The Setup

- Bob publishes $n = pq$, a product of two distinct primes, and a quadratic nonresidue y with Jacobi symbol 1.
- Bob keeps secret the factorization of n.
- Alice wants to send bit string $b_1 b_2 \cdots b_k$ to Bob.
- Alice encrypts the bits by choosing a random quadratic residue modulo n if b_i is 1 and a random quadratic nonresidue (with Jacobi symbol 1) otherwise.
- So a sequence of residues and nonresidues are sent.
- Knowing the factorization of n, Bob can efficiently test quadratic residuacity and thus read the message.

The Protocol for Alice

for $i = 1, 2, \ldots, k$ **do**
    Pick $r \in Z_n^*$ randomly;
    if $b_i = 1$ **then**
        Send $r^2 \bmod n$; {Jacobi symbol is 1.}
    else
        Send $r^2 y \bmod n$; {Jacobi symbol is still 1.}
    end if
end for

The Protocol for Bob

for $i = 1, 2, \ldots, k$ **do**
    Receive r;
    if $(r \,|\, p) = 1$ and $(r \,|\, q) = 1$ **then**
        $b_i := 1$;
    else
        $b_i := 0$;
    end if
end for

Semantic Security

- This encryption scheme is probabilistic.
- There are a large number of different encryptions of a given message.
- One is chosen at random by the sender to represent the message.
  – Encryption is a *one-to-many* mapping.
- This scheme is both polynomially secure and **semantically secure**.

What then do you call proof?
— Henry James (1843–1916),
The Wings of the Dove (1902)

Leibniz knew what a proof is.
Descartes did not.
— Ian Hacking (1973)

What Is a Proof?

- A proof convinces a party of a certain claim.
  - "$x^n + y^n \neq z^n$ for all $x, y, z \in \mathbb{Z}^+$ and $n > 2$."
  - "Graph G is Hamiltonian."
  - "$x^p = x \bmod p$ for prime p and $p \not| x$."
- In mathematics, a proof is a fixed sequence of theorems.
  - Think of it as a written examination.
- We will extend a proof to cover a proof *process* by which the validity of the assertion is established.
  - Recall a job interview or an oral examination.

Prover and Verifier

- There are two parties to a proof.
  - The **prover** (**Peggy**).
  - The **verifier** (**Victor**).
- Given an assertion, the prover's goal is to convince the verifier of its validity (**completeness**).
- The verifier's objective is to accept only correct assertions (**soundness**).
- The verifier usually has an easier job than the prover.
- The setup is similar to the Turing test.[a]

[a] Turing (1950).

Interactive Proof Systems

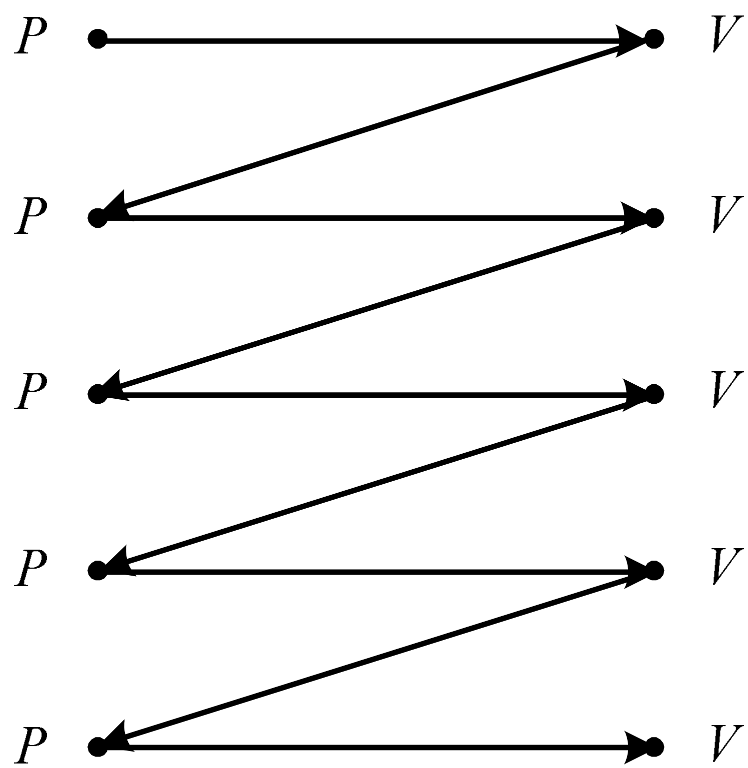
- An **interactive proof** for a language L is a sequence of questions and answers between the two parties.
- At the end of the interaction, the verifier decides whether the claim is true or false.
- The verifier must be a probabilistic polynomial-time algorithm.
- The prover runs an exponential-time algorithm.[a]
  - If the prover is not more powerful than the verifier, no interaction is needed!

