THE USE OF NP-COMPLETE PROBLEMS FROM GRAPH THEORY IN PROTOCOLS FOR ZERO-KNOWLEDGE PROOFS

Robert Jajcay

April 25, 2006

Outline

General Setting:

We assume a large network of interconnected users with no *hardware safety*

General Setting:

We assume a large network of interconnected users with no *hardware safety*
that is,

- conversations may be listened to
- everyone is assumed to be dishonest until proved otherwise
- noone's identity can be taken for sure

For any conversation to take place, a **proof of identity** is required: a strictly defined *challenge and response protocol* designed to convince the recipient that the other person is who she claims to be.

Typical "real-life" situations:

- calling over the phone to activate a new credit card
- receiving an e-mail from a supposedly unbiased friend

Required Features:

- a possibility to verify one's identity beyond "reasonable" doubt

Required Features:

- a possibility to verify one's identity beyond "reasonable" doubt
- a possibility to verify the identity for any pair of users in either direction

Required Features:

- a possibility to verify one's identity beyond "reasonable" doubt
- a possibility to verify the identity for any pair of users in either direction
- a need for a large number of verifications done in real time = quickly

Required Features:

- a possibility to verify one's identity beyond "reasonable" doubt
- a possibility to verify the identity for any pair of users in either direction
- a need for a large number of verifications done in real time = quickly
- safety against pretenders even after many years of use, and a substantial number of exchanges!

Zero-Knowledge Proof Protocols

A zero-knowledge proof protocol is a challenge/response protocol allowing for identity verification that provides the verifier with no additional information allowing her to pretend the identity of the prover.

Zero-Knowledge Proof Protocols

A zero-knowledge proof protocol is a challenge/response protocol allowing for identity verification that provides the verifier with no additional information allowing her to pretend the identity of the prover.

There is no such a scheme known that would provide for a 100% assured verification.

Zero-Knowledge Proof Protocols

A zero-knowledge proof protocol is a challenge/response protocol allowing for identity verification that provides the verifier with no additional information allowing her to pretend the identity of the prover.

There is no such a scheme known that would provide for a 100% assured verification.
All the known protocols allow for verifying the identity of the prover beyond any reasonable doubt.

Decision Problems: A *class* of problems with a yes/no answer.

An NP-Problem: A class of decision problems that can be solved in polynomial non-deterministic time, and for which there exists a deterministic polynomial *certificate* for the instances where the answer is yes.

Decision Problems: A *class* of problems with a yes/no answer.

An NP-Problem: A class of decision problems that can be solved in polynomial non-deterministic time, and for which there exists a deterministic polynomial *certificate* for the instances where the answer is yes.

Example: Given a formula in a conjunctive form in n logical variables, is there a specific choice of values for the variables that will make the formula true?
Note:

- non-deterministic polynomial algorithm
- certificate is any choice of variables making the formula true
- there is no certificate for the formula to be false
- it is an *NP-complete problem*

Theorem (Goldreich, Micali, Wigderson, 1991)

Under the assumption that secure encryption functions exist, all NP problems give rise to zero-knowledge proofs.

NP-Complete Problems in Graph Theory

There is a large number of graph-theoretical problems that are known to be in the NP-complete class.

NP-Complete Problems in Graph Theory

There is a large number of graph-theoretical problems that are known to be in the NP-complete class. The *size of the instance* is usually considered as the number v of vertices of the graph.

NP-Complete Problems in Graph Theory

There is a large number of graph-theoretical problems that are known to be in the NP-complete class. The *size of the instance* is usually considered as the number v of vertices of the graph. Intuitively, NP-complete graph-theoretical problems are "global" questions with the certificate usually involving most of the vertices of the graph.

Some of the better known NP-complete problems in graph theory include:

- **Clique**: given a graph G and a positive integer k, is there a set of k vertices in G that are all mutually adjacent?

Some of the better known NP-complete problems in graph theory include:

- **Clique**: given a graph G and a positive integer k, is there a set of k vertices in G that are all mutually adjacent?
- **Independent Set**: given a graph G and a positive integer k, is there a set of k vertices in G such that no two of its vertices are adjacent?

