Polynomial Time Approximation Schemes for Dense Instances of NP-Hard Problems

Sanjeev Arora David Karger
Marek Karpinski

Overview

- PTASs for dense instances of some NP-Hard problems.

- Several applications: max-cut, bisection, max-k-sat, etc.

Main Strategies

- *Exhaustive Sampling*

- Approximation of *polynomial integer problems*

Definitions

- A graph is $\delta - dense$ if it has $\delta n^2/2$ edges.
- A graph is $everywhere - \delta - dense$ if the minimum degree is δn.
- A graph is $dense$ if it is $\Omega(1) - dense$.
- A graph is $everywhere - dense$ if it is $everywhere - \Omega(1) - dense$.

Theorem 1 *There are PTASs for* $everywhere - dense$ *instances of BISECTION and SEPARATOR.*

Theorem 2 *There are PTASs for* $dense$ *instances of MAX-CUT, MAX-DICUT, MAX-k-SAT, DENSE-k-SUBGRAPH (for* $k = \Omega(n)$*), MAX-HYPERCUT(d), and any MAX-*$\mathcal{SNP}$ *problem.*

Theorem 3 *There are exact algorithms for* $everywhere - dense$ *instances of MIN-k-CUT (when* $k = o(n)$*) and of 3-COLORING.*

Smooth Integer Programs

Definition 1 *A polynomial integer program (PIP) is of the form*

$$\begin{array}{ll} \max & p_0(x_1, ..., x_n) \\ s.t. & l_i \leq p_i(x) \leq u_i \quad \text{for } i = 1, ..., m \\ & x \in \{0,1\}^n \end{array}$$

where $p_0, ..., p_m$ are polynomials. When all p_i's have degree at most d, we call this program a degree d PIP.

Definition 2 *An n-variate, degree-d polynomial has smoothness c if the absolute value of each coefficient of degree i term is at most cn^{d-i}.*

Definition 3 *A c-smooth degree-d PIP is a PIP which the objective function and constraints are c-smooth polynomials with degree at most d.*

Definition 4 *A solution a is said to satisfy a constraint $l_i \leq p_i(x) \leq u_i$ with an additive error δ if $l_i - \delta \leq p_i(a) \leq u_i + \delta$.*

Main theorem

There is a randomized polynomial-time algorithm that approximately solves smooth PIPs in the following sense:

- Input: a feasible c-smooth degree d PIP with n variables, objective function p_0 and K constraints
- Output: a 0/1 solution z satifying $p_0(z_1, ..., z_n) \geq OPT - \epsilon n^d$.
- z satisfies each degree d constraint to within an additive error of ϵn^d.
- Running time: $O((dKn^d)^t)$, where $t = (2ced/\epsilon)^2 = O(1/\epsilon^2)$.
- The algorithm can be derandomized, while increasing the running time by only a polynomial factor.

An example: dense MAX-CUT

Modeling

- Let x be the characteristic vector of a cut, i.e.,

$$x_i = \begin{cases} 1 & \text{if } i \text{ is on the } right \text{ side} \\ 0 & \text{if } i \text{ is on the } left \text{ side} \end{cases}$$

- Let $N(i)$ be the set of neighbors of i
- Let $r_i(x)$ denote the number of neighbors of i that are on the left side of cut x, i.e.,

$$r_i(x) = \sum_{j \in N(i)} (1 - x_j)$$

- Then, the MAX-CUT problem can be expressed as

$$\max \sum_i x_i r_i(x)$$
$$\text{s.t. } x \in \{0,1\}^n$$

An example: dense MAX-CUT

Solving

1. Use exhaustive sampling to estimate $r_i(a)$, where $a = (a_1, ..., a_n)$ is the optimum solution.

2. Replace $r_i(x)$ by $r_i(a)$, converting the quadratic integer program into a linear integer problem.

3. Solve the continuous relaxation of the linear integer program, and use *randomized rounding* to get an integer solution.

