Promise Problems

Panagiotis Theofilopoulos

$\mu \prod \lambda \forall$

June 22, 2012

Table of Contents

Promise problems overview

- Introduced and initially studied by Even, Selman and Yacobi.
- Can be thought as a generalization of language-recognition problems.
- Every decision problem can be viewed as a promise problem:
  - In some cases the promise is trivial or tractable
  - In some cases Promise problems appear to be the most appropriate representation of natural decision problems

Promise problems overview

- Introduced and initially studied by Even, Selman and Yacobi.
- Can be thought as a generalization of language-recognition problems.
- Every decision problem can be viewed as a promise problem:
  - In some cases the promise is trivial or tractable
  - In some cases Promise problems appear to be the most appropriate representation of natural decision problems

Q Do they provide a really useful framework?

Q How are they connected with the familiar language-recognition problems?

Q What are the implications of studing the complexity of promise problems?

Informal description

A Promise problem is a partition of the set of all strings over an alphabet into three subsets:

1. The set of strings representing YES-instances
2. The set of strings representing NO-instances
3. The set of disallowed strings (representing neither YES-instances nor NO-instances)

Informal description

A Promise problem is a partition of the set of all strings over an alphabet into three subsets:

1. The set of strings representing YES-instances
2. The set of strings representing NO-instances
3. The set of disallowed strings (representing neither YES-instances nor NO-instances)

An algorithm solving a Promise problem is required to distinguish YES-instances from NO-instances and is allowed *arbitrary* behaviour on disallowed strings.

Motivation: Hard promise

Motivation: Hard promise

Observe that there are two alternatives to promise problems in the case of the description of a decision problem:

Motivation: Hard promise

Observe that there are two alternatives to promise problems in the case of the description of a decision problem:

- meaningless (non-canonical) representations are interpreted as a representation of some fixed instance.
- meaningless representations are interpreted as NO-instances.

Motivation: Hard promise

Observe that there are two alternatives to promise problems in the case of the description of a decision problem:

- meaningless (non-canonical) representations are interpreted as a representation of some fixed instance.
- meaningless representations are interpreted as NO-instances.

But imagine a problem with a hard promise:

Example

Given a Hamiltonian graph, determine whether or not...

Motivation: Hard promise

Observe that there are two alternatives to promise problems in the case of the description of a decision problem:

- meaningless (non-canonical) representations are interpreted as a representation of some fixed instance.
- meaningless representations are interpreted as NO-instances.

But imagine a problem with a hard promise:

Example

Given a Hamiltonian graph, determine whether or not...

Both alternatives fail: the first cannot be implemented and the second could substantially affect the complexity of the problem.

Table of Contents

Definition of Promise Problems

Definition (Promise problem)

A promise problem Π *is a pair of sets* $(\Pi_{\text{yes}}, \Pi_{\text{no}})$ *such that* $\Pi_{\text{yes}}, \Pi_{\text{no}} \subseteq \{0,1\}^*$ *and* $\Pi_{\text{yes}} \cap \Pi_{\text{no}} = \emptyset$*.*

The set $\Pi_{\text{yes}} \cup \Pi_{\text{no}}$ is called the *promise*.

Classes of Promise problems

Definition ($\mathbf{P}$ in terms of Promise problems)

A promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ *is in* $\mathbf{P}$ *if there exists a deterministic polynomial-time algorithm* M *such that:*

- $\forall x \in \Pi_{\text{yes}} \Longrightarrow M(x) = 1$
- $\forall x \in \Pi_{\text{no}} \Longrightarrow M(x) = 0$

Classes of Promise problems

Definition ($\mathbf{P}$ in terms of Promise problems)

A promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ *is in* $\mathbf{P}$ *if there exists a deterministic polynomial-time algorithm* M *such that:*

- $\forall x \in \Pi_{\text{yes}} \implies M(x) = 1$
- $\forall x \in \Pi_{\text{no}} \implies M(x) = 0$

Definition ($\mathbf{NP}$ in terms of Promise problems)

A promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ *is in* $\mathbf{NP}$ *if there exists a polynomially bounded binary relation* R *recognized by a polynomial-time deterministic algorithm such that:*

- $(\forall x \in \Pi_{\text{yes}})(\exists y)[(x, y) \in R]$
- $(\forall x \in \Pi_{\text{no}})(\forall y)[(x, y) \notin R]$

Classes of Promise problems

Definition ($\mathbf{BPP}$ in terms of Promise problems)

A promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ *is in* $\mathbf{BPP}$ *if there exists a probabilistic polynomial-time algorithm* M *such that:*