Some of the better known NP-complete problems in graph theory include:

- **Clique**: given a graph G and a positive integer k, is there a set of k vertices in G that are all mutually adjacent?
- **Independent Set**: given a graph G and a positive integer k, is there a set of k vertices in G such that no two of its vertices are adjacent?
- **Vertex Cover**: given a graph G and a positive integer k, is there a set of k vertices in G such that each edge of G is incident with at least one of the vertices in the set?

Some of the better known NP-complete problems in graph theory include:

- **Clique**: given a graph G and a positive integer k, is there a set of k vertices in G that are all mutually adjacent?
- **Independent Set**: given a graph G and a positive integer k, is there a set of k vertices in G such that no two of its vertices are adjacent?
- **Vertex Cover**: given a graph G and a positive integer k, is there a set of k vertices in G such that each edge of G is incident with at least one of the vertices in the set?
- **3-Colorability**: given a graph G is there a coloring of the vertex set of G with three colors such that each edge is incident with vertices of distinct colors?

Some of the better known NP-complete problems in graph theory include:

- **Clique**: given a graph G and a positive integer k, is there a set of k vertices in G that are all mutually adjacent?
- **Independent Set**: given a graph G and a positive integer k, is there a set of k vertices in G such that no two of its vertices are adjacent?
- **Vertex Cover**: given a graph G and a positive integer k, is there a set of k vertices in G such that each edge of G is incident with at least one of the vertices in the set?
- **3-Colorability**: given a graph G is there a coloring of the vertex set of G with three colors such that each edge is incident with vertices of distinct colors?
- **Hamiltonicity**: given a graph G, is there a cycle in G of length $v = |V(G)|$?

Example of a Zero-Knowledge Protocol; First Approximation

The Provider assigns each user a graph that satisfies the condition, *an instance with a yes answer*, together with a certificate.

Example of a Zero-Knowledge Protocol; First Approximation

The Provider assigns each user a graph that satisfies the condition, *an instance with a yes answer*, together with a certificate.
Each user keeps the certificate away from all the other users while the assigned graphs are all published in a public “phonebook”.

Example of a Zero-Knowledge Protocol; First Approximation

The Provider assigns each user a graph that satisfies the condition, *an instance with a yes answer*, together with a certificate.

Each user keeps the certificate away from all the other users while the assigned graphs are all published in a public "phonebook".

When identification is required, the prover convinces the verifier of actually possessing the certificate without giving the certificate away!

Example Using Hamilton Cycles

Peggy and Victor both know the graph G assigned to Peggy. In addition, Peggy knows a Hamilton cycle for G.

Example Using Hamilton Cycles

- In order to verify her identity, Peggy chooses a *random* permutation p of $V(G)$, and generates a “new” graph $H = p(G)$ that she sends to Victor.

Example Using Hamilton Cycles

- In order to verify her identity, Peggy chooses a *random* permutation p of $V(G)$, and generates a "new" graph $H = p(G)$ that she sends to Victor.
- Victor asks (randomly) Peggy to either
  - prove to him that H is isomorphic to G
  - show him a Hamiltonian path for H

Example Using Hamilton Cycles

- In order to verify her identity, Peggy chooses a *random* permutation p of $V(G)$, and generates a "new" graph $H = p(G)$ that she sends to Victor.
- Victor asks (randomly) Peggy to either
  - prove to him that H is isomorphic to G
  - show him a Hamiltonian path for H
- Peggy (in response to Victor's request) either
  - reveals the isomorphism, but not the hamiltonian cycle,
  - reveals the Hamiltonian path, but not the isomorphism.

Example Using Hamilton Cycles

- In order to verify her identity, Peggy chooses a *random* permutation p of $V(G)$, and generates a "new" graph $H = p(G)$ that she sends to Victor.
- Victor asks (randomly) Peggy to either
  - prove to him that H is isomorphic to G
  - show him a Hamiltonian path for H
- Peggy (in response to Victor's request) either
  - reveals the isomorphism, but not the hamiltonian cycle,
  - reveals the Hamiltonian path, but not the isomorphism.
- These steps are repeated until Victor is satisfied

Example Using Hamilton Cycles

If Peggy does not know the Hamilton cycle, she will fail the request for providing the Hamilton cycle (which Victor can check in polynomial time).

Example Using Hamilton Cycles

If Peggy does not know the Hamilton cycle, she will fail the request for providing the Hamilton cycle (which Victor can check in polynomial time).