An example: dense MAX-CUT

Estimating Coefficients

- Let $\rho_i = r_i(a)$, where a is the optimum solution

- Then, a is also the optimum solution of

$$\begin{array}{ll} \max & \sum_i x_i \rho_i \\ \text{s.t.} & r_i(x) = \rho_i \quad \forall i \\ & x \in \{0,1\}^n \end{array}$$

- Use *exhaustive sampling* to compute e_i such that

$$\rho_i - \epsilon n \leq e_i \leq \rho_i + \epsilon n, \quad \forall i$$

An example: dense MAX-CUT

Exhaustive Sampling

- Take a random sample of $O(\log n)$ vertices.
- Try all possible $n^{O(1)}$ placements, "guessing" placements in the optimum cut.
- For any unsampled vertex i:
  If $|N(i)| = \Omega(n)$, then whp, $\Theta(\log n)$ of its neighbors are in the random sample, and we obtain an estimate of $r_i(a)/|N(i)|$.

An example: dense MAX-CUT

Sampling Lemma

- Let (a_i) be a sequence of n numbers, such that $|a_i| \leq M$
- Let $f > 0$
- Consider a multiset of $s = g \log n$ of the a_i's at random (with replacement)
- The sum q of the elements of the multiset satisfies

$$\frac{qn}{s} \in \sum_i a_i \pm nM\sqrt{\frac{f}{g}}$$

with probability at least $1 - n^{-f}$

An example: dense MAX-CUT

Sampling Lemma applied to MAX-CUT

- Goal: estimate $\rho_i = \sum_{j \in N(i)} (1 - a_j)$
- If $|N(i)| \leq \epsilon n/10$, then estimate 0 for ρ_i
- Otherwise...
  $\Rightarrow$ Choose $g \log n$ indices at random, with $g = O(1/\epsilon^2)$
  $\Rightarrow$ "Guess" their values by exhaustively trying all possibilities

- Let $M = 1$, $f = 2$. Then by the sampling lemma,

  $$\frac{qn}{s} \in \sum_{j \in N(i)} a_j \pm \epsilon n \text{ or } e_i \in \rho_i \pm \epsilon n$$

  with probability at least $1 - 1/n^2$

An example: dense MAX-CUT

Linearizing the Quadratic Program

$$\begin{array}{ll} \max & \sum_i x_i r_i(x) \\ \text{s.t.} & x \in \{0,1\}^n \end{array}$$

$$\Downarrow$$

$$\begin{array}{ll} \max & \sum_i x_i \rho_i \\ \text{s.t.} & r_i(x) = \rho_i \quad \forall i \\ & x \in \{0,1\}^n \end{array}$$

$$\Downarrow$$

$$\begin{array}{ll} \max & \sum_i x_i e_i \\ \text{s.t.} & e_i - \epsilon n \leq r_i(x) \leq e_i + \epsilon n \quad \forall i \\ & x \in \{0,1\}^n \end{array}$$

An example: dense MAX-CUT

Linearizing the Quadratic Program

$$\begin{aligned}
\sum z_i r_i(x) &\geq \sum z_i (e_i - \epsilon n) \\
&\geq \sum z_i e_i - \epsilon n^2 \\
&\geq \sum a_i e_i - \epsilon n^2 \\
&\geq \sum a_i (\rho_i - \epsilon n) - \epsilon n^2 \\
&\geq \sum a_i \rho_i - 2\epsilon n^2 \\
&= \text{MAX-CUT} - 2\epsilon n^2
\end{aligned}$$

An example: dense MAX-CUT

Approximating the Linear Program

$\Rightarrow$ Solve the continuous relaxation

$\Rightarrow$ *Randomized Rounding* lemma:

- Let c and f be positive integers
- Let $y = (y_i)$ be an n-vector, such that $0 \leq y_i \leq 1$, and $a^t y = b$, where $|a_i| \leq c$
- Construct $z = (z_i)$ by setting z_i to 1 with probability y_i, and 0 with probability $1 - y_i$
- Then,

$$a^t z \in b \pm c\sqrt{fn \ln n}$$

with probability at least $1 - n^{-f}$

An example: dense MAX-CUT

$$a^t z \in b \pm c\sqrt{fn \ln n}$$

$$r_i(y) = a^t y = b, |a_i| \leq c = 1$$

Applying the lemma...

