**Optimization Problems and Approximation**

We are unable to solve NP-complete problems efficiently, i.e., there is no known way to solve them in polynomial time. Most of them are decision versions of optimization problems... with a set of feasible solutions for each instance with an associated quality measure. Why not looking for an approximate solution? Is there a difference in complexity?

---

**Example Knapsack revisited**

All set $T \subseteq S$ satisfy $\sum_{i\in T} w(i) \leq W$ are feasible solutions. $\sum_{i\in T} \nu(i)$ is the quality of the solution $T$ wrt. to the instance $i$.

**Definition of Optimization Problems**

**Example Problem: MaxkSat**

MaxSat has all CNF - Expressions as instances. There is also a weighted version: Each clause has a weight -- the measure is the sum of the weights of the satisfied clauses.

---

**Example Problem: MaxkSat**

Max3Sat(D) is certainly NP - complete (thus Max3Sat is NP - hard): 3SAT is a special case. But also Max2Sat(D) is NP - complete....
Optimization Problems and Approximation

Performance Ratio

Approximation algorithms deliver solutions of guaranteed quality – they are not heuristics.

But how to measure the quality of a solution?

Let \( O := \{s, m, \text{type} \} \) be an optimization problem.

Given \( i \in I \) and a \( s \in \text{sol}(i) \) we define

\[
\mathcal{R}(i, s) = \max \left\{ \frac{\text{opt}(i)}{m(i, s)} : \frac{m(i, s)}{\text{opt}(i)} \right\}
\]

as the performance ratio.

\( s \in \text{sol}(i) \) is an \( r - \text{approximate solution} \) if \( \mathcal{R}(i, s) \leq r \).

Example Problem MaxkSat

Performance Ratio

MaxkSat \( := \{s, m, \text{max} \} \)

\( I \in \text{CNF} - \text{Formulas} \) with at most \( k \) literals per clause

\( \text{sol}(\phi) \) = set of assignments to the vars. of \( \phi \)

\( m(\phi, A) \) = the number of clauses which are satisfied by \( A \)

\[
\mathcal{R}(\phi, A) = \frac{\text{opt}(\phi)}{m(\phi, A)}
\]

If we have an \( A \) with \( \mathcal{R}(\phi, A) \leq \frac{3}{2} \)

no \( A \) can satisfy more than \( \frac{3}{2} m(\phi, A) \) clauses.

Optimization Problems and Approximation

Approximation Problem

Let \( O := \{s, m, \text{type} \} \) be an optimization problem and \( r \) a function \( N \to \{1, \infty\} \).

Then the approximation problem \( \langle O, r \rangle \) is to find for all instances \( i \in I \) an \( r(i) \) - approximate solution \( s \in \text{sol}(i) \).

The question is which approximation problems \( \langle O, r \rangle \) are located in \( \text{FP} \).

And how to prove that they are not (under some assumption such as \( \text{P} \neq \text{NP} \)).

Optimization Problems and Approximation

The Class NPO

NPO is the class of optimization problems whose decision versions are in \( \text{NP} \).

\[
\text{OPTPROB} := \{s, m, \text{type} \} \in \text{NPO} \iff
\]

\( \exists \text{ polynomial } p : \forall i \in I, s \in \text{sol}(i) : |s| \leq p(i) \)

deciding \( s \in \text{sol}(i) \) is in \( \text{P} \)

computing \( m(s, i) \) is in \( \text{FP} \)

Optimization Problems and Approximation

Approximation Algorithm

Example Problem: MaxSat

approxMaxSat(\( \phi \))

1. for \( i = 1 \) to \( n \)
2. \( \text{val} := E(m(\phi, A \cup \{x_i = \text{true}\}) : E(m(\phi, A \cup \{x_i = \text{false}\})) ;
3. \( A := A \cup \{x_i = \text{val}\} ;\) \( \phi := \phi \cup \{x_i = \text{val}\};
4. return \( A;\)

\[
E(\phi) = \sum_{x_i} 1 - 2^{-x_i} \geq \sum_{x_i} 1 - 2^{-x_i} = \frac{1}{2} |\phi|
\]

