9^{th} Week

Computational Optimization Problems and Uniform Circuits

Synopsis.

- Computational Optimization Problems
- NLO, NL, APXL, NC^1O, and AC^0O
- Uniform Circuit Families
- Circuit Complexity

June 4, 2018. 23:59

Course Schedule: 16 Weeks

Subject to Change

- Week 1: Basic Computation Models
- Week 2: NP-Completeness, Probabilistic and Counting Complexity Classes
- Week 3: Space Complexity and the Linear Space Hypothesis
- Week 4: Relativizations and Hierarchies
- Week 5: Structural Properties by Finite Automata
- Week 6: Stype-2 Computability, Multi-Valued Functions, and State Complexity
- Week 7: Cryptographic Concepts for Finite Automata
- Week 8: Constraint Satisfaction Problems
- Week 9: Combinatorial Optimization Problems
- Week 10: Average-Case Complexity
- Week 11: Basics of Quantum Information
- Week 12: BQP, NQP, Quantum NP, and Quantum Finite Automata
- Week 13: Quantum State Complexity and Advice
- Week 14: Quantum Cryptographic Systems
- Week 15: Quantum Interactive Proofs
- Week 16: Final Evaluation Day (no lecture)

YouTube Videos

- This lecture series is based on numerous papers of T. Yamakami. He gave conference talks (in English) and invited talks (in English), some of which were video-recorded and uploaded to YouTube.
- Use the following keywords to find a playlist of those videos.
- YouTube search keywords:

Tomoyuki Yamakami conference invited talk playlist

Conference talk video

Main References by T. Yamakami

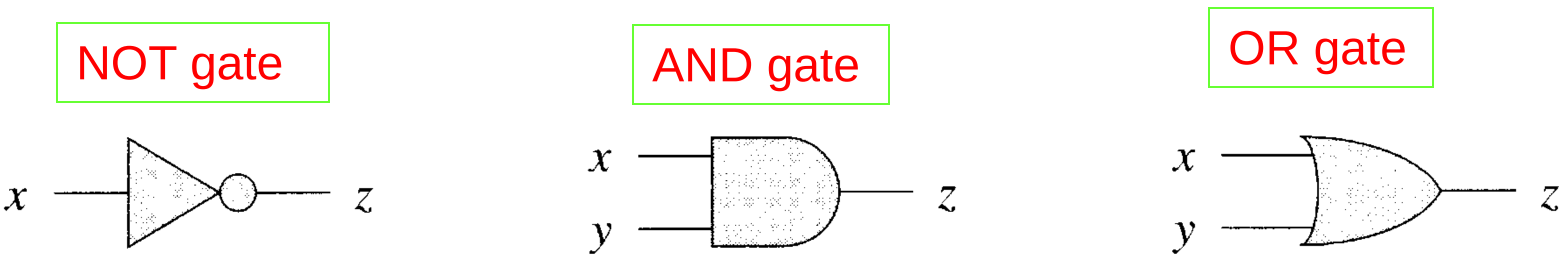
- ✎ T. Yamakami. Optimization, randomized approximability, and Boolean constraint satisfaction problems. In Proc. of ISAAC 2011, Lecture Notes in Computer Science, vol. 7074, pp. 454-463 (2011)
- ✎ T. Yamakami. Uniform-circuit and logarithmic-space approximations of refined combinatorial optimization problems. In Proc. of COCOA 2013, Lecture Notes in Computer Science, vol. 8287, pp. 318-329 (2013). A complete version is available at arXiv:1601.01118.

I. Uniform Circuit Families

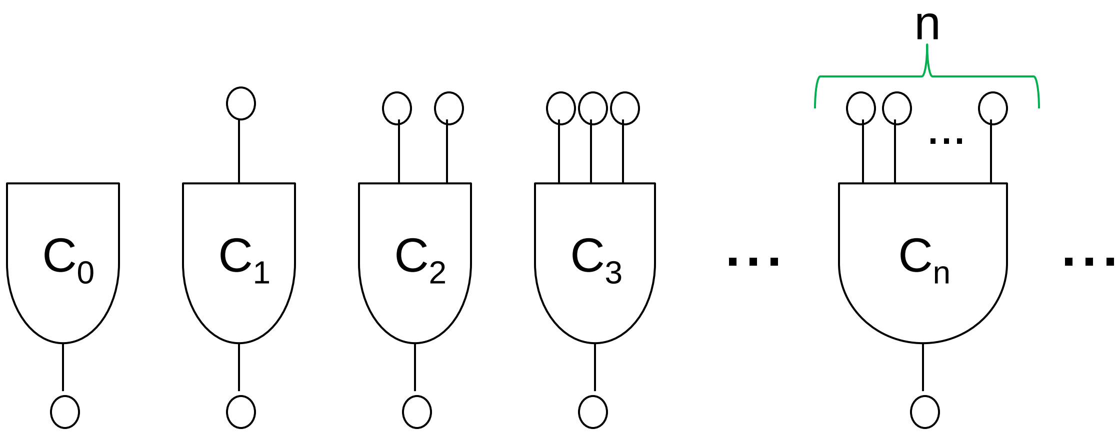
1. Families of Boolean Circuits
2. Circuit Complexity measures
3. Fan-in of Circuits
4. Uniform Families of Circuits
5. Complexity Classes AC^k and NC^k

Families of Boolean Circuits (revisited)

- Recall that each Boolean circuit is composed of the following logical gates and wires (or edges).

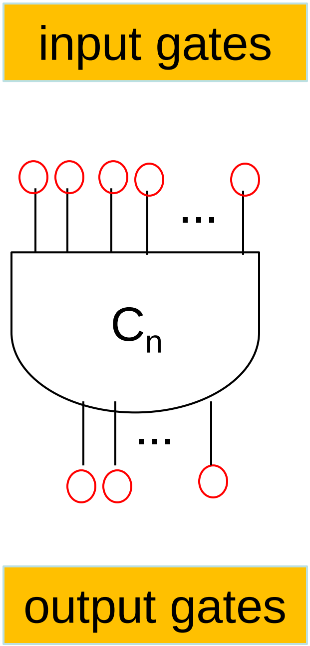


- In a family $\{C_n\}_{n \in N}$ of Boolean circuits, each C_n is a Boolean circuit taking n-bit inputs.



Circuit Complexity Measures (revisited)

- Recall that we treat "inputs" as input gates, which are technically in-degree-0 nodes, and treat "outputs" as output gates, which are out-degree-0 nodes.
- For circuits, we usually use the following complexity measures.
- Circuit complexity measures:
  - size of circuit C = number of gates in C
  - depth of circuit C = number of logical gates in the longest path from an input to an output



Fan-in of Circuits

- For simplicity, we often consider Boolean circuits with AND and OR gates but not NOR gates.
- Thus, input gates are labeled by literals (i.e., variables or the negation of variables).
- To cope with decision problems (i.e., languages), we are interested in circuits that have only one output gate.

- We say that a circuit C has bounded fan-in if all AND and OR gates used in C are of in-degree 2.
- A circuit is said to have unbounded fan-in if its AND and OR gates may have an arbitrary number of in-coming edges.

Uniform Families of Circuits

- In Week 3, we have already discussed the notion of non-uniformity. Here, we consider its opposite notion: uniformity.
- There are numerous concepts of uniformity in use to describe different collections of circuit families.
- Here, we use logarithmic-space (or L) uniformity.
- Other uniformity concepts in use include "P-uniform" and "DLOGTIME-uniform."

- A family $\{ C_n \}_{n \in N}$ of circuits is said to be logarithmic-space uniform (log-space uniform or L-uniform) if there exists a log-space DTM such that, for any length parameter $n \in N$,
  - ✓ on input 1^n, M produces an encoding $\langle C_n \rangle$ of C_n.

Complexity Classes AC^k and NC^k

- Let us define circuit complexity classes. Let $k \in \mathbb{N}$.
- NC^k = class of languages recognized by log-space uniform families of circuits, each C_n of which has polynomial-size, $O(\log^k(n))$-depth, and bounded fan-in.
- NC^k is known as Nick's class.
- AC^k = class of languages recognized by log-space uniform families of circuits, each C_n of which has polynomial-size, $O(\log^k(n))$-depth, and unbounded fan-in.

- (Claim) $AC^k \subseteq NC^{k+1}$ for any $k \geq 0$.
- (Claim) $AC^0 \neq NC^1$. [Yao (1985), Håstad (1987)]

Open Problems

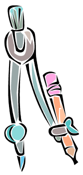
- There are numerous open problems associated with circuit families.
- Is $AC^k \neq NC^{k+1}$ for any $k \geq 1$?
- SAC^k = languages recognized by L-uniform families of $O(\log^k(n))$-depth, polynomial-size, semi-unbounded fan-in (i.e., all AND gates have in-degree 2) circuits
- Recall the CFL hierarchy $\{\Delta_k^{CFL}, \Sigma_k^{CFL}, \Pi_k^{CFL} \mid k \geq 1\}$ from Week 4.
  - It is known that, for example, $AC^0(\Sigma_1^{CFL}) = SAC^1$.
  - Find more relationships between Σ_{k+1}^{CFL} and circuit complexity classes, such as SAC^{k+1}.

