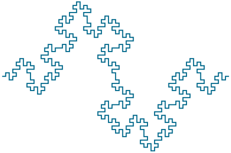
Σ_2^P: Multilevel Programming, Preprocessing and Counterexamples

Theja Tulabandhula
Massachusetts Institute of Technology



Based on Johannes' PhD thesis: Ch 3,4,5.

October 24, 2011

Three sets of problems and Σ_2^P-C

Background

Multilevel Programming

Preprocessing Problems

Counterexamples to Conjectures

Running theme: We will exploit properties of NP-C transformations to show completeness or hardness at the second level.

BACKGROUND

- Σ_2^P lies one level above NP: contains all problems that can be efficiently solved by a non deterministic algorithm having access to a NP oracle. Each call to the oracle is considered one computational step.
- P, NP, coNP are at the bottom of the PH
- The next most interesting class is Σ_2^P
- Proving Σ_2^P-C is more interesting than just proving NP-hard.
- less abundant natural complete problems than NP
- If a complete problem is efficiently solvable, so are all the other members of the class.
- Σ_2^P-C gives us a sense of which problems may not be solved in (deterministic) polynomial time even if one had access to a NP oracle.

- First problem to be shown Σ_2^P-C is the 2-ALTERNATING QUANTIFIED SATISFIABILITY (B_2) by Meyer and Stockmeyer'72.
  - Instance: Two sets of boolean variables X and Y. DNF expression E.
  - Question: Is there a truth assignment to X such that for all truth assignments to Y, E is satisfied?
- Equivalent: CNF E and ask: is there a truth assignment to X such that for all assignments to Y, E is not satisfied. Defines B_2^{CNF}.
- Practical Σ_2^P-C problem (VLSI Design): DNF MINIMIZATION
  - Given a boolean DNF formula and integer K, is there an equivalent DNF formula with at most k occurence of literals.

- Families of problems are Σ_2^P-C (Johannes' thesis). Technique based on established NP-C transformations.
- Example 4 classes of problems listed below:
  1. Adversarial Problems
  2. Multilevel Programming
  3. Preprocessing Problems
  4. Counterexamples to Conjectures
- Adversarial Problems: Every problem in this class is based on a combinatorial feasibility problem in NP and can be formulated using $0-1$ variables.
- Consists of splitting the variable set into two sets X and Y
- Asks: Is there an assignment to X so that it is not possible to complete this assignment to a feasible solution no matter what we assign to Y.
- Example: B_2^{CNF}.

- Under conditions, a poly time transform from SAT to another problem Π can be used to derive a poly time transformation from B_2^{CNF} to the adversarial version of Π.
- Thus, if adversarial version of Π is in Σ_2^P, then it is also Σ_2^P-C.
- Multilevel Programming In k-level programming, k levels with own set of variables.
- levels sequentially choose their variables, knowing objectives of lower levels.
- Bilevel integer programming and trilevel linear programming are Σ_2^P-C.

- Preprocessing Problems Given combinatorial optimization problem with flexible objective, ask if $\exists$ objective such that a particular element becomes a part of the optimal solution.
- If in the wiggle room, no objective allows the element, the element can be eliminated. Hence called "preprocessing".
- Applications: faster solutions.
- Counterexamples to Conjectures Explain why difficult to disprove conjectures using Σ_2^P theory.
- Finding counterexamples is Σ_2^P-C if deciding the existence of certain related objects is NP-C.

Notation

- Decision problem Π. Set of instances D_Π. Yes instances $Y_\Pi \subseteq D_\Pi$.
- Decision problem: Whether a given instance is a yes instance or not.
- Deterministic algorithm solves Π if
  - it halts $\forall I \in D_\Pi$
  - Returns “YES” iff $I \in Y_\Pi$ and “NO” otherwise.
  - If number of steps polynomial in input size, then polynomial time deterministic algorithm.
- Class P: class of Π for which there is a polynomial time deterministic algorithm that solves Π.