Thus, this algorithm is a 2-approximate algorithm or better.
Approximation Algorithm

Example Problem: VertexCover

1. \( C := \emptyset \)
2. while \( E \neq \emptyset \) do
3. \( a \leftarrow \langle u, v \rangle \in E \)
4. \( C := C \cup \{u, v\} \)
5. remove \( \{u, v\} \) from \( V, E \)
6. return \( C \)

C is indeed a valid cover. Every cover must cover all the edges picked in line 3. Thus every cover must contain at least \( |C| / 2 \) vertexes.

\[ R(G, C) = \frac{m(G, C)}{\text{opt}(G)} \leq 2 \]

Approximation Classes

APX

We have two approximation problems, which can be solved within a constant performance ratio within polynomial time.

So it’s time to define a corresponding class: APX.

Let \( O \) be an NPO problem. \( O \in \text{APX} \) if there exists an \( r \) -approximation algorithm for \( O \) which run in polynomial time for some constant \( r \geq 1 \).

Approximation Classes

Example Problem: TSP (I)

We will show that \( \text{TSP} \in \text{APX} \Leftrightarrow P = \text{NP} \).

We use another NP-complete problem to reduce from: \( \text{HAMILTONIANCYCLE} \)

\( \text{HAMILTONIANCYCLE} \): Given a graph \( G = (V, E) \), is there a cycle, which visits any node exactly once?

We construct a distance matrix \( M \) as follows (for \( r \geq 1 \)):

\[
M(u, v) = \begin{cases} 
1 & \langle u, v \rangle \in E \\
\left\lfloor \frac{r |V|}{|V|} \right\rfloor & \text{otherwise} 
\end{cases}
\]

Approximation Classes

Example Problem: TSP (II)

We construct a distance matrix \( M \) as follows (for \( r \geq 1 \)):

\[
M(u, v) = \begin{cases} 
1 & \langle u, v \rangle \in E \\
\left\lfloor \frac{r |V|}{|V|} \right\rfloor & \text{otherwise} 
\end{cases}
\]

If \( G \) is a positive instance, then \( \text{opt}(M) = |V| \).
Otherwise \( \text{opt}(M) \geq \left\lfloor \frac{r |V|}{|V|} \right\rfloor - 1 \).

Now assume that there is an \( r \)-approximate algorithm for \( \text{TSP} \).

Approximation Classes

Example Problem: TSP (III)

If \( G \) is a positive instance, then \( \text{opt}(M) = |V| \).
Otherwise \( \text{opt}(M) \geq \left\lfloor \frac{r |V|}{|V|} \right\rfloor - 1 \).

Now assume that there is an \( r \)-approximate algorithm for \( R(M) \).

If \( G \in \text{HAMILTONIANCYCLE} \), we find

\[
r \geq R(M, x) = \frac{m(M, x)}{\text{opt}(M)} \geq \frac{m(M, x)}{|V|} \quad \text{and so} \quad |V| r \geq m(M, x).
\]

But otherwise we have

\[
m(M, x) \geq \text{opt}(M) \geq \left\lfloor \frac{r |V|}{|V|} \right\rfloor - 1 > \left\lfloor \frac{r |V|}{|V|} \right\rfloor
\]

Approximation Classes

Example Problem: TSP (IV)

So we could prove that \( \text{TSP} \not\in \text{APX} \) (assuming \( P \neq \text{NP} \)) by giving a reduction from an \( \text{NP} \)-hard problem, which established a gap between positive and negative instances.

The gap was large enough to distinguish whether we reduced from a positive or a negative instance.