II. NP Optimization Problems

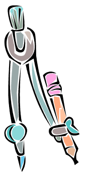
1. Combinatorial Optimization Problems
2. NP Optimization Problems
3. NPO and PO_{NPO}
4. Performance Ratios
5. Approximation Schemes
6. APXP
7. Approximation-Preserving (APP) Reductions
8. Completeness by APP-Reductions
9. A Map of Complete Problems

Combinatorial Optimization Problems

- Optimization problems are found everywhere and they have been discussed in theory and in practice.
- A combinatorial optimization problem P is defined as a tuple (I, SOL, m, goal), where
  - I = the set of input instances;
  - SOL(x) = a set of (feasible) solutions associated with instance x;
  - m: objective function (or measure function) mapping $I \times SOL(x)$ to $R^{\geq 0}$; and
  - goal $\in$ { max, min }.
- Let $m^*(x) = goal\{ m(x,y) \mid y \in sol(x) \}$.
- y is an optimal solution w.r.t. $x \leftrightarrow m(x,y) = m^*(x)$



NP Optimization Problems I



- We are interested in NP optimization problems.
- An NP optimization problem (or an NPO problem) is a combinatorial optimization problem $P = (I, SOL, m, goal)$ satisfying the following extra conditions:
  - the set I is recognized in polynomial time,
  - there are a polynomial p such that, for any $x \in I$ and for any $y \in SOL(x)$, $|y| \leq p(|x|)$; moreover, for any y with $|y| \leq p(|x|)$, it is decidable in polynomial time whether $y \in SOL(x)$, and
  - m is computable in polynomial time.
- If goal = max, then P is a maximization problem; otherwise, P is a minimization problem.

NP Optimization Problems II

- Many NP problems can be turned into NP optimization problems. Here, we see one simple example.
- Partition Problem (decision problem)
  - instance: a finite set A of items and a weight function $w: A \rightarrow N^+$
  - question: is there any partition X,Y of A such that

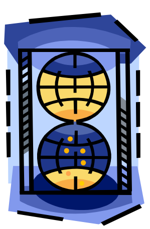
    $$\Sigma_{x \in X} w(x) = \Sigma_{y \in Y} w(y)?$$
- Minimum Partition Problem (optimization problem)
  - instance: a finite set A of items and a weight function $w: A \rightarrow N^+$
  - solution: a partition X,Y of A
  - measure: $\min\{ \Sigma_{x \in X} w(x), \Sigma_{y \in Y} w(y) \}$

NPO and PO_{NPO}

- In a polynomial-time setting, two typical classes of optimization problems are discussed.

- NPO = a class of NP optimization problems
- PO_{NPO} (or PO) = a class of polynomial-time solvable NP optimization problems

- (Claim) $PO_{NPO} \subseteq NPO$



Performance ratios

- Performance ratio

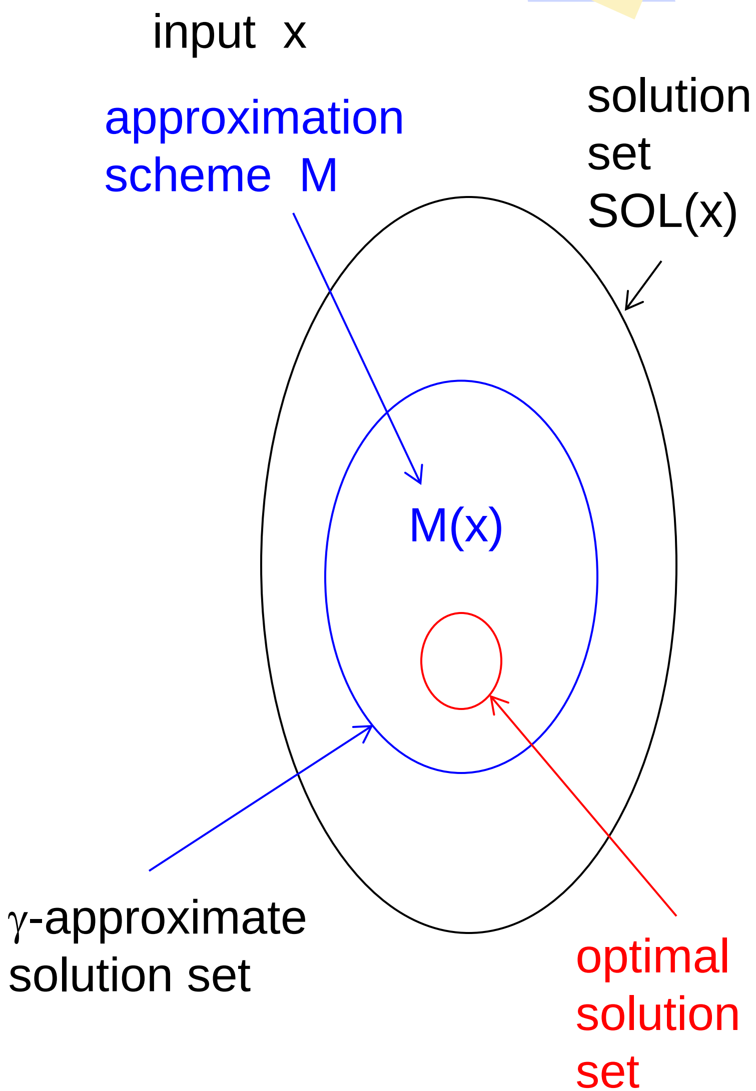
$$R(x,y) = \max\left\{ \left| \frac{m(x,y)}{m^*(x)} \right|, \left| \frac{m^*(x)}{m(x,y)} \right| \right\},$$

where $m^*(x), m(x,y) \neq 0.$

- Consider a machine M approximating x. In this case,

$$R(x, M(x)) \leq \gamma \quad \Leftrightarrow \quad \frac{x}{\gamma} \leq M(x) \leq \gamma x$$

- Note that m(x,y) = m*(x) $\leftrightarrow$ R(x,y) = 0 $\leftrightarrow$ y is optimal.



Approximation Schemes

- We define approximation algorithms or schemes.
- Let P = (I, SOL, m, goal) be any optimization problem.

- An algorithm M is said to be a γ-approximate algorithm
  $\Leftrightarrow$ $\forall x \in I$ [$R(x,M(x)) \leq \gamma$].

- P is polynomial-time γ-approximable
  $\Leftrightarrow$ $\exists$M polynomial-time DTM s.t. $\forall x \in I$ [$R(x,M(x)) \leq \gamma$].

APXP_{NPO} (or APX)

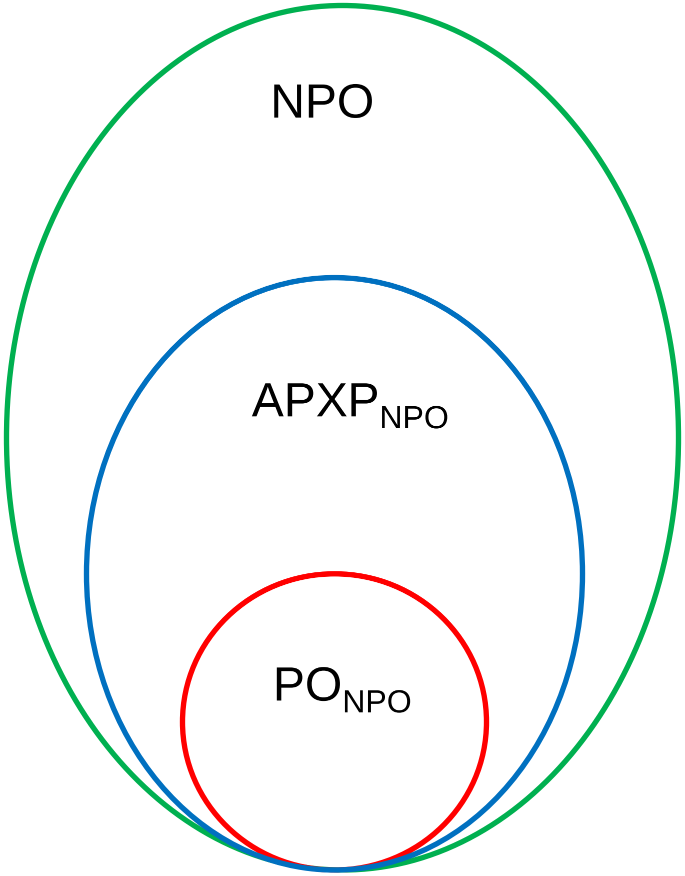
- In a polynomial-time setting, we take one typical class of optimization problems whose optimal solutions can be approximable.

- APXP_{NPO} (or APX) = a class of NP optimization problems that are polynomial-time γ-approximable for certain constant $\gamma > 0$.

- There are other notions of approximation algorithms.
  - polynomial-time approximation scheme (PTAS)
  - fully polynomial-time approximation scheme (FPTAS)

Relationships among Optimization Classes

- NPO
  - Contains combinatorial optimization problems defined in a form of NP problems
- APXP_{NPO} (or simply, APX)
  - Contains NPO problems whose optimal solutions can be relatively approximately found by deterministic TMs in poly time
- PO_{NPO} (or simply, PO)
  - Contains NPO problems whose optimal solutions can be found by deterministic TMs in poly time



Assuming $P \neq NP$

Reductions and Completeness

- To discuss the complexity of optimization problems, we need a notion of "completeness" for a given class.
- For complete problems, we further need a notion of "reduction."
- A reduction is a way to compare the computational difficulty of two optimization problems by transforming an optimization problem $P = (I_1, SOL_1, m_1, goal)$ to another optimization problem $Q = (I_2, SOL_2, m_2, goal)$ so that if Q is easy to solve then P is also easy to solve.
- A complete problem is one of the most difficult problems in a given class C.