- $\forall x \in \Pi_{\text{yes}} \implies \Pr\left[M(x) = 1\right] \geq \frac{2}{3}$
- $\forall x \in \Pi_{\text{no}} \implies \Pr\left[M(x) = 0\right] \geq \frac{2}{3}$

Classes of Promise problems

Definition ($\mathbf{BPP}$ in terms of Promise problems)

A promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ *is in* $\mathbf{BPP}$ *if there exists a probabilistic polynomial-time algorithm* M *such that:*

- $\forall x \in \Pi_{\text{yes}} \implies \Pr\left[M(x) = 1\right] \geq \frac{2}{3}$
- $\forall x \in \Pi_{\text{no}} \implies \Pr\left[M(x) = 0\right] \geq \frac{2}{3}$

Other classes can be defined in the same way: the conditions used in the standard definition are applied to the partition $\Pi_{\text{yes}} \cup \Pi_{\text{no}}$ of all possible inputs and nothing is required with respect to inputs that violate the promise.

Reductions of Promise problems

Definition (Karp reduction)

A Promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ *is Karp-reducible to the problem* $\Pi' = (\Pi'_{\text{yes}}, \Pi'_{\text{no}})$ *if there exists polynomial-time computable function* f *such that:*

- $\forall x \in \Pi_{\text{yes}} \implies f(x) \in \Pi'_{\text{yes}}$
- $\forall x \in \Pi_{\text{no}} \implies f(x) \in \Pi'_{\text{no}}$

Reductions of Promise problems

Definition (Karp reduction)

A Promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ *is Karp-reducible to the problem* $\Pi' = (\Pi'_{\text{yes}}, \Pi'_{\text{no}})$ *if there exists polynomial-time computable function* f *such that:*

- $\forall x \in \Pi_{\text{yes}} \implies f(x) \in \Pi'_{\text{yes}}$
- $\forall x \in \Pi_{\text{no}} \implies f(x) \in \Pi'_{\text{no}}$

Definition (Cook reduction)

A Promise problem $\Pi = (\Pi_{\text{yes}}, \Pi_{\text{no}})$ *is Cook-reducible to the problem* $\Pi' = (\Pi'_{\text{yes}}, \Pi'_{\text{no}})$ *if there exists polynomial-time oracle machine* M *such that:*

- $\forall x \in \Pi_{\text{yes}} \implies M^{\Pi'}(x) = 1$
- $\forall x \in \Pi_{\text{no}} \implies M^{\Pi'}(x) = 0$

Reductions of Promise problems

Remark

The query q to oracle Π' is answered as follows:

- 1, if $q \in \Pi'_{\text{yes}}$
- 0, if $q \in \Pi'_{\text{no}}$
- arbitrarily, otherwise

Reductions of Promise problems

Remark

The query q to oracle Π' is answered as follows:

- 1, if $q \in \Pi'_{\text{yes}}$
- 0, if $q \in \Pi'_{\text{no}}$
- arbitrarily, otherwise

Randomized reductions can be defined analogously.

Table of Contents

Finding unique solutions

Finding unique solutions

- The intractability of SAT does not seem to be due to instances that have many solutions.

Finding unique solutions

- The intractability of SAT does not seem to be due to instances that have many solutions.
- Contrary, SAT instances having very few satisfying assignments are hard.

Finding unique solutions

- The intractability of SAT does not seem to be due to instances that have many solutions.
- Contrary, SAT instances having very few satisfying assignments are hard.
- In the extreme case, distinguishing *uniquely* satisfiable formulae from unsatisfiable ones in not easier than distinguishing satisfiable formulae from unsatisfiable ones.[5]

Finding unique solutions

Definition

For any boolean predicate Q, the problem USAT_Q is defined as follows:

$$\mathrm{USAT}_Q(x) = \begin{cases} 0 & \#\mathrm{SAT}(x) = 0 \\ 1 & \#\mathrm{SAT}(x) = 1 \\ Q(x) & \#\mathrm{SAT}(x) > 1 \end{cases}$$

Theorem (Valiant, Vazirani [5])

There is a randomized polynomial-time reduction from SAT to USAT_Q for **any** *boolean predicate Q.*

Finding unique solutions

The problem of distinguishing between uniquely satisfiable and unsatisfiable formulae can be easily formulated in terms of Promise problems:

Definition

The problem uSAT *is the Promise problem with*

- *YES-instances the formulae that have a unique satisfying assignment*
- *NO-instances the formulae that have no satisfying assignment*

Finding unique solutions

The problem of distinguishing between uniquely satisfiable and unsatisfiable formulae can be easily formulated in terms of Promise problems:

Definition

The problem uSAT *is the Promise problem with*

- *YES-instances the formulae that have a unique satisfying assignment*
- *NO-instances the formulae that have no satisfying assignment*

Now the Valiant-Vazirani Theorem can be stated as follows:

Theorem

There exists a randomized Cook-reduction of SAT *to* uSAT.