Wanted: A generic reduction from \( \text{NP} \)-hard problems, to approximation problems which produces gaps.
Approximation Classes

Relationships

\[ \text{APX} \subseteq \text{NPO} \]

\[ \text{TSP} \in \text{APX} \iff \text{P} = \text{NP} \]

\[ \text{APX} \subseteq \text{NPO} \iff \text{P} \neq \text{NP} \]

Max3Sat and VertexCover are in APX.

Approximation Classes

Approximation Schemes

An algorithm which can be parametrized with the performance ratio to achieve is called approximation scheme.

Let \( O := (I, \text{sol}, m, \text{type}) \) be an optimization problem. Then an algorithm \( A \) is an approximation scheme for \( O \) iff for all \( i \in I \), \( r \geq 1 \) and \( s = A(i, r) \)\n
\[ s \in \text{sol}(i) \text{ and } R(i, s) \leq r. \]

Approximation Schemes

The classes PTAS and FPTAS

\[ O \in \text{FPTAS} \text{ if there is an approximation scheme } A \]

such that \( A(i, r) \) runs in \( \text{DTIME}(\text{poly}(|i|/ (r-1))) \)

for all \( i \in I \) and \( r > 1 \).

\[ O \in \text{PTAS} \text{ if there is an approximation scheme } A \]

such that \( A(i, r) \) runs in \( \text{DTIME}(\text{poly}(|i|)) \)

for all \( i \in I \) and any fixed \( r > 1 \).

Example Problem: KNAPSACK

A Pseudo-Polynomial Algorithm

Let \( W(x, v) \) be the minimum weight attainable by selecting among the first \( x \) items such that their total value is exactly \( v \).

\[ W(0, 0) = 0 \]

\[ W(0, v) = \infty \text{ if } v > 0 \]

\[ W(x+1, v) = \min \{ W(x, v), W(x, v - (x+1)) + w(x+1) \} \]

By building the table of the \( W(x, v) \) for \( 0 \leq x \leq n \) and \( 0 \leq v \leq V = \sum v(x) \) we can solve KNAPSACK.

This algorithm runs in \( \text{DTIME}(|n, V|) \) (pseudo-poly.)

Example Problem: KNAPSACK

An FPTAS (I)

This algorithm runs in \( \text{DTIME}(\text{poly}(n, V)) \) (pseudo-poly.)

Assume \( \varepsilon > 0 \) fixed.

Let \( l = \left\lfloor \log_{\log_{n}} v(x) \right\rfloor \)

Choose \( k \) with \( \frac{n}{n} < \varepsilon \).

Set \( L = l - k \log n \).

Define \( v' \) with \( v'(x) = \left\lfloor v(x)/2^{l-k} \right\rfloor \).

We keep the most significant \( k \) log \( n \) bits.

The rest, i.e., \( L = l - k \log n \), gets zeroized.
Example Problem: KNAPSACK
An FPTAS (II)
This algorithm runs in $\text{DTIME}(\text{poly}(n,V^2))$ (pseudo-poly.)
Assume $\varepsilon > 0$ fixed.
Let $l = \left\lfloor \log \max_{x \in S} v(x) \right\rfloor$
Choose $k$ with $\frac{n}{\varepsilon} < k$.
Set $L = l - k \log n$.
Define $i'$ with
$v'(x) = \left\lfloor \frac{v(x)}{2^l} \right\rfloor$
$$\sum_{i \in S} v(x) + \sum_{i' \in S} v'(x) + |T| 2^l$$
$$\text{opt}(i) \leq \text{opt}(i') + n2^l$$
$$\text{opt}(i') \leq 1 + \frac{n2^l}{\max_{x \in S} v(x)}$$
$$\leq 1 + \frac{n2^l}{\varepsilon} \leq 1 + \varepsilon$$