Preserving Approximability of NPO Problems

- Let us discuss an appropriate reducibility notion for optimization problems.
- For NPO problems, every reduction must preserve the approximability of those problems.
- More precisely, let $P = (I_1,SOL_1,m_1,goal)$ and $Q = (I_2,SOL_2,m_2,goal)$.
- When P is "reducible" to Q, we require that, if Q is approximable, then P is also approximable.
- This means that "reductions" must preserve "approximability."
- (*) In the next slide, we explain "approximation-preserving reduction" (or "APP-reduction").

Approximation-Preserving (APP) Reductions

- P is APP-reducible to Q (denoted by $P \leq_{AP}^{P} Q$) $\Leftrightarrow$
  - $\exists f,g\ \exists c \geq 1$ s.t.
    1. $\forall x \in I_1\ \forall r \in \mathbb{Q}^{>1}\ [\ f(x,r) \in I_2\ ]$
    2. $\forall x \in I_1\ \forall r \in \mathbb{Q}^{>1}\ [\ SOL_1(x) \neq \varnothing \rightarrow SOL_2(f(x,r)) \neq \varnothing\ ]$
    3. $\forall x \in I_1\ \forall r \in \mathbb{Q}^{>1}\ \forall y \in SOL_2(f(x,r))\ [\ g(x,y,r) \in SOL_1(x)\ ]$
    4. $f, g \in$ auxFL for each fixed $r \in \mathbb{Q}^{>1}$
    5. $\forall x \in I_1\ \forall r \in \mathbb{Q}^{>1}\ \forall y \in SOL_2(f(x,r))\ [\ R_2(f(x,r),y) \leq r \rightarrow R_1(x,g(x,y,r)) \leq 1+c(r-1)\ ]$,

  where $\mathbb{Q}^{>1} = \{\ r \in \mathbb{Q} \mid r > 1\ \}$.

$$
\begin{tikzcd}
x \in I_1 \arrow[r, "f"] \arrow[d] & f(x,r) \in I_2 \arrow[d] \\
g(x,y,r) \in SOL_1(x) & y \in SOL_2(f(x,r)) \arrow[l, "g"]
\end{tikzcd}
$$

Completeness by APP-Reductions

- With the use of APP-reductions, we can define completeness.

- Let C be a subclass of NPO. (E.g., PO_{NPO}, $APXP_{NPO}$, etc.)
- Let P be any NPO problem.

- We say that P is C-complete if
  1. P is in C, and
  2. for any optimization problem B in C, B is APP-reducible to A, i.e., $\forall B \in C$ [$B \leq_{AP}^{p} A$].

Example: MinLP

- Minimum {0,1}-Linear Programming Problem (MinLP)
  - **Instance:** matrix $A \in \mathbb{Z}^{mn}$, vectors $b \in \mathbb{Z}^m$, $w \in \mathbb{N}^n$
  - **Solution:** vector $x \in \{0,1\}^n$ s.t. $Ax \geq b$
  - **Measure:** scalar product

$$w \bullet x = \sum_{i=1}^{n} w_i x_i$$

- (Claim) MinLP is NPO-complete.

$$A = \begin{bmatrix} 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$w = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 2 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

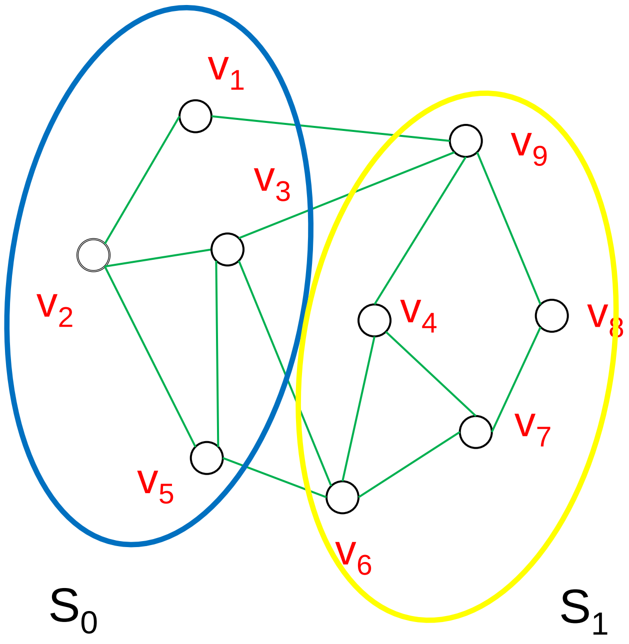
$$w \bullet x = 2x_1 + 3x_2 + x_4 + 2x_5$$

$$Ax \geq b \quad \Leftrightarrow \quad \begin{cases} x_1 - x_2 + x_3 \geq 0 \\ x_4 - x_5 \geq 0 \\ x_2 + x_3 + x_4 \geq 1 \end{cases}$$

Example: MaxCut

- Maximum Cut Problem (MaxCut)
  - Instance: an undirected graph G=(V,E)
  - Solution: a cut (i.e., a partition (S_0,S_1) of V)
  - Measure: cut capacity (i.e., the number of edges crossing between S_0 and S_1)

- (Claim) MaxCut is $APXP_{NPO}$-complete.



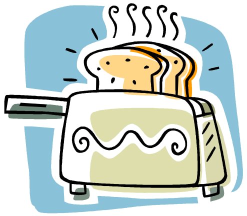
A cut

$S_0 = \{ v_1,v_2,v_3, v_5 \}$

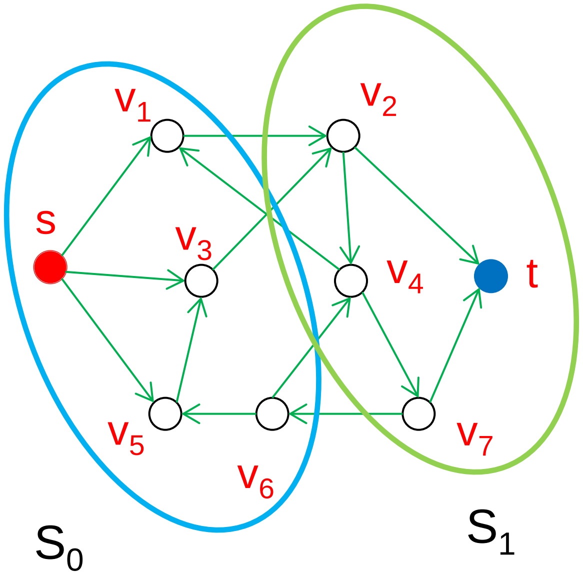
$S_1 = \{ v_4,v_6,v_7,v_8,v_9 \}$

cut capacity = 4

Example: Min st-Cut



- Minimal s-t Cut Problem (Min st-Cut)
  - ➢ Instance: directed graph G, source s, and sink t
  - ➢ Solution: st-cut (S_0,S_1) with $s \in S_0$ and $t \in S_1$
  - ➢ Measure: capacity of st-cut (total number of edges from S_0 to S_1)



- (Claim) Min st-Cut is PO_{NPO}-compete.

An st-cut

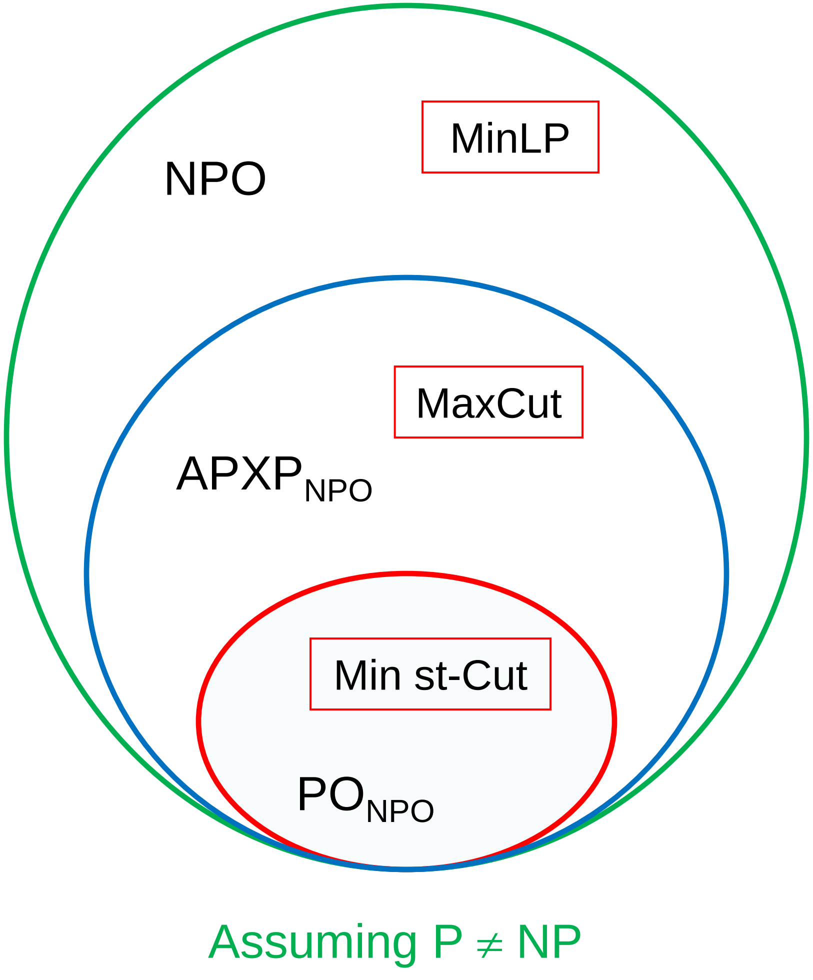
$S_0 = \{ s,v_1,v_3,v_5,v_6 \}$
$S_1 = \{ v_2,v_4,v_5,t \}$

$\boxed{\text{cut capacity} = 3}$

A Map of Complete Problems

- We then obtain complete problems for each optimization/approximation classes.