- nondeterministic algorithm has 2 parts: guessing and checking. Solves $I \in D_\Pi$
  - if $I \in Y_\Pi$, then there is a certificate S when guessed will lead to answering "YES".
  - if $I \notin Y_\Pi$, then there is no certificate S when guessed will lead to answering "YES".
  - said to operate in P time if deterministic checking works polynomial to the input size while answering "YES".
- Polynomial transformation $f : D_{\Pi_1} \to D_{\Pi_2}$ is such that
  - there is a P time deterministic algorithm that computes f
  - I is a "YES" instance in D_{Π_1} iff $f(I)$ is a "YES" instance in D_{Π_2}
  - is transitive
- Decision problem is complete for a class $\mathcal{C}$ (wrt polynomial transformability) if there is a f mapping to Π from every $\Pi' \in \mathcal{C}$

- Polynomial hierarchy (PH).
  - $\forall k \geq 1, \quad \Sigma_k^p = NP^{\Sigma_{k-1}^p}$
  - $\Pi_k^p = \mathrm{co}\Sigma_k^p$
  - $\Sigma_0^p = \Pi_0^p = \mathrm{P}$
  - $\Sigma_1^p = \mathrm{NP}, \Pi_1^p = \mathrm{coNP}$
  - $\Sigma_k^p \subseteq \Sigma_{k+1}^p, \Pi_k^p \subseteq \Pi_{k+1}^p, \Sigma_k^p \subseteq \Pi_{k+1}^p$ and $\Pi_k^p \subseteq \Sigma_{k+1}^p$
  - $PH = \cup_{k \in \mathbb{N}} \Sigma_k^p$
- Each class is closed under polynomial transformation in PH. Thus, can assume the oracle solves a complete problem.
- k-ALTERNATING QUANTIFIER SAT B_k is Σ_k^p-C.
- PH $\subseteq$ PSPACE.

- B_k
  - Instance: boolean expression E over set $X = \{x_{ij}, i = 1, ..., k, j = 1, ..., m_i\}$
  - Question: Does

$$\begin{aligned}
\exists x_{11}, ..., x_{1m_1} \\
\forall x_{21}, ..., x_{2m_2} \\
\vdots \\
Q x_{k1}, ..., x_{km_k} E
\end{aligned}$$

    where $Q = \exists$ if k is odd and $\forall$ otherwise.
  - $k = 1$ gives SAT which is NP-C
  - $k = 2$ gives B_2 which is Σ_2^p-C.
- B_k is Σ_k^p-C.
- B_k is PSPACE-C if k is allowed to be unbounded.

MULTILEVEL PROGRAMMING

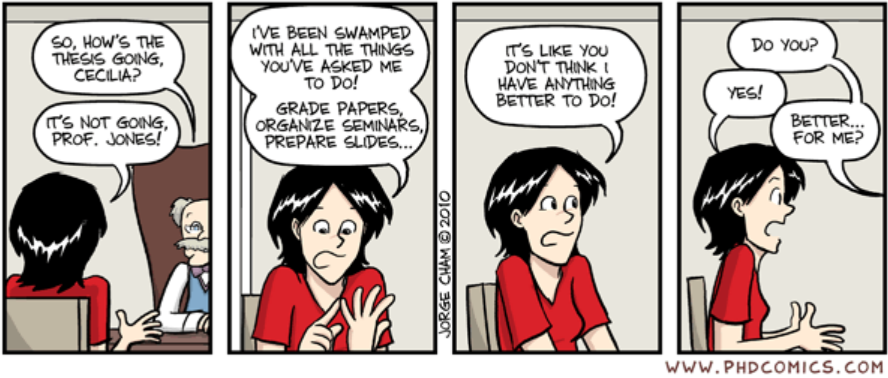
Hierarchical Structure of Decisions

- There is a hierarchy.
- Every level
  - makes decisions.
  - controls a subset of variables.
- Lower levels may have different objectives than the higher ones.
- How do they coordinate?

Many applications

Hierarchical decisions are involved in network design, toll setting, revenue management, utility planning etc.

An example from PhD life



- The professor optimizes his objective and sets certain variables knowing student will optimize her objective.