Example Problem: KNAPSACK
An FPTAS (III)
This algorithm runs in $\text{DTIME}(\text{poly}(n,V^2))$ (pseudo-poly.)
Assume $\varepsilon > 0$ fixed.
Let $l = \left\lfloor \log \max_{x \in S} v(x) \right\rfloor$
Choose $k$ with $\frac{n}{\varepsilon} < k$.
Set $L = l - k \log n$.
Define $i'$ with
$v'(i') = \left\lfloor \frac{v(i')}{2^l} \right\rfloor$
$$\sum_{i \in S} v(x) + \sum_{i' \in S} v'(x)$$
$$\text{opt}(i) \leq \text{opt}(i') + \frac{n2^l}{\max_{x \in S} v(x)}$$
$$\leq 1 + \frac{n2^l}{\varepsilon} \leq 1 + \varepsilon$$
Solving $i'$ optimally yields an $1 + \varepsilon$ approximate solution for $I$

Example Problem: KNAPSACK
An FPTAS (IV)
This algorithm runs in $\text{DTIME}(\text{poly}(n,V^2))$ (pseudo-poly.)
Assume $\varepsilon > 0$ fixed.
Let $l = \left\lfloor \log \max_{x \in S} v(x) \right\rfloor$
Choose $k$ with $\frac{n}{\varepsilon} < k$.
Set $L = l - k \log n$.
Define $i'$ with
$v'(x) = \left\lfloor \frac{v(x)}{2^l} \right\rfloor$
We can solve $i'$ in
$$\text{DTIME}(\text{poly}(n, V^2))$$
$$= \text{DTIME}(\text{poly}(n, n2^{\log n}))$$
$$= \text{DTIME}(\text{poly}(n/\varepsilon))$$
Solving $i'$ optimally yields an $1 + \varepsilon$ approximate solution for $I$
within $\text{DTIME}(\text{poly}(l/\varepsilon))$, KNAPSACK $\in$ FPTAS.

Approximation Schemes
Polynomially Bound Problems
Let $O = \langle I, \text{sol, m, type} \rangle$ be a problem in NPO.
If there is no polynomial $p$ such that
$$\forall t_i \in I \exists \text{sol}(i): m(t_i, x) \leq p(t_i)$$
then $O$ is polynomially bound, i.e.,
$$O \in \text{NPO} - \text{PB}$$
If there is an NP-hard problem in NPO - PB which admits an FPTAS, then $P = NP$.

Polynomially Bound Problems
Permit no FPTAS (I)
If there is an NP-hard problem in NPO - PB which admits an FPTAS, then $P = NP$.
Let $O$ be a maximization problem in NPO - PB.
Set $r(i) = 1 + \frac{1}{p(i)}$, where $p$ is the poly.-bound.
An $r(i)$-approximate solution $s$ for $i$ is optimal since,
$$p(i) \cdot (r(i) + 1) = r(i) \geq \frac{\text{opt}(i)}{\text{m}(s)}$$
gives
$$\text{m}(s, x) \geq \text{opt}(i) \cdot \frac{p(i)}{p(i) + 1} = \frac{\text{opt}(i)}{\text{m}(s)} > \text{opt}(i) - 1$$

Polynomially Bound Problems
Permit no FPTAS (II)
Set $r(i) = 1 + \frac{1}{p(i)}$, where $p$ is the poly.-bound.
An $r(i)$-approximate solution $s$ for $i$ is optimal since,
$$p(i) \cdot (r(i) + 1) = r(i) \geq \frac{\text{opt}(i)}{\text{m}(s)}$$
gives
$$\text{m}(s, x) \geq \text{opt}(i) \cdot \frac{p(i)}{p(i) + 1} = \frac{\text{opt}(i)}{\text{m}(s)} > \text{opt}(i) - 1$$
If $O$ would be in FPTAS then we can solve $O$ optimally
in $\text{DTIME}(\text{poly}(|I|, (r(i) - 1))) = \text{DTIME}(\text{poly}(i))$.  