- Completeness is based on $\leq_{AP}^{P}$-reductions.



Inside PO_{NPO}

- Consider the following two NP optimization problems.

1. Maximum vertex weight problem (Max Vertex)
2. Maximum Boolean formula value problem (Max BFVP)

- These problems are both in PO_{NPO}, but their computational complexities seem to be quite different.

- In the next section, we will look into the inside structure of PO_{NPO}.

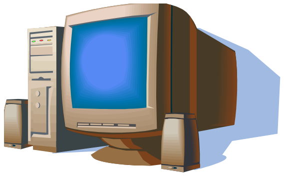
III. NL Optimization Problems

1. Auxiliary Turing Machines
2. auxL and auxFL
3. NL Optimization Classes
4. Polynomially-Bounded Problems
5. LO_{NLO}, LO_{NPO}, etc.
6. Log-Space Approximation Schemes
7. $APXL_{NLO}$, $APXL_{NPO}$, etc.
8. NC^1O_{NLO}, AC^0O_{NLO}, etc.

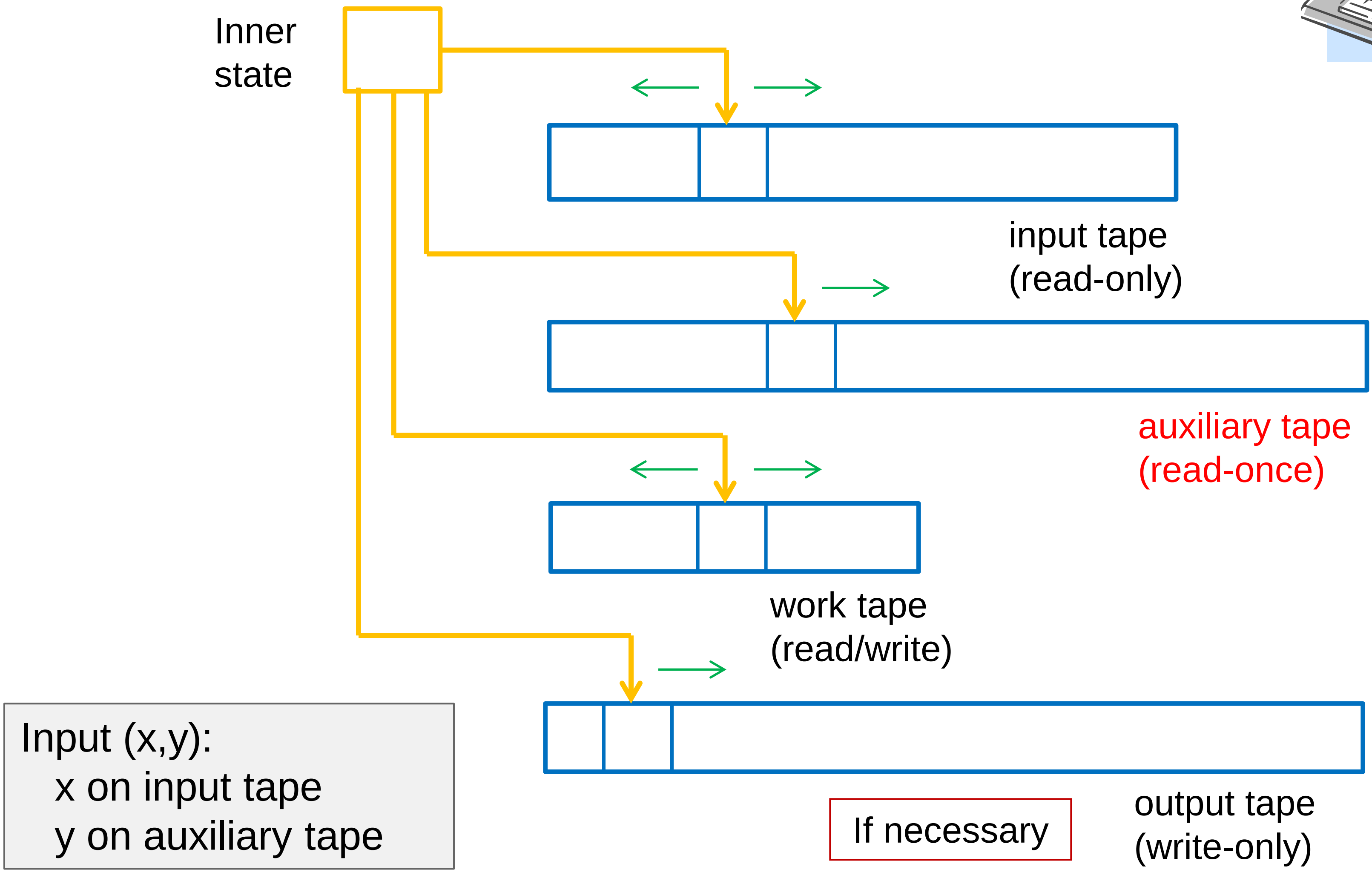
How to Refine Problems Inside PO_{NPO}

- To discuss optimization problems inside PO_{NPO}:
  - We need a refinement of the existing notions.
  - We look into log-space approximation and uniform-circuit (based) approximation schemes.

- First, we consider Turing machines equipped with extra read-once input tapes, called auxiliary tapes.
- See the next slide.



Auxiliary Turing Machines



Auxiliary TMs and Complexity Class NL

- Nondeterministic TMs are simulated by auxiliary TMs.
- NL: nondeterministic log-space
  - Input is given on input tape and a series of nondeterministic choices is given on auxiliary tape.
- L: deterministic log-space class
- Examples:
  - The s-t connectivity problem on directed graphs (DSTCON) is NL-complete.
  - The s-t connectivity problem on undirected graphs (USTCON) is L-complete.

auxL and auxFL

- We need to treat instances of the form (x,y).
- Auxiliary L (auxL)
  - auxL = problems A solvable by auxiliary TMs M using log space with the following condition:

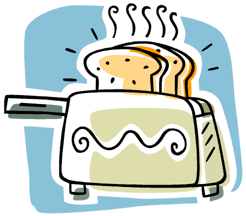
  (*) $\exists$p: poly s.t., for any input (x,y) to A,

    1) $(x,y) \in A \Rightarrow |y| \leq p(|x|)$
    2) when $|y| \leq p(|x|)$, M accepts $(x,y) \leftrightarrow (x,y) \in A$.
- Auxiliary FL (auxFL)
  - auxFL = functions computable by auxiliary TMs using log space with write-only output tapes with polynomial output size

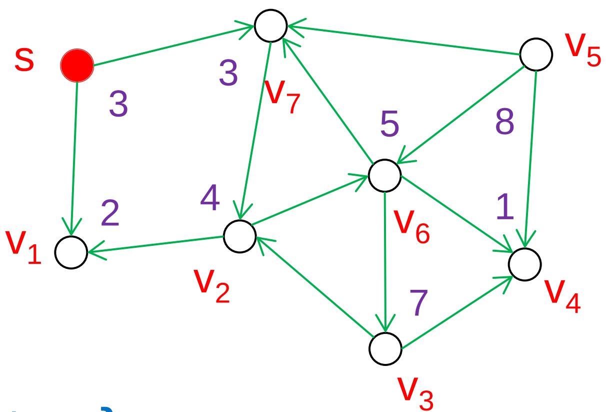
NL Optimization (or NPO) Problems

- NLO problem: P=(I,SOL,m,goal) [Tantau,2007]
  - I = finite set of admissible instances
  - SOL = function from I s.t. SOL(x) is a set of feasible solutions of x
    - $\exists$q:poly $\forall$x$\in$I $\forall$y$\in$SOL(x) [|y| $\leq$ q(|x|)]
    - I$\circ$SOL = {(x,y) | x$\in$I, y$\in$SOL(x)} is in auxL
  - goal = either max or min
  - m = measure (or objective) function from I$\circ$SOL to $\mathbb{N}$
    - m is in auxFL
    - m*(x) = optimal value among m(x,y) with y$\in$SOL(x)
- MinNL = minimization problems in NLO
- MaxNL = maximization problems in NLO
- NLO = MaxNL $\cup$ MinNL

Example: Max Vertex



- Maximum vertex weight problem (Max Vertex)
  - ➢ Instance: directed graph G, source s, (vertex) weight function w
  - ➢ Solution: path from s to a certain vertex t
  - ➢ Measure: weight of t



- We define:
  - I = {(G,s,w): graph G, source s, weight w }
  - SOL(G,s,w) = { path p: from s to some y }
  - m((G,s,w),p) = w(y), where y is an endpoint of p
- Thus, Max Vertex is in NLO.