Each level fixes a subset of variables

- Assume lower level objectives known to higher levels.
- The process:
  - Highest level fixes her set of variables.
  - Next highest fixes hers.
  - ...
  - Lowest level optimizes with higher level variables appearing as constants.
- The optimization:
  - Higher levels optimize knowing lower levels will optimize later.
  - They might incorporate anticipated responses while fixing variables.
  - Lower levels not dictated by higher levels.
  - They are influenced by fixed variables set by higher levels.

Bilevel programming

- 1^{st} level influences but does not control actions of the 2^{nd}.
- 1^{st} level sets variables x first and then 2^{nd} level sets y.
- Shared feasible region $\{(x, y) \geq 0 : Ax + By \leq b\}$
- Bilevel linear program:

$$\begin{aligned} &\min_{x \geq 0} c_1 x + d_1 y \\ &\text{where } y \text{ solves } \max_{y \geq 0} d_2 y \\ &\qquad\qquad\qquad \text{such that } Ax + By \leq b \end{aligned}$$

- Bilevel integer program: integer (x, y).
- Solution: specified by higher level variable (x).

Decision problems: BiLP, BiIP, TriLP

- BiLP
  - Instance: Bilevel linear program, rational K.
  - Question: Is there a solution such that objective of highest level $\leq K$?
  - NP-hard (1985). Strongly NP-Hard (1992).
- BiIP
  - Instance: Bilevel integer program, integer K.
  - Question: Is there a solution such that objective of highest level $\leq K$?
- TriLP
  - Instance: Trilevel linear program, rational K.
  - Question: Is there a solution such that objective of highest level $\leq K$?

THEOREM: {0-1} BIIP IS Σ_2^P-HARD

- is in Σ_2^P.
- Polynomial transformation: $I \in D_{B_2^{CNF}}$ to $f(I) \in D_{BiIP}$.
  - 3 binary variables x for higher and y, z for lower in f(I). $x \leftarrow$ boolean X of B_2^{CNF}. $y \leftarrow$ boolean Y of B_2^{CNF}.
  - For every clause $c_j \in C$ introduce z_j. Example:

$$c_j = (x_1 \vee \bar{x}_2 \vee y_1) \Rightarrow x_1 + (1 - x_1) + y_1 \geq z_j$$

    If c_j satisfied then $z_j = 1$; 0 otherwise.
  - Higher objective: $\min_x \sum_{j=1}^{|C|} z_j$. Lower: $\max_{j,z} \sum_{j=1}^{|C|} z_j$.
  - Lower level trying to maximize clauses satisfied, higher tries minimizing.
  - The objective of higher is less than $|C|$ if and only if I is a "YES" instance.

THEOREM: TRILP IS Σ_2^P-HARD.

- Is in Σ_2^P since BiLP is NP-hard.
- formulate ADV-PRT[1] as BiIP $\overset{\text{pol. trans.}f}{\rightarrow}$ TriLP.
- Step 1
  - ADV-PRT is Σ_2^P-C. Given $X, Y, \{l(a); a \in X \cup Y\}$, does $\exists X^* \subseteq X$ so that $\forall Y^* \subseteq Y, \sum_{a \in X^* \cup Y^*} l(a) \neq \sum_{a \in \overline{X^* \cup Y^*}} l(a)$
  - Let $L = \frac{1}{2} \sum_{a \in X \cup Y} l(a)$. Let $I \in D_{ADV-PRT}$.
  - Then a corresponding bilevel integer program (■) is:

$$\min_{x \in \{0,1\}^X} \sum_{a \in X} x_a l(a) + \sum_{a \in Y} y_a l(a)$$

$$\text{where y solves } \max_{y \in \{0,1\}^Y} \sum_{a \in X} x_a l(a) + \sum_{a \in Y} y_a l(a)$$