### Approximation Classes

#### Relationships

- \( \text{FPTAS} \subseteq \text{PTAS} \subseteq \text{APX} \subseteq \text{NPO} \)

- \( \text{TSP} \in \text{APX} \iff P = \text{NP} \)
- \( \text{Max3Sat} \in \text{FPTAS} \iff P = \text{NP} \)

Two questions:
- Are there problems in \( \text{PTAS-FPTAS} \)?
- Are there problems in \( \text{APX-PTAS} \)? (as usual, based on \( P = \text{NP} \))

#### Problems in PTAS-FPTAS

- PLANAR INDEPENDENTSET is in NPO - PB and is NP-hard.
- PLANAR INDEPENDENTSET \( \in \text{FPTAS} \) \( \Rightarrow P = \text{NP} \).

Unproven: PLANAR INDEPENDENTSET \( \in \text{PTAS} \).

### Hardness in Approximation

#### PCP-Theorem (I)

A language \( L \) is in \( \text{PCP}(r(n), q(n)) \)

- \( \forall x \in L \ \exists \Pi: \text{Pr}_{\tilde{V}}[V(x, \Pi, \tilde{R}) = \text{accept}] = 1 \)
- \( \forall x \notin L \ \forall \Pi: \text{Pr}_{\tilde{V}}[V(x, \Pi, \tilde{R}) = \text{accept}] \leq 1/2 \)

with \( \tilde{R} = O(1/\epsilon) \), and \( V \) reading \( O(q(n)) \) bits non-adaptively from \( \Pi \).

**Easy**: \( \text{NP} \nsubseteq \text{PCP}(\log n, 1) \)

**Hard**: \( \text{NP} \nsubseteq \text{PCP}(\log n, 1) \)

#### PCP-Verification

1. reads the input \( x \)
2. reads \( O(r(n)) \) random bits
3. computes proof positions to read
4. reads \( O(q(n)) \) proof bits
5. decides
Hardness in Approximation

PCP-Theorem (I)

A language $L$ is in $\text{PCP}(r(n), q(n))$ if there is a polynomial time $\text{PCP}(r(n), q(n))$-Verifier $V$ such that

\[
\forall x \in L \implies \exists \Pi: \Pr[V(x, \Pi, \overline{R}) = \text{accept}] = 1
\]

\[
\forall x \notin L \implies \forall \Pi: \Pr[V(x, \Pi, \overline{R}) = \text{accept}] \leq \frac{1}{2}
\]

with $|\overline{R}| = O(r(n))$, and $V$ reading $O(q(n))$ bits non-adaptively from $\Pi$.

PCP-Theorem: $NP = PCP(\log n, 1)$

How to use?

Reduce the verification process to an approximation problem such that the gap of the PCP-Verifier translates into a gap in the measure of the optimal solution(s).

Example Problem: Max3Sat (I)

Observe that once the $O(q(n))$ bits have been read from the proof $\Pi$, the decision of $V$ is only depending on them.

Thus we can define a set of Boolean Expressions $\phi(x, \overline{R}[\overline{p}])$ where $x$ is the input, $\overline{R}$ is the random string of length $O(\log n)$, $\overline{p}$ are the bits read in $\Pi$, $\phi(x, \overline{R}[\overline{p}]) = 1 \iff V(x, \Pi, \overline{R}) = \text{accept}$.

Example Problem: Max3Sat (II)

Each $\phi(x, \overline{R}[\overline{p}])$ can be expressed by $d$ clauses, where $d$ is constant (since $|\overline{R}|$ is constant).

Let $\phi$ be the conjunction of the expressions $\phi(x, \overline{R}[\overline{p}])$ for all $\overline{R} (|\overline{R}| = c \log n)$.

\[
x \in L \implies \exists \Pi: \Pr[V(x, \Pi, \overline{R}) = \text{accept}] = 1
\]

\[
\implies \forall \phi, \overline{R} \text{can be satisfied simultaneously}
\]

\[
\implies \phi \text{ satisfiable.}
\]

Example Problem: Max3Sat (III)

Each $\phi(x, \overline{R}[\overline{p}])$ can be expressed by $d$ clauses, where $d$ is constant (since $|\overline{R}|$ is constant).