Polynomially-Bounded Problems

- Note that, if $m \in$ auxFL, $m(x,y) \leq 2^{p(|x|)}$ for an absolute polynomial p.
- It is useful to focus our attention to polynomially-bounded problems.

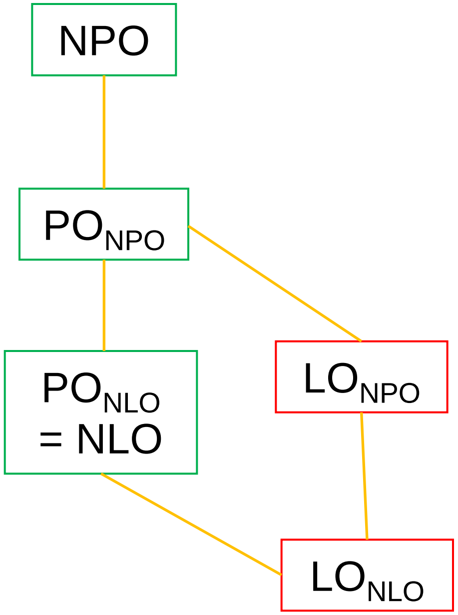
- Problem P is polynomially-bounded

  $\Leftrightarrow \quad \exists p\text{:poly } \forall (x,y) \in I^\circ SOL \; [\; m(x,y) \leq p(|x|,|y|) \;]$.

- PBO = set of polynomially-bounded optimization problems

LO_{NLO}, LO_{NPO}, etc.

- LO problems inside C: P=(I,SOL,m,goal)
  - P is an optimization problem in C.
  - P is L-solvable; that is, a certain DTM M finds an optimal solution y of x using log space for every $x \in I$.
- LO_C = set of all LO problems inside C
- Examples
  - LO_{NPO} = set of all LO problems in NPO
  - LO_{NLO} = set of all LO problems in NLO



- (Claim) PO_{NLO} = NLO. [Yamakami (2013)]

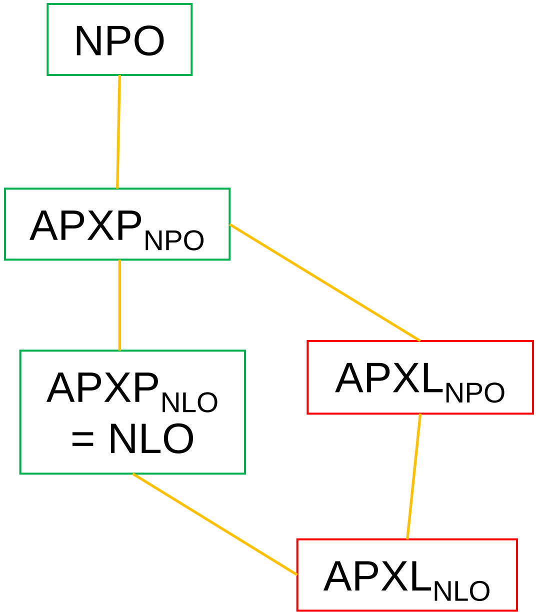
Log-Space Approximation Schemes

- We introduce log-space approximable problems.

- Let P = (I, sol, m, goal) be any optimization problem.
- Recall that an algorithm M is a γ-approximate algorithm $\Leftrightarrow \ \forall x \in I\ [\ R(x,M(x)) \leq \gamma\ ]$.
- P is log-space γ-approximable $\Leftrightarrow \ \exists$M log-space DTM s.t. $\forall x \in I\ [\ R(x,M(x)) \leq \gamma\ ]$.

$APXL_{NLO}$, $APXL_{NPO}$, etc.

- APXL problems P=(I,SOL,m,goal) in C:
  - P is an optimization problem in C.
  - P is L-approximable; that is, a certain DTM M finds an approximate optimal solution y of x using log space for every $x \in I$.
- $APXL_C$ = set of all APXL problems inside C
  - $APXL_{NPO}$ = set of all APXL problems in NPO
  - $APXL_{NLO}$ = set of all LO problems in NLO
- (Claim) $APXP_{NLO}$ = NLO. [Yamakami (2013)]



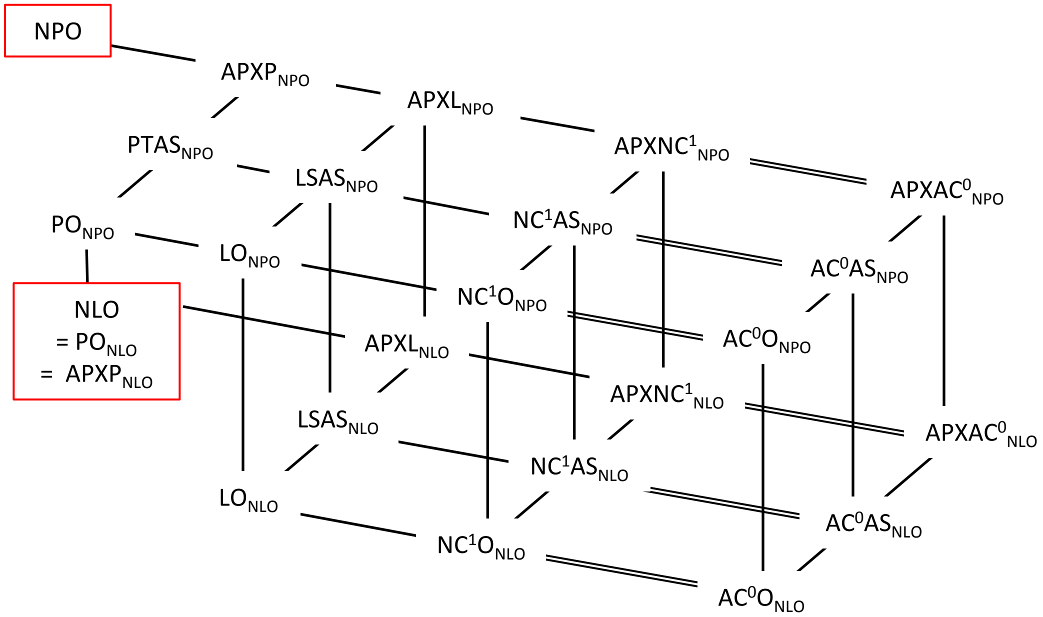
NC^1O_{NLO}, AC^0O_{NLO}, etc.

- We introduce circuit-based optimization problems.
- Recall AC^0 and NC^1.
- AC^0 = Uniform circuits of constant-depth and unbounded-fain
- NC^1 = Uniform circuits of O(log n)-depth and bounded-fain

- We define the following two classes.
  - AC^0O_{NLO} = class of all NLO problems that are AC^0-solvable
  - NC^1O_{NLO} = class of all NLO problems that are NC^1-solvable

Relations among Refined Classes

- We summarize the inclusion relationships among refined classes.



IV. Complete NL Optimization Problems

1. Reductions and Completeness
2. Approximation-Preserving Reductions
3. Exact Reductions
4. Complete Problems

Approximation-Preserving Reductions

- Let P = (I_1,SOL_1,m_1,goal) and Q = (I_2,SOL_2,m_2,goal).
- P is APL-reducible to Q ($P \leq_{AP}^{P} Q$) $\Leftrightarrow$
  - $\exists f,g \in FL \ \ \exists c \geq 1$ s.t.
    1. $\forall x \in I_1 \ \forall r \in \mathbb{Q}^{>1}$ [$f(x,r) \in I_2$]
    2. $\forall x \in I_1 \ \forall r \in \mathbb{Q}^{>1}$ [$SOL_1(x) \neq \emptyset \rightarrow SOL_2(f(x,r)) \neq \emptyset$]
    3. $\forall x \in I_1 \ \forall r \in \mathbb{Q}^{>1} \ \forall y \in SOL_2(f(x,r))$ [$g(x,y,r) \in SOL_1(x)$]
    4. f, g $\in$ auxFL for each fixed $r \in \mathbb{Q}^{>1}$
    5. $\forall x \in I_1 \ \forall r \in \mathbb{Q}^{>1} \ \forall y \in SOL_2(f(x,r))$ [$R_2(f(x,r),y) \leq r \rightarrow R_1(x,g(x,y,r)) \leq 1+c(r-1)$],

  where $\mathbb{Q}^{>1} = \{ r \in \mathbb{Q} \mid r>1 \}$.

$$
\begin{tikzcd}
x \in I_1 \arrow[r, "f"] \arrow[d] & f(x,r) \in I_2 \arrow[d] \\
g(x,y,r) \in SOL_1(x) & y \in SOL_2(f(x,r)) \arrow[l, "g"']
\end{tikzcd}
$$

More Reductions

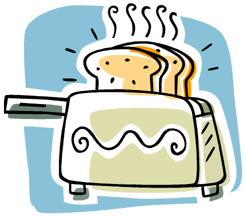
- We define three additional reductions:
  - APL reductions ($P \leq_{AP}^{L} Q$)
  - $APNC^1$ reductions ($P \leq_{AP}^{NC1} Q$)
  - $APAC^0$ reductions ($P \leq_{AP}^{AC0} Q$)
- Moreover, if we replace $\mathbb{Q}^{>1}$ in $P \leq_{sAP}^{L} Q$ with $\mathbb{Q}^{\geq 1}$, we obtain
  - Strong APL reductions ($P \leq_{sAP}^{L} Q$)

Exact Reductions

- We introduce another type of reduction.
- Let P = (I_1,SOL_1,m_1,goal) and Q = (I_2,SOL_2,m_2,goal).