Peggy may try to fool Victor by sending him a graph H for which she actually knows a Hamilton cycle, but then she will fail to provide the isomorphism if asked for that.

Example Using Hamilton Cycles

If Peggy does not know the Hamilton cycle, she will fail the request for providing the Hamilton cycle (which Victor can check in polynomial time).

Peggy may try to fool Victor by sending him a graph H for which she actually knows a Hamilton cycle, but then she will fail to provide the isomorphism if asked for that.

With each round the probability that Peggy can fool Victor decreases by one half.

Example Using Hamilton Cycles

If Peggy does not know the Hamilton cycle, she will fail the request for providing the Hamilton cycle (which Victor can check in polynomial time).

Peggy may try to fool Victor by sending him a graph H for which she actually knows a Hamilton cycle, but then she will fail to provide the isomorphism if asked for that.

With each round the probability that Peggy can fool Victor decreases by one half.

Victor never receives the Hamilton cycle for the original graph G.

Practical Consequences for Specific Graph Classes

Note: the zero-knowledge proof protocol requires for the verifier not to learn anything beyond the information computable without the challenge/response exchange.

Practical Consequences for Specific Graph Classes

Note: the zero-knowledge proof protocol requires for the verifier not to learn anything beyond the information computable without the challenge/response exchange.
It is not clear however, how "safe" (i.e., computationally hard) is finding the certificate for any realistic "random" graph.

Practical Consequences for Specific Graph Classes

The provider must be able to produce a large pool of graphs with at least one certificate

Practical Consequences for Specific Graph Classes

The provider must be able to produce a large pool of graphs with at least one certificate

$$\Rightarrow$$

The provider cannot use random graphs, as that would require testing the graphs for certificates - the very thing that is computationally infeasible

Practical Consequences for Specific Graph Classes

Even if randomness is somehow achieved (say, a random set of edges is added to a cycle to obtain a Hamiltonian graph with a certificate)

Practical Consequences for Specific Graph Classes

Even if randomness is somehow achieved (say, a random set of edges is added to a cycle to obtain a Hamiltonian graph with a certificate)

NP-completness does not imply that the decision is hard for any random graph

In particular, it does not imply that the problem is hard for graphs constructed in any way that guarantees a certificate.

Practical Consequences for Specific Graph Classes

- many known heuristical algorithms are known to perform quite well for large classes of graphs; there is no class of graphs guaranteed to be resistant to *all* potential heuristical attacks

Practical Consequences for Specific Graph Classes

- many known heuristical algorithms are known to perform quite well for large classes of graphs; there is no class of graphs guaranteed to be resistant to *all* potential heuristical attacks
- the size of the graph is not directly indicative of its complexity; hence, it is hard to evaluate the algorithms

Practical Consequences for Specific Graph Classes

- many known heuristical algorithms are known to perform quite well for large classes of graphs; there is no class of graphs guaranteed to be resistant to *all* potential heuristical attacks
- the size of the graph is not directly indicative of its complexity; hence, it is hard to evaluate the algorithms
- there is a large number of graph theoretical results that give conditions that guarantee/produce a certificate for large subclasses of graphs and that may be verified in polynomial time; the class of graphs for which the problem is hard may actually be "thin" within the class of all graphs

Sample Results Guaranteeing Certificates

Theorem (Bollobás)

Almost all regular graphs are Hamiltonian.

Note: the proof rests on the Hamiltonicity of high-degree regular graphs.

Sample Results Guaranteeing Certificates

Theorem (Bollobás)

Almost all regular graphs are Hamiltonian.

Note: the proof rests on the Hamiltonicity of high-degree regular graphs.

Theorem (Robinson, Wormald)

Almost all k-regular graphs for a fixed $k \geq 3$ are Hamiltonian.

Note: knowing the Hamiltonicity of a graph may not be helpful for finding a Hamilton cycle.

Sample Results Guaranteeing Certificates

Theorem (Bollobás)

Almost all regular graphs are Hamiltonian.

Note: the proof rests on the Hamiltonicity of high-degree regular graphs.

Theorem (Robinson, Wormald)

Almost all k-regular graphs for a fixed $k \geq 3$ are Hamiltonian.

Note: knowing the Hamiltonicity of a graph may not be helpful for finding a Hamilton cycle.
Frieze, 1988, designed an $O(n^3 \log n)$ time algorithm that finds a Hamilton cycle in a high degree regular graph with a very high probability.

