Probabilistically Checkable Proofs

Jonathan Kelner
5/5/2011

Dictionary Between the Two Views
- **Proof View:**
  - PCP verifier (V)
  - PCP proof (π)
  - Length of proof
  - # queries (q)
  - # random bits (r)
  - Soundness prob. (usually 1/2)

- **Hardness View:**
  - CSP instance (ϕ)
  - Assignment to vars
  - # variables (n)
  - Arity of constraints (q)
  - log(#constraints)
  - Max val(ϕ) for NO instance

- Given PCP, make CSP by making constraint for each possible set of bits queried
  - Constraint will be satisfied by vals verifier would accept
- Given CSP, make PCP by giving proof equal to variable assignments
  - Verifier will pick random constraint and check if corresponding variables from proof satisfy it

Last Time
- L has (r(n),q(n))-PCP verifier V if:
  - V uses ≤ r(n) random bits
  - V makes ≤ q(n) queries to proof
  - Proof is of length ≤ q(n)2ⁿʳ
    - Completeness: if x ∈ L, exists proof s.t. accept with prob. 1
    - Soundness: if x ∉ L, V does not accept any proof with prob > 1/2
- PCP(r(n),q(n))=languages with (O(r(n)),O(q(n))-verifiers
- qCSP: arbitrary collection of constraints ϕ s.t. each depends on ≤ q variables (call q the “arity”)
  - val(ϕ)=fraction of constraints that can be simultaneously satisfied
- ρ-Gap-qCSP problem: distinguish val=1 from val<ρ
- PCP Theorem has a few equivalent statements:
  - NP = PCP(log n, 1)
  - ∃ q∈ N, ρ∈ [0,1) s.t. ρ-Gap-qCSP is NP-hard
  - ∃ ρ<1 s.t., for every L ∈ NP, exists poly f mapping strings to 3CNF formulas s.t.:
    - x ∈ L =⇒ val(f(x))=1
    - x ∉ L =⇒ val(f(x))< ρ

Warmup: NP ⊆ PCP(poly(n),1)
- Will sketch this result using proof viewpoint
  - Will be used in proof of full theorem too
- Need exponential-sized proof, O(1) queries
- Enough give PCP for one NP-complete language
  - Will use L={satisfiable systems of quadratic eqs over GF(2)}
    - E.g. lists of equations like u₁u₂ + u₁u₄ + u₁u₅ = 1
      - Can assume no linear terms, since uᵢ² = uᵢ
  - Let U=n²-sized vector (uᵢ uᵢ)ᵢ,j
  - Can find A, b s.t. satisfying your system of eqs means finding U s.t.:
    - AU=b
    - U=(uᵢ uᵢ)ᵢ,j for some vector u
  - (If m=#constraints, A is m x n², b is n x 1)
If valid proof, will pass all tests.

Given proof, verifier will:

- Proof will consist of f and g, s.t.: f = Walsh-Hadamard encoding of u
g = Walsh-Hadamard encoding of u ⊕ u
- Can think of them as fn s., f : {0,1}^n → {0,1}, g : {0,1}^n → {0,1}
- Valid codewords in W-H iff linear
- Know can locally decode
- Can also locally verify: distinguish valid codewords from things at least 0.001 away from any codeword
- Total proof size is 2^n + 2^n
- Given proof, verifier will:
  - Verify f, g are (close to) linear
  - Use local verifier. Then locally decode, so can assume actually linear.
  - Verify that g r u ⊕ u, where u is string encoded by f
  - Choose random r, r' ∈ {0,1}^n, check if f(r) ⊕ f(r') ⊕ g(r ⊕ r')
  - Check that g encodes satisfying assignment
  - Need to satisfy all equations, but can’t check them all, and not okay to sample
  - Instead, add up a random subset of equations, take mod 2
- If valid proof, will pass all tests. If unsatisfiable, will fail with constant prob.

How We’ll Prove Stronger Theorem

- Take CSP view, show how to amplify gaps
- Main Theorem: ∃ constants q_o ≥ 3, ε_o > 0 s.t., for any ε << ε_o, can
  - Start with q_o CSP φ over alphabet {0,1}
  - Get new q_o CSP ψ over alphabet {0,1} with val(ψ) ≤ 1-2ε
  - New formula ψ only bigger by factor of C=C(q_o)
    - If val(φ)=1 then val(ψ)=1
    - If val(φ) ≤ 1-ε then val(ψ) ≤ 1-2ε
  - Implies PCP Theorem (CSP version):
    - Start with CSP φ with m constraints
    - Apply theorem log(m) times, get new CSP ψ
      - If φ satisfiable, so is ψ
      - If φ unsatisfiable, val(φ) ≤ 1-1/m
      - So val(ψ) ≤ 1- min(2ε_o, 1-2log m/m)=1-2ε_o
      - Size of ψ = Clog m = poly(m)

What the Steps Will Do

- Preprocessing:
  - Given constraint graph G, make G’=prep(G)
  - For some constants 0<λ<δ, δ > 0, will have:
    - G’ is d-regular, has self loops at all vertices, λ(G’)=δ< λd
    - G’ has same alphabet as G, size(G’)=O(size(G))
    - δ_1, UNSAT(G) ≤ UNSAT(G) ≤ UNSAT(G’)
  - Amplification:
    - For constant t, get new constraint graph G” s.t.:  
      - Same vertices
      - # edges multiplied by δ
      - Alphabet size |Σ|^d^2
    - UNSAT(G”) ≥ β_t · t^{1/2} · (UNSAT(G’))^{1/2}
      - β_t = const. depending only on λ, d, |Σ|
  - Alphabet Reduction:
    - Size increases by a constant factor
    - Alphabet reduced to a (fixed) constant
    - UNSAT only decreases by a constant factor

How We’ll Prove Main Theorem

- Will work with 2 variable constraints
  - Includes 3-coloring, so NP hard
- Make constraint graph G=(V,E) with vertices = vars, edges = constraints
- Assignment σ : V → Σ
- UNSAT(σ)=fraction of unsatisfied constraints
- Size(G)=|V|+|E|
- Everything we do will preserve satisfiability
- Will apply 3 kinds of steps:
  - Preprocessing (make G nice)
  - Amplification (make UNSAT larger, alphabet bigger)
  - Alphabet reduction (make alphabet =2, keep UNSAT almost the same)
Amplification

- Why can’t we just (in proof viewpoint) repeat verifier a bunch of times?
  - Increases arity, no longer a 2CSP
  - Allowed to increase alphabet, but not arity
  - Instead: graph powering
- Vertices of $G'$ will be same as vertices of $G$
- # of (parallel) edges between $u$ and $v$ will = # of $t$-step walks from $u$ to $v$ in $G$
- $\sum \rightarrow \sum^{d^{t/2}}$
  - Each vertex $v$ will have opinion of values of verts within distance $\lfloor d/2 \rfloor$. (Call this $I(v)$.)
- Constraint for edge $(u,v)$ satisfied if can find an assignment to all (old) vars in $I(v)$ s.t.:
  - Agrees with opinions of both $u$ and $v$
  - Satisfies all constraints in $E \cap I(u) \times I(v)$