- EXL reductions ($P \leq_{EX}^{L} Q$) $\Leftrightarrow$
  - $\exists f,g\ \exists c \geq 1$ s.t.
    1. $\forall x \in I_1\ [\ f(x) \in I_2\ ]$
    2. $\forall x \in I_1\ [\ SOL_1(x) \neq \emptyset \rightarrow SOL_2(f(x)) \neq \emptyset\ ]$
    3. $\forall x \in I_1\ \forall y \in SOL_2(f(x))\ [\ g(x,y) \in SOL_1(x)\ ]$
    4. f, g $\in$ auxFL
    5. $\forall x \in I_1\ \forall y \in SOL_2(f(x))\ [\ R_2(f(x),y) = 1 \rightarrow R_1(x,g(x,y)) = 1\ ]$.

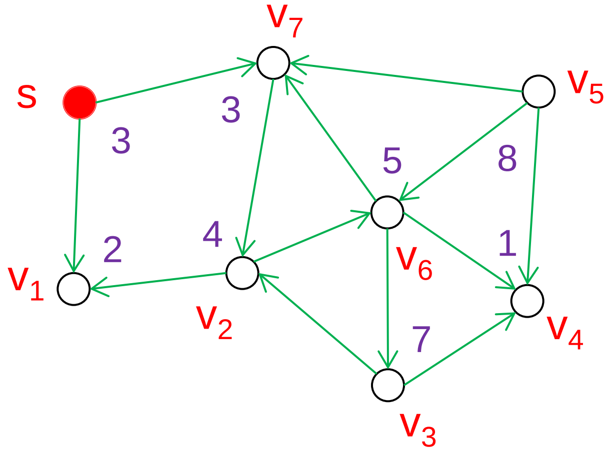
Example: Max Vertex



- Maximum vertex weight problem (Max Vertex)
  - Instance: directed graph G, source s, (vertex) weight function w
  - Solution: path from s to a certain vertex t
  - Measure: weight of t (= w(t))

Optimal solution

$s \rightarrow v_7 \rightarrow v_2 \rightarrow v_6 \rightarrow v_3$



- Max Vertex is in PO_{NPO}.
- However, it does not seem to be complete for PO_{NPO}.
- (Question) What is the exact complexity of this problem?

Example: Max BFVP

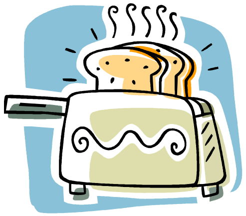
- Maximum Boolean formula value problem (MaxBFVP)
  - Instance: set F of Boolean formulas and a Boolean assignment σ
  - Solution: subset C of satisfied formulas by σ
  - Measure: number of formulas in C

$$F = \{x_1 \vee x_3, (x_1 \vee \neg x_2) \wedge x_3, x_1 \vee x_2 \vee x_3\}$$

$$\sigma(x_1) = 1, \sigma(x_2) = 0, \sigma(x_3) = 1$$

- MaxBFVP is in PO_{NPO}.
- However, it does not seem to be complete for PO_{NPO}.
- (Question) What is the exact complexity of this problem?

Complete Problems I



- Finally, we exhibit a short list of complete problems.

- Max Vertex
  - $\leq_{AP}^{L}$-complete for $APXL_{MaxNL}$ [Tantau (2007)]
- Min Path-Weight
  - $\leq_{sAP}^{NC1}$-complete for MinNL
- Min Forest-Path-Weight
  - $\leq_{sAP}^{NC1}$-complete for $APXL_{MinNL}$

Complete Problems II

- Recall that a problem P is polynomially-bounded

  $\Leftrightarrow \quad \exists p{:}\mathrm{poly}\ \forall (x,y) \in I \circ SOL\ [\ m(x,y) \leq p(|x|,|y|)\ ]$.

- PBO = set of polynomially-bounded optimization problems

- Max B-Vertex
  - $\leq_{EX}^{NC1}$-complete for $LO_{NLO} \cap PBO$
- Max BFVP
  - $\leq_{EX}^{NC1}$-complete for $NC^1O_{NLO} \cap PBO$

V. Relations among Log-Space Classes

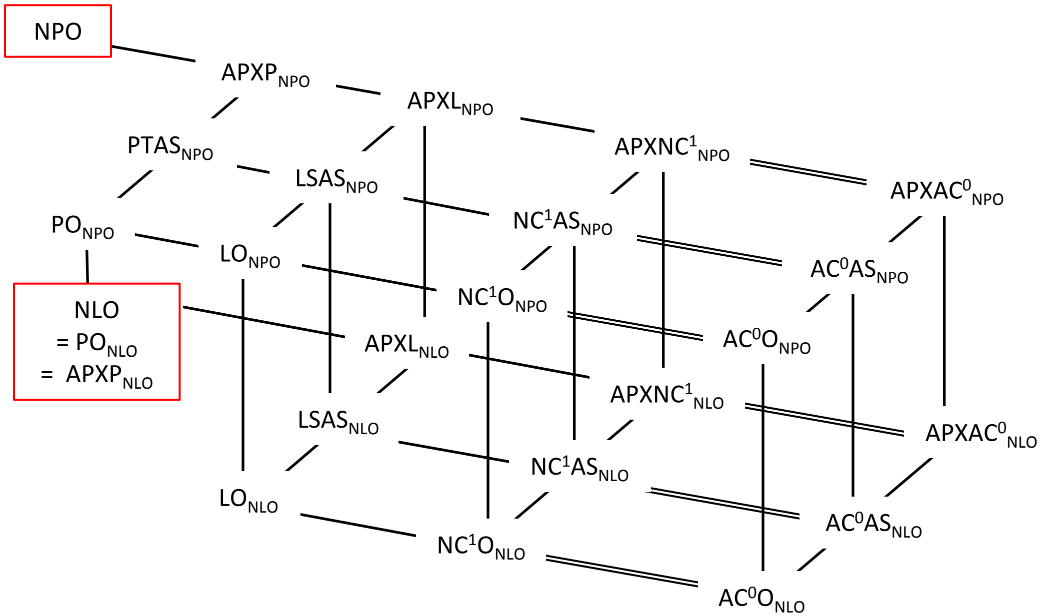
1. Relations among Classes
2. Inclusion Relationships

Relations among Classes

- Yamakami (2011) showed the following.
- Implications
  - $L = P \Leftrightarrow LO_{NLO} \cap PBO = PO_{NLO} \cap PBO$
  - $NC^1 = L \Leftrightarrow NC^1O_{NLO} \cap PBO = LO_{NLO} \cap PBO$
  - $L \neq P \Rightarrow PO_{NPO} \not\subseteq APXL_{NLO}$
  - $NC^1 \neq NL \Rightarrow NC^1O_{NLO} \neq APXNC^1_{NLO}$
- Separations
  - $NC^1O_{NLO} \not\subseteq APXAC^0_{NLO}$
  - $AC^0O_{NLO} \neq APXAC^0_{NLO}$

Inclusion Relationships (again)

- We review our optimization classes again.



- Open Problem: We need to find more interesting problems inside those classes.

VI. Optimization CSPs

1. Combinatorial Optimization Problems
2. Maximization CSPs
3. Visualizing MAX-CSP(F)
4. A Known Classification Theorem
5. A Use of Products of Objective Functions
6. Definition of MAX-PROD-CSPs

Counting Constraint Satisfaction Problems (revisited)

- We recall the notion of counting CSPs or #CSPs from Week 8.
- Let F be any set of constraints.
- Counting CSP: #CSP(F)
  - Instance:
    - a set of Boolean variables
    - a set of constraints in F
  - Question:
    - How many (variable) assignments satisfy all the given constraints?
- NOTE: all #CSPs (counting CSPs) are #P problems.

Weighted Constraints

- Creignou (1995) first gave a formal treatment to maximization problem. He used unweighted Boolean constraints.

- Here, we consider nonnegative real weighted constraints, which are functions from $\{0,1\}^k$ to $R^{\geq 0} = \{ r \in R \mid r \geq 0 \}$.

- Creignou (1995) and Khanna, Sudan, Trevisan, and Williamson (2001) studied maximization CSPs.

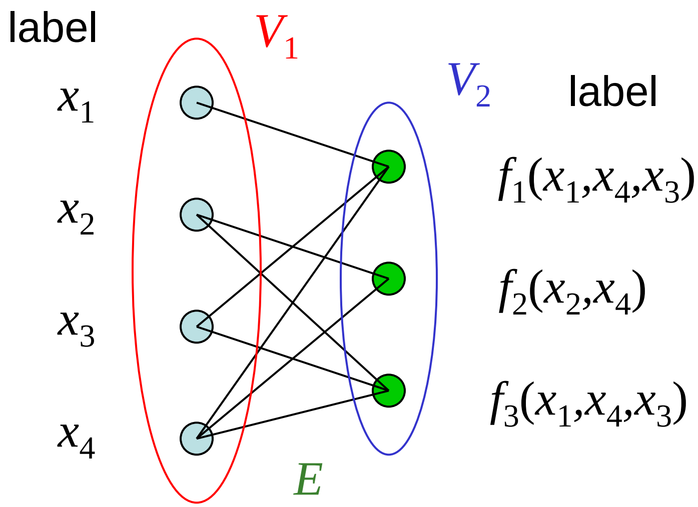
Maximization CSPs (or MAX-CSPs)

- Let F be any set of constraints.
- Maximization CSP: MAX-CSP(F)
  - Instance:
    - A finite set of elements of the form $\langle h,(x_{i1},x_{i2},\ldots,x_{ik})\rangle$ on Boolean variables, where $h \in F$, $\{i_1,i_2,\ldots,i_k\} \subseteq [n]$.
  - Solution:
    - A truth assignment σ to $x_1,x_2,\ldots,x_n$.
  - Measure:
    - The sum $\sum h(\sigma(x_{i1}),\sigma(x_{i2}), \ldots, \sigma(x_{ik}))$, where the sum is taken over all $\langle h,(x_{i1},x_{i2},\ldots,x_{ik})\rangle \in H$.
- MAX-CSP(XOR) coincides with MAX-CUT, which is MAX-SNP-complete [Papadimitriou-Yannakakis (1991)]

This measure is referred to as an additive measure.