Sample Results Guaranteeing Certificates

Frieze's algorithm allows for the pretender to find a certificate for "any" specific graph in real time.

$\Rightarrow$

Although nothing extra is revealed in the challenge/response sequence, there is a potential for the pretender to find a certificate for someone else's graph – but there is no way to verify the correctness of someone's certificate!.

Note: Gurevich and Shelah have improved the result to a linear algorithm.

Some Few More Related Results

A number of NP-complete problems remain NP-complete even if their domains are substantially restricted.

Some Few More Related Results

A number of NP-complete problems remain NP-complete even if their domains are substantially restricted.

Theorem (Garey, Johnson, Stockmeyer)

The vertex cover and the directed Hamiltonian path problems remain NP-complete even when restricted to planar graphs.

Some Few More Related Results

A number of NP-complete problems remain NP-complete even if their domains are substantially restricted.

Theorem (Garey, Johnson, Stockmeyer)

The vertex cover and the directed Hamiltonian path problems remain NP-complete even when restricted to planar graphs.

Theorem (Garey, Johnson, Stockmeyer)

The graph 3-colorability, vertex cover, and the directed Hamiltonian path problems remain NP-complete even when restricted to graphs of degree bounded from below by a constant.

Graph Isomorphism Problem

Note: many of the zero-knowledge proof protocols rely on this problem.
An instance for this problem consists of two graphs G_1 and G_2 with a decision to be made whether the two graphs are isomorphic.

Definition

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two finite graphs. An *isomorphism* from G_1 to G_2 is a bijection $\varphi : V(G_1) \longrightarrow V(G_2)$ preserving the adjacency of vertices; $u - v$ in G_1 if and only if $\varphi(u) - \varphi(v)$ in G_2.

Graph Isomorphism Problem

Notes:

- there is an obvious non-deteministic algorithm for the graph isomorphism problem, as well as a polynomially verifiable certificate; hence, the problem belongs to the NP-hard class

Graph Isomorphism Problem

Notes:

- there is an obvious non-deteministic algorithm for the graph isomorphism problem, as well as a polynomially verifiable certificate; hence, the problem belongs to the NP-hard class
- this problem has a definite "global" character

Graph Isomorphism Problem

Notes:

- there is an obvious non-deteministic algorithm for the graph isomorphism problem, as well as a polynomially verifiable certificate; hence, the problem belongs to the NP-hard class
- this problem has a definite "global" character
- no one knows of a polynomial algorithm for this problem

Graph Isomorphism Problem

Notes:

- there is an obvious non-deteministic algorithm for the graph isomorphism problem, as well as a polynomially verifiable certificate; hence, the problem belongs to the NP-hard class
- this problem has a definite "global" character
- no one knows of a polynomial algorithm for this problem
- no one has been able to prove that no polynomial algorithm for this problem exists (which would of course prove that $P \neq NP$)

Graph Isomorphism Problem

Notes:

- there is an obvious non-deteministic algorithm for the graph isomorphism problem, as well as a polynomially verifiable certificate; hence, the problem belongs to the NP-hard class
- this problem has a definite "global" character
- no one knows of a polynomial algorithm for this problem
- no one has been able to prove that no polynomial algorithm for this problem exists (which would of course prove that $P \neq NP$)
- no one has been able to show this problem to be NP-complete

Graph Isomorphism Problem

Further notes:

- there is a number of obvious heuristics for this problem based on polynomially computable invariants

Graph Isomorphism Problem

Further notes:

- there is a number of obvious heuristics for this problem based on polynomially computable invariants
- B. McKays program "Nauty" is being widely used and believed to be the fastest for isomorphism testing for graphs of seemingly surprisingly large sizes; however, one needs to be aware that two "randomly chosen" graphs are extremely likely to be dramatically different; several classes of graphs have been shown to be exponentially hard for Nauty (see, e.g., *A performance comparison of five algorithms for graph isomorphism*, by Foggia, Sansone, and Vento).

Graph Isomorphism Problem

Further notes:

- the problem has been shown to be linear for planar graphs

Graph Isomorphism Problem

Further notes:

- the problem has been shown to be linear for planar graphs
- altered or generalized versions of the isomorphism problem have been shown to be NP-complete; for example, the problem of determining whether a given graph has a fixed-point-free automorphism is NP-complete