Observations and Benefits

- It seems that the notion of Promise problems is necessary for handling "unique solution" problems.
- The formulation in terms of Promise problems leads to a proper definition of "unique solution" problems that also captures the essence of their hardness: distinguishing instances with a unique solution from instances with no solution.

Approximately counting the number of solutions

Approximately counting the number of solutions

Given a relation R recognized by a polynomial-time algorithms we are interested in determining the number of certificates of an instance x, that is the cardinality of the set $R_{\text{certs}}(x) \stackrel{\text{def}}{=} \{y \mid (x, y) \in R\}$. We denote this problem by $\#R$.

Approximately counting the number of solutions

Given a relation R recognized by a polynomial-time algorithms we are interested in determining the number of certificates of an instance x, that is the cardinality of the set $R_{\text{certs}}(x) \stackrel{\text{def}}{=} \{y \mid (x, y) \in R\}$. We denote this problem by $\#R$.

Remark

$\#R$ is not easier than the decision problem: for a given x the decision version asks whether $|R_{\text{certs}}(x)|$ is positive or zero.

Approximately counting the number of solutions

... but we are also interested in approximating $|R_{\text{certs}}(x)|$ up to a factor $f(|x|)$, $f : \mathbb{N} \to \{r \in \mathbb{R} : r \geq 1\}$, that is finding solutions SOL for which $|R_{\text{certs}}(x)|/f(|x|) \leq \mathrm{SOL} \leq |R_{\text{certs}}(x)| \cdot f(|x|)$.

Approximately counting the number of solutions

...but we are also interested in approximating $|R_{\text{certs}}(x)|$ up to a factor $f(|x|)$, $f : \mathbb{N} \to \{r \in \mathbb{R} : r \geq 1\}$, that is finding solutions SOL for which $|R_{\text{certs}}(x)|/f(|x|) \leq \text{SOL} \leq |R_{\text{certs}}(x)| \cdot f(|x|)$. This problem can be formulated in terms of Promise problems by reducing it to the following promise problem:

Definition

The problem $\#R^f$ is the Promise problem with

- *YES-instances the pairs (x, N) such that $|R(x)_{\text{certs}}| \geq N$*
- *NO-instances the pairs (x, N) such that $|R(x)_{\text{certs}}| < N/f(|x|)$*

Approximately counting the number of solutions

Observe that $\#R^f$ is at least as hard as deciding L_R:

Approximately counting the number of solutions

Observe that $\#R^f$ is at least as hard as deciding L_R:

$$x \in L_R \Longrightarrow |R(x)_{\text{certs}}| \geq 1 \Longrightarrow (x, 1) \in \text{YES}$$
$$x \notin L_R \Longrightarrow |R(x)_{\text{certs}}| = 0 \Longrightarrow |R(x)_{\text{certs}}| < 1 \Longrightarrow (x, 1) \in \text{NO}$$

Approximately counting the number of solutions

Observe that $\#R^f$ is at least as hard as deciding L_R:

$$x \in L_R \implies |R(x)_{\text{certs}}| \geq 1 \implies (x,1) \in \text{YES}$$
$$x \notin L_R \implies |R(x)_{\text{certs}}| = 0 \implies |R(x)_{\text{certs}}| < 1 \implies (x,1) \in \text{NO}$$

Interestingly, $\#R^f$ is not much harder than deciding L_R:

Approximately counting the number of solutions

Observe that $\#R^f$ is at least as hard as deciding L_R:

$$x \in L_R \implies |R(x)_{\text{certs}}| \geq 1 \implies (x, 1) \in \text{YES}$$
$$x \notin L_R \implies |R(x)_{\text{certs}}| = 0 \implies |R(x)_{\text{certs}}| < 1 \implies (x, 1) \in \text{NO}$$

Interestingly, $\#R^f$ is not much harder than deciding L_R:

Theorem ([4],[1])

For every $f : \mathbb{N} \to \mathbb{R}$ *such that* $f(n) > 1 + (1/\text{poly(n)})$, *the problem* $\#R^f_{\text{SAT}}$ *is randomly Karp-reducible to* SAT.