Let $\phi$ be the conjunction of the expressions $\phi(x, \overline{R}[\overline{p}])$ for all $\overline{R} (|\overline{R}| = c \log n)$.

\[
x \in L \implies \forall \Pi: \Pr[V(x, \Pi, \overline{R}) = \text{accept}] \leq \frac{1}{2}
\]

\[
\implies \text{each assignment must leave } \frac{1}{2}
\]

\[
\text{of the expressions } \phi(x, \overline{R}) \text{ unsatisfied.}
\]

\[
\implies \text{opt}(\phi) = 1
\]

\[
\text{opt}(\phi) \leq f = \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
\]

\[
x \in L \implies \text{opt}(\phi) = f = |\phi|
\]

\[
x \notin L \implies \text{opt}(\phi) \leq f = |\phi|
\]

\[
\text{for all } A.
\]

\[
r-\text{approximating Max3Sat is } NP-hard \text{ (constant } r > 1). \quad \blacksquare
\]

Example Problem: Max3Sat (IV)

Each $\phi(x, \overline{R}[\overline{p}])$ can be expressed by $d$ clauses, where $d$ is constant (since $|\overline{R}|$ is constant).

Let $\phi$ be the conjunction of the expressions $\phi(x, \overline{R}[\overline{p}])$ for all $\overline{R} (|\overline{R}| = c \log n)$.

\[
x \in L \implies \forall \Pi: \Pr[V(x, \Pi, \overline{R}) = \text{accept}] = 1
\]

\[
x \notin L \implies \forall \Pi: \Pr[V(x, \Pi, \overline{R}) = \text{accept}] \leq \frac{1}{2}
\]

\[
x \in L \implies \text{opt}(\phi) = f = |\phi|
\]

\[
x \notin L \implies \text{opt}(\phi) \leq f = |\phi|
\]

\[
\text{for all } A.
\]

\[
r-\text{approximating Max3Sat is } NP-hard \text{ (constant } r > 1). \quad \blacksquare
\]
Hardness in Approximation

Remark: Decoding of PCP-Proofs

\[ \forall x \in L \exists \Pi : P_{\mathbb{E}}^{\Pi}(x, \Pi, \overline{\Pi}) = \text{accept} = 1 \]
\[ \forall x \notin L \forall \Pi : P_{\mathbb{E}}^{\Pi}(x, \Pi, \overline{\Pi}) = \text{accept} \leq 1/2 \]

Given a proof \( \Pi \) with \( P_{\mathbb{E}}^{\Pi}(x, \Pi, \overline{\Pi}) = \text{accept} > 1/2 \), a proof \( \Pi' \) with \( P_{\mathbb{E}}^{\Pi'}(x, \Pi', \overline{\Pi'}) = \text{accept} \) can be reconstructed efficiently (in FP).

\( \Pi \) is basically encoded for error-correction --- thus it possible to find the corresponding "usually encoded" proof efficiently.

Approximation Classes

Relationships

\[ \text{FPTAS} \subseteq \text{PTAS} \subseteq \text{APX} \subseteq \text{NPO} \]

\[ \text{FPTAS} \subseteq \text{PTAS} \subseteq \text{APX} \subseteq \text{NPO} \iff P \neq \text{NP} \]

Approximation Classes

More Classes

Let \( O \) be an NPO problem.
\( O \in \text{F} \iff \text{APX} \) iff there exists an \( r \)-approximation algorithm for \( O \) which run in polynomial time for some function \( f \in \text{F} \).

\[ \text{FPTAS} \subseteq \text{PTAS} \subseteq \text{APX} \subseteq \text{log} - \text{APX} \subseteq \text{poly} - \text{APX} \subseteq \text{exp} - \text{APX} \subseteq \text{NPO} \]