$$\text{such that } \sum_{a \in X} x_a l(a) + \sum_{a \in Y} y_a l(a) \leq L$$

[1] ADVERSARIAL PARTITION

- Two observations
  - Assume $\sum_{a \in X} x_a l(a) \leq L$.
  - If I is yes instance, then $\sum_{a \in X^*} x_a l(a) \neq L$.
- Step 2
- Lemma 1: The optimal objective of higher level in the integer program (■) is $< L$ if and only if $I \in D_{ADV-PRT}$ is a "YES" instance.
  - ($\Rightarrow$) Let $I \in D_{ADV-PRT}$ be a "YES" instance. $\exists X^*$.
  - Consider the $0-1$ assignment of $f(I) \in D_{BiLP}$ with $x_a = 1$ if $a \in X^*$ and 0 otherwise.
  - If lower level succeeds in getting an objective equal to L, picking a corresponding to $y_a = 1$ leads to contradiction.
  - ($\Leftarrow$) Let $I \in D_{ADV-PRT}$ be a "NO" instance. $\forall X^*, \exists Y^*$ such that $\sum_{a \in X^* \cup Y^*} l(a) = L$. Set $y_a = 1$ if $a \in Y*$. Then, $\sum_{a \in X} x_a l(a) + \sum_{a \in Y} y_a l(a) = L$.
- Second Proof that $\{0-1\}$ BiIP is Σ_2^P-C. Previous one was from B_2^{CNF}.

- Step 3
  - Transform (■) to a trilevel linear program (new level with slack variables s)
  - s relate to distance of $\{x_a, y_a\}$ to nearest integers.
  - Let $M = (\max_{a \in X \cup Y} l(a))^2$,
    $T_1(x,y) = \sum_{a \in X} x_a l(a) + \sum_{a \in Y} y_a l(a)$ and $T_2(s) = \sum_{a \in X \cup Y} s_a$.

$$
\begin{aligned}
&\min_x T_1(x,y) + M\, T_2(s) \\
&\qquad \text{where } y \text{ solves } \max_y T_1(x,y) - M\, T_2(s) \\
&\qquad\qquad \text{where } s \text{ solves } \max_s T_2(s) \\
&\qquad\qquad \text{such that } T_1(x,y) \le L \\
&\qquad\qquad 0 \le x_a \le 1 \ \ \forall a \in X; 0 \le y_a \le 1 \ \ \forall a \in Y \\
&\qquad\qquad s_a \le x_a, s_a \le 1 - x_a \ \forall a \in X; \\
&\qquad\qquad s_a \le y_a, s_a \le 1 - y_a \ \forall a \in Y.
\end{aligned}
$$

- Lemma 2: For every feasible (x, y, s), which contains at least one fractional in x or y, there is a feasible (x^*, y^*, s^*) with one less fractional component with objectives of top two levels being same or better. Thus the optimal solution to trilevel program is integral.
  - W.L.O.G $M \geq 4$. Go by cases.
  - Case (a): If a frac. component $\leq 1 - 1/\sqrt{M}$, then set it to 0 to get (x^*, y^*, s^*).
  - Case (b): If two components are $> 1 - 1/\sqrt{M}$ pick new values such that "contribution" to $T_1(x, y)$ is the same and one of them is 1 in (x^*, y^*, s^*).
  - Case (c): If only one frac component $> 1 - 1/\sqrt{M}$, set it to 1 in (x^*, y^*, s^*).
- Proof omitted.

- Step 4
  - In cases (a) and (c), objectives of both top level players become strictly better.
  - No fractional solution can become integral by just case (b)
  - Thus, need to use case (a) or (c)
  - Since each player is selfish, the optimal integer solution will be attained by the program.
- Summary:
  - ADV-PRT to BiIP one to one.
  - Bilevel integer to trilevel linear such that if latter feasible, will achieve optimal integral solution.

PREPROCESSING PROBLEMS

Preprocessing

- Related to partial inverse optimization problems (introduced by Orlin).
  - Requires minimal adjustment of objective to make given partial feasible solution optimal.
- In preprocessing,
  - Only 1 element given
  - Requires finding whether this is part of an optimal solution.
  - Assume objective is not fully known.
  - Our case: bounds on each component of objective given.
- What we will see: Preprocessing problems associated with many NP-C problems are Σ_2^P-C.