Observations and Benefits

- Appealing approach when one wants to establish the hardness of obtaining an approximation of the optimal value.

Gap problems

Gap problems

Gap problems, for example in the maximization case, can be seen as Promise problems having YES-instances with relative high optimum value and NO-instances with relative low optimum value.

Gap problems

Gap problems, for example in the maximization case, can be seen as Promise problems having YES-instances with relative high optimum value and NO-instances with relative low optimum value.

Example

The corresponding with $\mathrm{Max3SAT}$, *gap problem* $\mathrm{gap3SAT}_s$, *is the Promise problem with*

- *YES-instances the satisfiable* $3\mathrm{CNF}$ *formulae*
- *NO-instances the* $3\mathrm{CNF}$ *formulae for which every assignment that satisfies less than an* s *fraction of its clauses*

Hastad showed [2] that for every $\epsilon > 0$ $\mathrm{gap3SAT}_{(7/8)+\epsilon}$ *is* $\mathbf{NP}$*-hard under Karp-reductions.*

Gap problems

Now consider the assertion "For every $\epsilon > 0$ it is $\mathbf{NP}$-hard to approximate $\mathrm{Max3SAT}$ within a factor of $(7/8) + \epsilon$".

Gap problems

Now consider the assertion "For every $\epsilon > 0$ it is $\mathbf{NP}$-hard to approximate $\mathrm{Max3SAT}$ within a factor of $(7/8) + \epsilon$".

Q Does this assertion capture the full strength of Hastad's result?

Gap problems

Now consider the assertion "For every $\epsilon > 0$ it is $\mathbf{NP}$-hard to approximate $\mathrm{Max3SAT}$ within a factor of $(7/8) + \epsilon$".

Q Does this assertion capture the full strength of Hastad's result?

A No!

Gap problems

Now consider the assertion "For every $\epsilon > 0$ it is $\mathbf{NP}$-hard to approximate $\mathrm{Max3SAT}$ within a factor of $(7/8) + \epsilon$".

Q Does this assertion capture the full strength of Hastad's result?

A No! For example, consider the following question: Given a **satisfiable** $\mathrm{3CNF}$ formula, can we find an assignment that satisfies 90% of its clauses?

- The fact that $\mathrm{gap3SAT}_{(7/8)+\epsilon}$ is $\mathbf{NP}$-hard rules out this possibility.
- However, the assertion above tells us nothing about it.

Gap problems

Given a $3\mathrm{CNF}$ formula:

1. Try to find an assignment that satisfies 90% of the clauses
2. Check whether the returned assignment satisfies 90% of the clauses
   - If it does then the formula is either a YES-instance of $\mathrm{gap3SAT}_{9/10}$ or it is a disallowed instance. In any case, answer 'YES'.
   - Else the formula is a NO-instance of $\mathrm{gap3SAT}_{9/10}$, or it is a disallowed instance. In any case, answer 'NO'.

Observations and Benefits

- Promise problems provide useful expressiveness which is necessary for capturing the full extend of some results.

A complete problem for $\mathbf{BPP}$

A complete problem for $\mathbf{BPP}$

In terms of language recognition, no complete problem is known for $\mathbf{BPP}$.

A complete problem for $\mathbf{BPP}$

In terms of language recognition, no complete problem is known for $\mathbf{BPP}$. However, in terms of promise problems, there is the followin complete problem for the class $\mathbf{promise{-}BPP}$:

- YES-instances are Boolean circuits that evaluate to 1 on at least a $2/3$ fraction of their inputs
- NO-instances are Boolean circuits that evaluate to 0 on at least a $2/3$ fraction of their inputs

A complete problem for $\mathbf{BPP}$

In terms of language recognition, no complete problem is known for $\mathbf{BPP}$. However, in terms of promise problems, there is the follown complete problem for the class $\mathbf{promise{-}BPP}$:

- YES-instances are Boolean circuits that evaluate to 1 on at least a $2/3$ fraction of their inputs
- NO-instances are Boolean circuits that evaluate to 0 on at least a $2/3$ fraction of their inputs

A reduction of a problem $\Pi \in \mathbf{promise{-}BPP}$ to this complete problem maps input x to circuit C_x, which on input r emulates the computation of M_Π on input x and random tape r.