Visualizing MAX-CSP(F)

- An input Ω = (G,F',π), where
  1. a bipartite graph $G = (V_1|V_2,E)$,
  2. F' $\subseteq$ F, a finite subset,
  3. π: $V_2 \rightarrow$ F' (a labeling function).



Boolean variables: $\{x_1,x_2,x_3,x_4\}$

(Boolean) constraints:

$\{ f_1(x_1,x_4,x_3), f_2(x_2,x_4), f_3(x_1,x_4,x_3) \}$

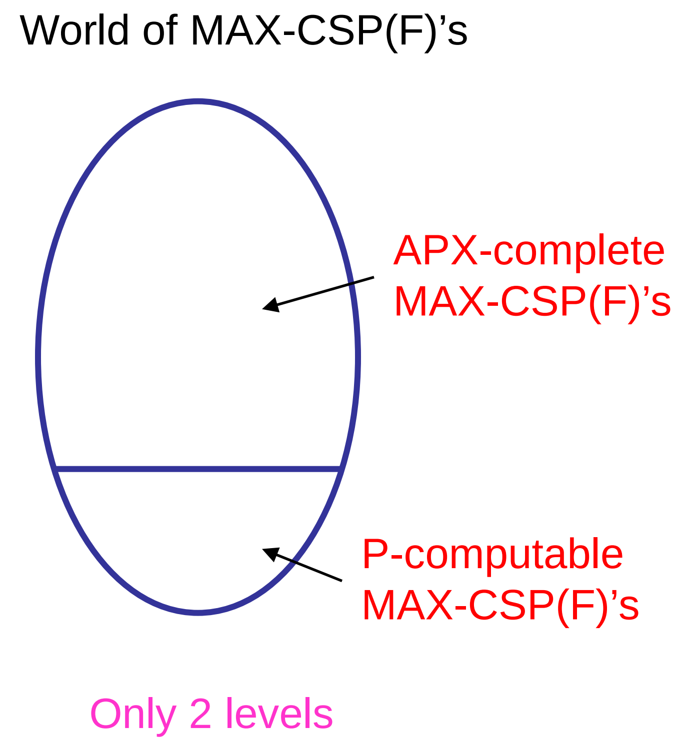
We want to maximize the sum of all objective values.

$$\max_{x_1,x_2,x_3,x_4 \in \{0,1\}} \{ f_1(x_1,x_4,x_3) + f_2(x_2,x_4) + f_3(x_3,x_2,x_4) \}$$

- MAX-CSP(F)
  - Instance: an input Ω
  - Solution: find an optimal solution
  - Measure: sum of all constraints

A Known Classification Theorem

- Creignou (1995) and Khanna, Sudan, Trevisan, and Williamson (2001) proved the following classification theorem on MAX-CSPs.
- Let F be any set of constraints.
- Dichotomy Theorem
  - If F is 0-valid, 1-valid, or 2-monotone, then MAX-CSP(F) is in PO.
  - Otherwise, MAX-CSP(F) is APX-complete.



A Use of Products of Objective Functions

- There are a number of cases where products of objective values have been used.
- Linear multiplicative programming
  - This minimizes the product of two positive linear cost functions, subject to linear constraints.
- Geometric programming
  - A certain type of product objective function can be reduced to additive one if function values are all positive.
- MAX-PROD-KNAPSACK
  - Marchetti-Spaccamela and Romano (1985) proved that a maximization problem whose maximization is measured by the product of objective values.
- We call such a measure a multiplicative measure.

Example: MAX-PROD-KNAPSACK

- MAX-PROD-KNAPSACK
  - instance: a finite set X of items, value $p_i \in N^+$ and size $a_i \in N^+$ for each item $x_i \in X$, and a number $b \in N^+$
  - solution: a set $Y \subseteq X$ such that $\Sigma_{xi \in Y} a_i \leq b$
  - measure: multiplicative value $\Pi_{xi \in Y} p_i$

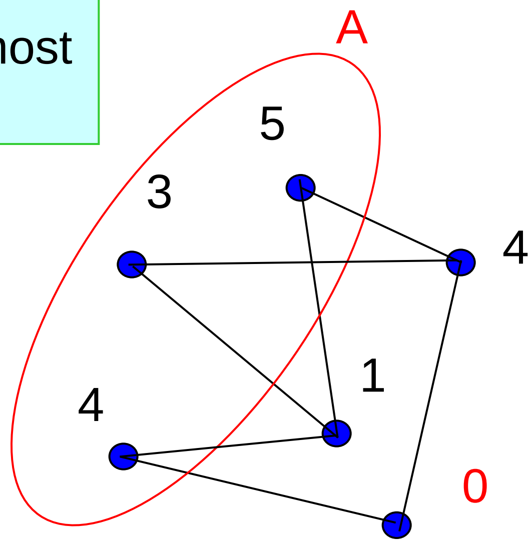
- (Claim) MAX-PROD-KNAPSACK has an FPTAS. [Marchetti-Spaccamela and Romano (1985)]

- In comparison:
- (Fact) MAX-KNAPSACK (with additive measure) has an FPTAS. [Ibarra-Kim (1975)]

Example: MAX-PROD-IS

- We see an example with a multiplicative measure.
- MAX-PROD-IS (maximum product independe
  - Instanc
    - An undirected graph G = (V,E)
    - A series $\{ w_x \}_{x \in V}$ of vertex weights with $w_x \in R^{\geq 0}$
  - Solution:
    - An independent set A on G
  - Measure:
    - The maximum product weight $\prod_{x \in A} w_x$

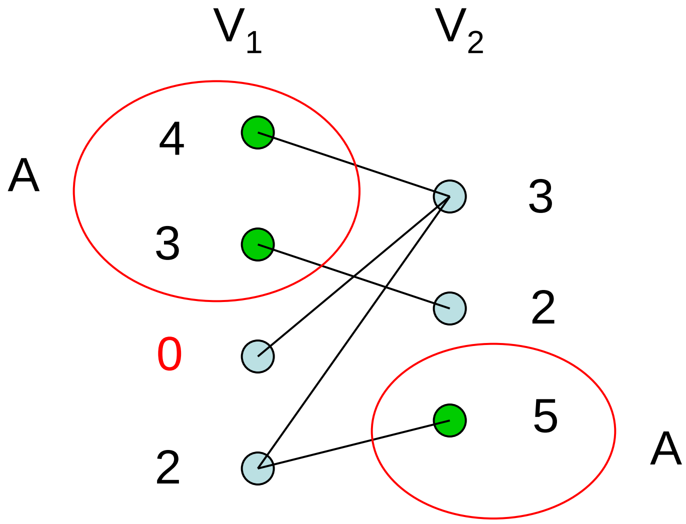
An independent set A is a subset of V s.t. each edge in E is incident on at most one vertex in A.



product weight = 60

Example: MAX-PROD-BIS

- Another example is a restricted form of MAX-PROD-IS.
- MAX-PROD-BIS (bipartite independent set)
  - In MAX-PROD-IS, all input graphs are limited to bipartite graphs.



$G = (V_1|V_2, E)$

A: an independent set

product weight = 60

Example: MAX-PROD-FLOW

- MAX-PROD-FLOW (maximum product flow)
  - Instance:
    - A directed graph G = (V,E), a series $\{\rho_e\}_{e \in E}$ of flow rates with $\rho_e \geq 1$, and a series $\{w_z\}_{z \in V}$ of influx rates with $w_x \geq 0$
  - Solution:
    - A Boolean assignment σ of V
  - Measure:
    - The product

$$\left(\prod_{(x,y)\in E,\sigma(x)\geq\sigma(y)} \rho_{(x,y)}\right)\left(\prod_{z\in V,\sigma(z)=1} w_z\right)$$

Definition of MAX-PROD-CSPs

- We want to conduct a general study about optimization CSPs whose maximization is taken by multiplicative measures.
- Let F be any set of constraints.

- Maximization Product CSP: MAX-PROD-CSP(F)
  - Instance:
    - A finite set of elements of the form)⟩ on Boolean variables $x_{i1},x_{i2},\ldots,x_{ik}$, w $\{i_1,i_2,\ldots,i_k\} \subseteq [n]$.

This is an multiplicative measure.

  - Solution:
    - A truth assignment σ to $x_1,x_2,\ldots,x_n$.
  - Measure:
    - The product $\prod h(\sigma(x_{i1}),\sigma(x_{i2}), \ldots, \sigma(x_{ik}))$, where the product is taken over all $\langle h,(x_{i1},x_{i2},\ldots,x_{ik})\rangle \in H$.