$$r_i(z) \in r_i(y) \pm O(\sqrt{n} \ln n)$$

$$\sum z_i r_i(y) \in \sum y_i r_i(y) \pm O(n^{3/2} \ln n)$$

Also,

$$\sum y_i r_i(y) \geq \text{MAX-CUT} - 2\epsilon n^2$$

An example: dense MAX-CUT

Recalling that:

$$r_i(z) \in r_i(y) \pm O(\sqrt{n} \ln n)$$

$$\sum z_i r_i(y) \in \sum y_i r_i(y) \pm O(n^{3/2} \ln n)$$

$$\sum y_i r_i(y) \geq \text{MAX-CUT} - 2\epsilon n^2$$

We get:

$$\begin{aligned} \sum z_i r_i(z) &\geq \sum z_i (r_i(y) - O(\sqrt{n} \ln n)) \\ &\geq \sum z_i r_i(y) - O(n^{3/2} \ln n) \\ &\geq \sum y_i r_i(y) - O(n^{3/2} \ln n) \\ &\geq \text{MAX-CUT} - (2\epsilon + o(1))n^2 \end{aligned}$$

Another Application: Bisection

Large Bisections

- Everywhere-dense graph: min degree δn.
- Let $k \geq \alpha n^2$ be the min bisection.

$$\begin{aligned} k = \quad & \min \quad \sum_i x_i r_i(x) \\ & \text{s.t.} \quad \sum_i x_i = n/2 \\ & \qquad x \in \{0,1\}^n \end{aligned}$$

- Algorithm finds solution within $k + \epsilon n^2 \leq k(1 + \epsilon/\alpha)$.
- Approx solution might not be a bisection. But randomized rounding guarantees $n/2 \pm (\sqrt{n} \log n)$ on each side.
  Hence, we need to move only $O(\sqrt{n} \log n)$, affecting the bisection value by no more than $O(n^{1.5} \log n) = o(n^2)$.

Small Bisections

Here, $k \leq \alpha n^2$.

Lemma 1 *In a minimum bisection, there is one side whose every degree-d vertex has at most $d/2+1$ of its neighbors on the other side.*

Let L_{opt} be this side in the optimum solution, and R_{opt} the other. Consider the following Algorithm:

1. S = random set of $O((\log n)/\delta)$ vertices.

2. For each possible partition of $S = (S_L, S_R)$, build (L, R):
   (a) Let T be the set of vertices that has more than 5/8 of their sampled neighbors in S_R. Put T in R.
   (b) $\forall v \notin T : bias(v) = |N(v)/T| - |N(v) \cap T|$.

   (c) Put the $n/2 - |T|$ vertices with the smallest bias in R.

Small Bisections

$\Rightarrow v$ is radical-right if at least 3/4 of its neighbors are in R_{opt}. Note that every such $v \in R_{opt}$.

Lemma 2 *Whp, $T \subseteq R_{opt}$ and T contains every radical-vertex.*

Proof

- $\forall v$, whp, $\Theta(\log n)$ neighbors of v are sampled.

- If $v \in L_{opt}$, then it has less than 1/2 of it neighbors in R_{opt}. And, wlp more than 5/8 of its **sampled** neighbors are in R_{opt}.

- If v is a radical-vertex, then wlp, less than 5/8 of its **sampled** neighbors are in R_{opt}. Hence, whp, $v \in T$.

Small Bisections

Lemma 3 *If T satisfies the conditions in the previous lemma ($T \subseteq R_{opt}$ and T contains every radical-right vertex), then $|T| \geq \frac{n}{2} - \frac{4k}{\delta n}$.*

Proof

- $v \in (R_{opt} - T) \Rightarrow |N(v) \cap L_{opt}| \geq \frac{1}{4}\delta n$
- Let $s = |R_{opt} - T| = n/2 - |T|$. Then, $k \geq s\frac{1}{4}\delta n$.
- Hence $s \leq \frac{4k}{\delta n}$.

Small Bisections

Theorem 4 *If $k \leq \alpha n^2$, whp, the bisection produced by the algorithm has value at most $k(1+\epsilon)$, where $\epsilon = 16\alpha^2/\delta^2$.*

Proof

- Let $U \subseteq \bar{T}$.
- Let $d_{in}(U) = 2 \times \#$ of edges with both endpoints in U.
- Let $d_{out}(T) = \#$ of edges with exactly one endpoint in T.
- Let $bias(U) =$ sum of the biases of the vertices in U.
- Then, the bisection whose one side is $T \cup U$ has value $d_{out}(T) + bias(U) - d_{in}(U)$.

Small Bisections

Proof(cont...)

- whp, T satisfies the conditions in the previous lemmas. Hence $T \subseteq R_{opt}$ and $s = n/2 - |T| \leq \frac{4k}{\delta n}$.