A complete problem for $\mathbf{BPP}$

Let A be our $\mathbf{BPP}$-complete problem.

$$\begin{aligned}
x \in \Pi_{\text{yes}} &\Longrightarrow M_\Pi = \text{yes} \text{ with prob } \geq 2/3 \\
&\Longrightarrow C_{M_\Pi, x} = 1 \text{ for at least } 2/3 \text{ of all random inputs } r \\
&\Longrightarrow C_{M_\Pi, x} \in A_{\text{yes}} \\
x \in \Pi_{\text{no}} &\Longrightarrow M_\Pi = \text{no} \text{ with prob } \geq 2/3 \\
&\Longrightarrow C_{M_\Pi, x} = 0 \text{ for at least } 2/3 \text{ of all random inputs } r \\
&\Longrightarrow C_{M_\Pi, x} \in A_{\text{no}}
\end{aligned}$$

A complete problem for $\mathbf{BPP}$

- There are analogous results the promise versions of $\mathbf{RP}$ and $\mathbf{ZPP}$.
- There are complete problems for the class $\mathbf{SZK}$.

Table of Contents

Failure of some structural properties

Failure of some structural properties

- In the case of language-recognition problems, if an $\mathbf{NP}$-hard problem is in $\mathbf{NP} \cap \mathbf{coNP}$ then $\mathbf{NP} = \mathbf{coNP}$

Failure of some structural properties

- In the case of language-recognition problems, if an $\mathbf{NP}$-hard problem is in $\mathbf{NP} \cap \mathbf{coNP}$ then $\mathbf{NP} = \mathbf{coNP}$
- This is not so in the case of promise problems...

Failure of some structural properties

- In the case of language-recognition problems, if an $\mathbf{NP}$-hard problem is in $\mathbf{NP} \cap \mathbf{coNP}$ then $\mathbf{NP} = \mathbf{coNP}$
- This is not so in the case of promise problems...

Definition

The problem xSAT *is the promise problem for which*

- *YES-instances are the pairs* (ϕ_1, ϕ_2) *such that* $\phi_1 \in \mathrm{SAT}$ *and* $\phi_2 \notin \mathrm{SAT}$
- *NO-instances are the pairs* (ϕ_1, ϕ_2) *such that* $\phi_1 \notin \mathrm{SAT}$ *and* $\phi_2 \in \mathrm{SAT}$

Failure of some structural properties

- In the case of language-recognition problems, if an $\mathbf{NP}$-hard problem is in $\mathbf{NP} \cap \mathbf{coNP}$ then $\mathbf{NP} = \mathbf{coNP}$
- This is not so in the case of promise problems...

Definition

The problem xSAT *is the promise problem for which*

- *YES-instances are the pairs* (ϕ_1, ϕ_2) *such that* $\phi_1 \in \mathrm{SAT}$ *and* $\phi_2 \notin \mathrm{SAT}$
- *NO-instances are the pairs* (ϕ_1, ϕ_2) *such that* $\phi_1 \notin \mathrm{SAT}$ *and* $\phi_2 \in \mathrm{SAT}$

Theorem ([3])

Any problem in $\mathbf{NP}$ *is Cook-reducible to* xSAT, *which is in* $\mathbf{NP} \cap \mathbf{coNP}$.

Failure of some structural properties

Q What has happened?

Failure of some structural properties

Q What has happened?

A A Cook-reduction to a Promise problem does not maintain the standard meaning of the concept when the promise is not tractable.

Failure of some structural properties

Q What has happened?

A A Cook-reduction to a Promise problem does not maintain the standard meaning of the concept when the promise is not tractable.

Definition (Smart reduction)

A smart reduction is a reduction that does not make queries that violate the promise.

Failure of some structural properties

Q What has happened?

A A Cook-reduction to a Promise problem does not maintain the standard meaning of the concept when the promise is not tractable.

Definition (Smart reduction)

A smart reduction is a reduction that does not make queries that violate the promise.

Theorem

If the Promise problem Π' *is reducible via smart reduction to the Promise problem* Π *and* $\Pi \in \mathbf{NP} \cap \mathbf{coNP}$ *then* $\Pi' \in \mathbf{NP} \cap \mathbf{coNP}$.

References I

Oded Goldreich.
On promise problems.
Electronic Colloquium on Computational Complexity, 18, 2005.

Johan Hastad.
Some optimal inapproximability results.
Journal of ACM, 48:798–859, 2001.

A.L Selman S. Even and Y.Yacobi.
The complexity of promise problems with applications to public-key cryptography.
Inform. and Control, 61:159–173, 1984.

Larry Stockmeyer.
On approximation algorithms for $\#\mathrm{P}$.
SIAM Journal on Computing, 14(4), 1985.

References II

L.G. Valiant and V.V. Vazirani.
NP is as easy as detecting unique solutions.
Theoretical Computer Science, 47(1):85–93, 1986.

Thank you!