Unary Constraints are Free of Charge

- For our results, we allow any unary constraint to use for free.
- Simple examples of unary constraints are Δ_0, and Δ_1.
- Let U be the set of all unary constraints.
- Such a use of free unary constraints has been made elsewhere.
  - Feder (2001) for Boolean CSPs
  - Dalmau-Ford (2003) for Boolean CSPs
  - Cai-Huang-Lu (2010) for Holant problems
  - Cai-Lu-Xia (2009) for Holant problems
  - Dyer-Goldberg-Jalsenius-Richerby (2010) for bounded-degree #CSPs
  - Yamakami (2010) for bounded-degree #CSPs
- Notational convention:
  - MAX-PROD-CSP*(F) $=_{def}$ MAX-PROD-CSP(F,U)

$\Delta_0(x)$ = False(x)
$\Delta_1(x)$ = True(x)

Reminder

VII. Approximation Schemes

1. Randomized Approximation Schemes
2. Approximation-Preserving Turing Reductions
3. exp-$APXP_{NPO}$
4. Important Sets of Constraints
5. Classification Theorem

Randomized Approximation Schemes

- We explain what type of approximation
- Let P = (I, sol, m, goal) be any optimiz
- A randomized approximation scheme (or RAS) for F is a probabilistic algorithm that
  - takes $(x,\varepsilon) \in I \times R^{\geq 0}$ as an input, and
  - outputs a solution $y \in sol(x)$ such that
  - m*(x) is approximated by m(x,y) with relative error of 2^{ε} with high probability.
- A fully polynomial-time randomized approximation scheme (or FPRAS) is a RAS that runs in time polynomial in $(|x|,1/\varepsilon)$.

NOTE: in this model, even if α and β are approximated, $\alpha+\beta$ may not be approximated properly for real numbers α, β.

$$2^{-\varepsilon} \leq \left| \frac{m(x,y)}{m^*(x)} \right| \leq 2^{\varepsilon}$$

Approximation-Preserving Turing Reductions

- Dyer, Goldberg, Greenhill, and Jerrum (2003) introduced a notion of approximation-preserving (Turing) reduction for counting CSPs.
- Yamakami (2011) introduced a similar reduction for optimization CSPs
- Let P=(I,sol,m,goal) ar optimization problems.

Notational conventions:

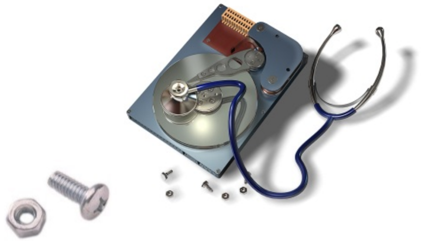
$P \leq_{APT} Q \Leftrightarrow$ P is APT-reducible to Q.

$P \equiv_{APT} Q \Leftrightarrow P \leq_{APT} Q$ and $Q \leq_{APT} P$.

- P is APT-reducible to Q by a reduction M $\Leftrightarrow$
  1. M is an oracle PTM working on input (x,ε) with an oracle,
  2. M is a RAS for F and the oracle is also a RAS for G,
  3. every oracle call made by M is of the form (w,δ) with $1/\delta \leq poly(|x|,1/\varepsilon)$,
  4. the running time of M is bounded by a polynomial in $(|x|,1/\varepsilon)$.

exp-APXP$_{NPO}$

- Recall that APXP$_{NPO}$ (or APX) consists of all NPO problems, each of which is polynomial-time γ-approximable for a certain constant γ>0.
- A function $f : N \to N$ is called exponentially bounded if there is a positive polynomial p such that $1 \leq f(n) \leq 2^{p(n)}$ for every $n \in N$.
- exp-APXP$_{NPO}$ consists of all NPO problems P such that there are an exponentially-bounded function r and a polynomial-time r-approximate algorithm for P.



Lemmas

- We can prove the following lemmas.

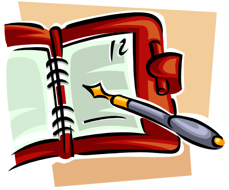
- Lemma [Yamakami (2011)]

  For any set F of constraints,

  - MAX-PROD-CSP(F) $\leq_{\mathrm{APT}}$ exp-APXP$_{\mathrm{NPO}}$

- Lemma [Yamakami (2011)]

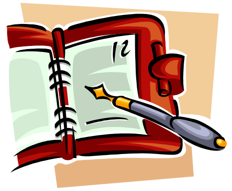
  - MAX-PROD-IS $\leq_{\mathrm{APT}}$ MAX-PROD-CSP*(OR)
  - MAX-PROD-BIS $\leq_{\mathrm{APT}}$ MAX-PROD-CSP*(Implies)



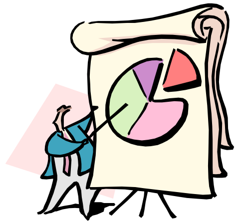
Important Sets of Constraints

- We introduce several important sets of constraints.

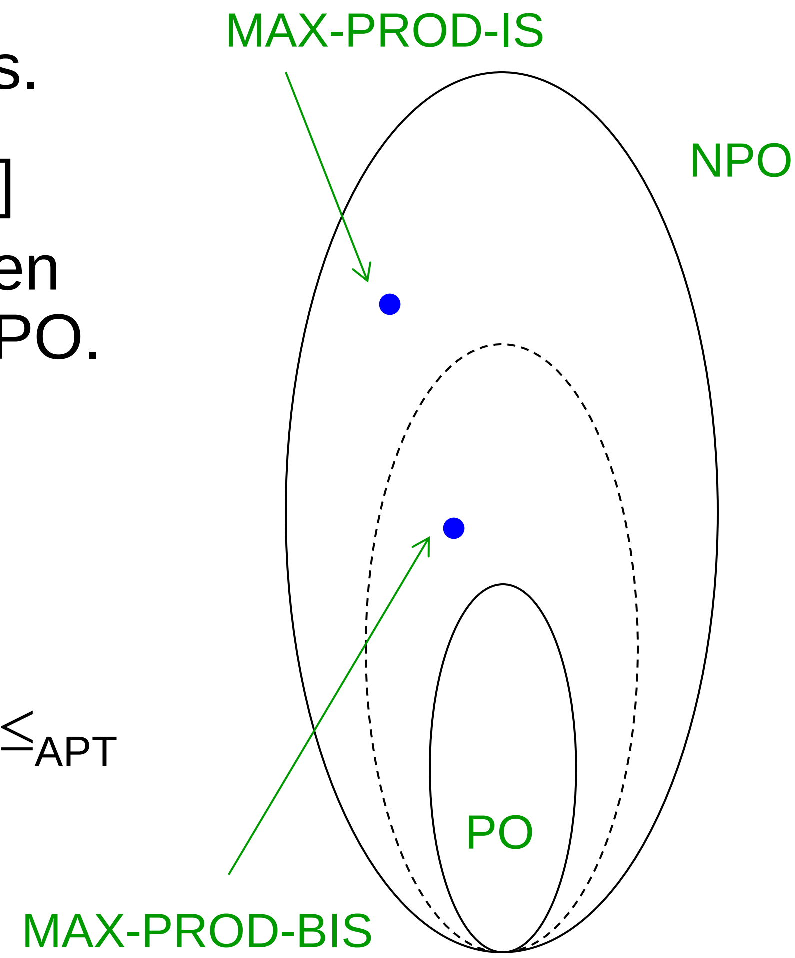
- DG = the set of degenerate constraints (that is, products of unary constraints)
- ED = the set of constraints which are products of unary constraints, EQ_2, and XOR
- AF = the set of affine-like constraints
- IM_{opt} = the set of constraints which are products of unary constraints, $(1,1,\lambda,1)$ with $0 \leq \lambda < 1$.



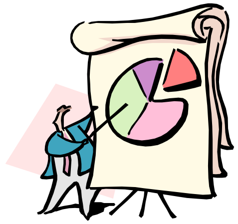
Classification Theorem I



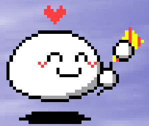
- Let F be any set of constraints.
- Theorem [Yamakami (2011)]
  1. If either $F \subseteq AF$ or $F \subseteq ED$, then MAX-PROD-CSP*(F) is in PO.
  2. Otherwise, if $F \subseteq IM_{opt}$, then MAX-PROD-BIS $\leq_{APT}$ MAX-PROD-CSP*(F) $\leq_{APT}$ MAX-PROD-FLOw.
  3. Otherwise, MAX-PROD-IS $\leq_{APT}$ MAX-PROD-CSP*(F).



Classification Theorem II



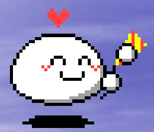
- A key is the following proposition about single constraints f.
- Proposition [Yamakami (2011)]
  - Assume that $f \notin \mathrm{AF} \cup \mathrm{ED}$. Let F be any signature set.
  1. If $f \in \mathrm{IM}_{opt}$, then MAX-PROD-CSP*(Implies,F) $\leq_{APT}$ MAX-PROD-CSP*(f, F).
  2. If $f \notin \mathrm{IM}_{opt}$, then there exists a constraint g$\in$\{ OR, NAND \} such that MAX-PROD-CSP*(g,F) $\leq_{APT}$ MAX-PROD-CSP*(f, F).



Thank you for listening

Q & A

I’m happy to take your question!



END