PROBLEM

- Given
  - Ground set Z
  - $\mathcal{F}$: Collection of subsets of Z
  - Problem: $\min_{S\in\mathcal{F}} \sum_{j\in S} c_j$
- Question: For input $e \in Z$, integer vectors l, u is there
  - an integer cost vector $l \leq c' \leq u$, and
  - a solution $S' \in \mathcal{F}$

  such that $e \in S'$ and S' optimal to the problem $\min_{S\in\mathcal{F}} \sum_{j\in S} c'_j$.
- If answer is no, eliminate e and simplify problem.
- Applications: Real time optimization problems

Preprocessing Satisfiability

- Recall B_2^{CNF}.
- PP 3SAT: PREPROCESSING 3SAT
  - Instance: Boolean set U, $\{(l(z), u(z)) \;\; \forall z \in U\}$. CNF expression E. $z^* \in U$.
  - Question: Does $\exists$ a truth assignment S_U with $S_U(z^*) = 1$ and a cost vector c of length $|U|$ with $l(z) \leq c(z) \leq u(z)$ such that $S_U \in \text{argmin}_{\{S:\text{satisfiable}\}} \sum_{z \in U} c(z)S(z)$?

Theorem

PP 3SAT is Σ_2^P-C.

- Proof via $B_2^{CNF} \rightarrow$ PP 4SAT $\rightarrow$ PP 3SAT.

PREPROCESSING VERTEX COVER

- PP-VC: PREPROCESSING VERTEX COVER.
  - Instance: $G = (V, E), v^* \in V, \{(l(v), u(v)) \ \ \forall v \in V\}$ and integer K.
  - Question: Is there a cost vector c of size $|V|$ with $l(v) \leq c(v) \leq u(v)$ and a vertex cover of size at most K which includes v^* and is minimal with respect to c?

Theorem

PP-VC is Σ_2^P-C.

- A polynomial transformation from PP 3SAT to PP-VC shows latter is Σ_2^P-C.

- Finally, note that Preprocessing versions of 3DM and HC are also Σ_2^P-C.

COUNTEREXAMPLES TO CONJECTURES

COUNTEREXAMPLES TO NP CONJECTURES

- Characterization of graph properties that are NP-C to decide.
  - Example: G has a hamiltonian cycle.
- Property $\mathcal{P}$ is a sufficient condition for G to have hamiltonian cycle if, every G with $\mathcal{P}$ is hamiltonian.
  - Example: If G 2-connected, and if $\text{dist}(u, v) = 2$, then either u or v has degree $\geq |V|/2$, then G has a hamiltonian cycle.

Conjecture I (refuted, 46 node counterexample)

Every 3-connected planar graph has a hamiltonian cycle.

Conjecture II (open)

Every 4-connected line graph is hamiltonian.

Complexity of refuting conjectures

- concerning NP-C graph properties.
- Concrete example: hamiltonicity of graphs.
    - How hard in complexity theoretic sense, is it to find a counterexample to a conjecture that suggests a sufficient condition for a graph to be hamiltonian?
- Let $\mathcal{P}$ be some property and Conj.$(\mathcal{P}, HC)$ be the conjecture.
- Instance: Property $\mathcal{P}$
- Question: Is there a counterexample to conjecture Conj.$(\mathcal{P}, HC)$?
    - Input size?
    - relate the graphs of above instance to an instance of some decision problem.
    - Every I gives $\mathcal{P}_I$ and related graphs are poly(size(I)).

Complexity of refuting conjectures

Theorem
COUNTEREXAMPLE HC (CExplHC) is Σ_2^P-C.

- $B_2^{CNF} \rightarrow$ CExplHC.

- Given this result, appreciate the hardness of coming up with counterexamples for conjectures for Hamiltonicity.
- Similar result for counterexamples to vertex cover conjecture.

Summary

- Classes of Problems we did not look at:
  - Defining set problems (can the size-restricted partial solution be completed in a unique way)
  - Cost denying set problem (existence of partial solution of limited cost which cannot be extended to a feasible solution)
    Defining set and cost denying set problems are Σ_2^p-C.
  - Minimum integer programming equivalence is also Σ_2^p-C.
- We looked at Multilevel programming, preprocessing and counterexamples to conjectures problem and established their completeness in a couple of cases.