variables.
- Constraint: Sum of every four adjacent cells $\geq 1$. 
Every tile must lie on exactly one dark cell. But there are only 24 dark cells.
## Applicability

<table>
<thead>
<tr>
<th>Problem</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baltic 2006</td>
<td>Small cases.</td>
</tr>
<tr>
<td>Ratio Ineq.</td>
<td>Guess the certificate.</td>
</tr>
<tr>
<td>IMO 1965</td>
<td>Guess the tight constraints.</td>
</tr>
<tr>
<td>IMO 1977</td>
<td>Guess the certificate.</td>
</tr>
<tr>
<td>Engel.</td>
<td>Guess the tight constraints.</td>
</tr>
<tr>
<td>Math. Lapok</td>
<td>Small cases.</td>
</tr>
<tr>
<td>IMO 2007</td>
<td>Guess the certificate.</td>
</tr>
<tr>
<td>IMO 1979</td>
<td>Guess the tight constraints.</td>
</tr>
</tbody>
</table>
Common theme

Extremality Principle

Ad-hoc problems

Computer science domain

Query Lower Bounds

Counter Examples for Greedy Algorithms

Tree Learning Problem

Linear Programming
Common theme

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

(Kőzépiskolai 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

<table>
<thead>
<tr>
<th>Phase</th>
<th>Topic</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>S₁</td>
</tr>
<tr>
<td>1*I</td>
<td>Adv.</td>
<td>Yes</td>
</tr>
<tr>
<td>6*III</td>
<td>P1</td>
<td>8m</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>5m</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>8m</td>
</tr>
<tr>
<td></td>
<td>P4</td>
<td>1m</td>
</tr>
<tr>
<td></td>
<td>P5</td>
<td>1m</td>
</tr>
<tr>
<td></td>
<td>P6</td>
<td>1m</td>
</tr>
</tbody>
</table>
## Experiment

Students found our method easier than Arora-Barak 😊

<table>
<thead>
<tr>
<th>Phase</th>
<th>Topic</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$S_1$</td>
</tr>
<tr>
<td>1*I</td>
<td>Adv.</td>
<td>Yes</td>
</tr>
<tr>
<td>6*III</td>
<td>P1</td>
<td>8m</td>
</tr>
<tr>
<td></td>
<td>P2</td>
<td>5m</td>
</tr>
<tr>
<td></td>
<td>P3</td>
<td>8m</td>
</tr>
<tr>
<td></td>
<td>P4</td>
<td>1m</td>
</tr>
<tr>
<td></td>
<td>P5</td>
<td>1m</td>
</tr>
<tr>
<td></td>
<td>P6</td>
<td>1m</td>
</tr>
</tbody>
</table>
Role of experiments

- **Pilot Exp.** Helped us identify problem areas.
- **Critical Graph vs Invariant Proof**
- **Refinement of Anchor Method.**
- **Pilot Exp.** Our method was better than a baseline