- Let $U_{opt} = R_{opt} - T$ be the optimum set of s vertices extending T to R_{opt}; and let U_{actual} be the set of s vertices picked by the algorithm extending T to R.

- Then, $k = d_{out}(T) + bias(U_{opt}) - d_{in}(U_{opt})$ And, U_{actual} minimizes $bias(U)$, and also $d_{out}(T) + bias(U)$.

- Therefore, the bisection produced by the algorithm has value at most $k + d_{in}(U_{opt}) - d_{in}(U_{actual}) \leq k + s^2 \leq k + \left(\frac{4k}{\delta n}\right)^2 \leq k(1 + 16\alpha^2/\delta^2)$.

Some corollaries...

Corollary 1 *If a dense graph has bisection value $O(n)$, then the optimum can be found in polynomial time.*

Corollary 2 *There is a PTAS for the optimum separator of a dense graph.*

Algorithms without PIPs

MIN-k-terminal-CUT

- Everywhere-dense graph: min degree δn.

- Let OPT be the capacity of the optimum cut. $OPT \leq kn$.

Lemma 4 *When $k = o(n)$, there is a k-cut of capacity $(1 + o(1))OPT$ that has a special form: at most $2/\delta$ groups of size $\Omega(n)$, and all other groups of size 1 (containing only the terminals).*

Min-k-terminal-cut

Proof

- Let $S_1, S_2, ..., S_k$ be the k groups in the optimum cut, **sorted in decreasing order by size**.

- Let an **ordinary** vertex be one that has at most 1/4 of its neighbors in groups different from its own. Note that there at most $8k/\delta$ such vertices.

- Let t be the # of groups with at least $\delta n/2$ vertices. Note that $t \leq 2/\delta$.

- Consider a modified cut: move all nonterminals from $S_{t+1}, ..., S_k$ to the other groups.

Min-k-terminal-cut

Proof(cont...)

- $|S_i| < \delta n/2$ for $i = t+1, ..., k \Rightarrow S_i$ has no ordinary vertices.

- $OPT \geq \sum_{i=t+1}^{k} \left(|S_i| \frac{\delta n}{2} - \binom{|S_i|}{2} \right)$.

- In modified cut: Increased capacity by at most $\sum_{i=t+1}^{k} \binom{|S_i|}{2}$, which is $o(.)$ of $\sum_{i=t+1}^{k} (|S_i| \delta n/2)$.

- But, $\sum_{i=t+1}^{k} (|S_i| \delta n/2) \leq 8k/\delta \times \delta n/2 = 4kn$. And since, $OPT \leq kn$...

MIN-k-terminal-CUT

The Algorithm

- Exhaustively try all $k^{2/\delta}$ ways of picking terminals that will go into non-singleton groups.

- Pick a random sample of $O(\log n)$ vertices and exhaustively try all partitions.

- For each unsampled vertex, place it in the group that contains more than 2/3 of its sampled neighbors (if such group exists).

- Remaining s vertices: sampling lemma guarantees that they are all non-ordinary. Place them in the group with the greatest number of sampled neighbors.

The Algorithm (cont...)

- Resulting cut may have misplaced s vertices. This increase the cost of our cut in at most $s-1$ for each misplaced vertex.

- The cost of our cut is, then, $OPT(1 + o(1)) + \left(\begin{array}{c} s \\ 2 \end{array} \right)$.

- But, $OPT(1+o(1)) \geq s\delta n/8$, and $s \leq 8k(1+o(1))/\delta = o(n)$.

- Hence, our cut has capacity $OPT(1+o(1))$.

3-COLORING

- Initially, all vertices uncolored.
- Pick a random sample of $O(\log n/\epsilon^2)$ of vertices and "guess" their colors.
- whp, at least one neighbor of every vertex is sampled and colored.
- If a vertex has colored neighbors of 2 different colors, color it with the 3^{rd} color.
- Remaining vertices v_i's: each has a neighbor of color c_i
- Solve the following 2-SAT problem:
  $x_i = true$ if v_i has color $c_i + 1$ mod 3; $x_i = false$ if v_i has color $c_i - 1$ mod 3 (in the optimum solution)
  for each edge (v_i, v_j) add a constraint to ensure that v_i and v_j have different colors.