[a]See the problem to Note 12.3.7 on p. 296 and Proposition 19.1 on p. 475, both of the textbook, about alternative complexity assumptions without affecting the definition. Contributed by Mr. Young-San Lin (B97902055) and Mr. Chao-Fu Yang (B97902052) on December 18, 2012.

Interactive Proof Systems (concluded)

- The system decides L if the following two conditions hold for any common input x.
  - If $x \in L$, then the probability that x is accepted by the verifier is at least $1 - 2^{-|x|}$.
  - If $x \notin L$, then the probability that x is accepted by the verifier with *any* prover replacing the original prover is at most $2^{-|x|}$.
- Neither the number of rounds nor the lengths of the messages can be more than a polynomial of $|x|$.

An Interactive Proof



IP ("Interactive Polynomial Time")[a]

- **IP** is the class of all languages decided by an interactive proof system.
- When $x \in L$, the completeness condition can be modified to require that the verifier accept with certainty without affecting IP.[b]
- Similar things cannot be said of the soundness condition when $x \notin L$.
- Verifier's coin flips can be public (called **Arthur-Merlin games**).[c]

---

[a] Goldwasser, Micali, & Rackoff (1985).
[b] Goldreich, Mansour, & Sipser (1987).
[c] Goldwasser & Sipser (1989).

The Relations of IP with Other Classes

- NP $\subseteq$ IP.
  - IP becomes NP when the verifier is deterministic and there is only one round of interaction.[a]
- BPP $\subseteq$ IP.
  - IP becomes BPP when the verifier ignores the prover's messages.
- IP = PSPACE.[b]

---

[a] Recall Proposition 41 on p. 346.
[b] Shamir (1990).

Graph Isomorphism

- $V_1 = V_2 = \{1, 2, \ldots, n\}$.
- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **isomorphic** if there exists a permutation π on $\{1, 2, \ldots, n\}$ so that $(u, v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2$.
- The task is to answer if $G_1 \cong G_2$.
- No known polynomial-time algorithms.[a]
- The problem is in NP (hence IP).
- It is not likely to be NP-complete.[b]

[a] The recent bound of Babai (2015) is $2^{O(\log^c n)}$ for some constant c.
[b] Schöning (1987).

GRAPH NONISOMORPHISM

- $V_1 = V_2 = \{ 1, 2, \ldots, n \}$.
- Graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are **nonisomorphic** if there exist *no* permutations π on $\{ 1, 2, \ldots, n \}$ so that $(u, v) \in E_1 \Leftrightarrow (\pi(u), \pi(v)) \in E_2$.
- The task is to answer if $G_1 \not\cong G_2$.
- Again, no known polynomial-time algorithms.
  - It is in coNP, but how about NP or BPP?
  - It is not likely to be coNP-complete.[a]
- Surprisingly, GRAPH NONISOMORPHISM $\in$ IP.[b]

---

[a] Schöning (1987).
[b] Goldreich, Micali, & Wigderson (1986).

A 2-Round Algorithm

Victor selects a random $i \in \{1, 2\}$;
Victor selects a random permutation π on $\{1, 2, \ldots, n\}$;
Victor applies π on graph G_i to obtain graph H;
Victor sends (G_1, H) to Peggy;
if $G_1 \cong H$ **then**
    Peggy sends $j = 1$ to Victor;
else
    Peggy sends $j = 2$ to Victor;
end if
if $j = i$ **then**
    Victor accepts; $\{G_1 \not\cong G_2.\}$
else
    Victor rejects; $\{G_1 \cong G_2.\}$
end if

Analysis

- Victor runs in probabilistic polynomial time.
- Suppose $G_1 \not\cong G_2$.
  - Peggy is able to tell which G_i is isomorphic to H, so $j = i$.
  - So Victor always accepts.
- Suppose $G_1 \cong G_2$.
  - No matter which i is picked by Victor, Peggy or any prover sees 2 *identical* copies.
  - Peggy or any prover with exponential power has only probability one half of guessing i correctly.
  - So Victor erroneously accepts with probability $1/2$.
- Repeat the algorithm to obtain the desired probabilities.

Knowledge in Proofs

- Suppose I know a satisfying assignment to a satisfiable boolean expression.
- I can convince Alice of this by giving her the assignment.
- But then I give her more knowledge than is necessary.
  - Alice can claim that she found the assignment!
  - Login authentication faces essentially the same issue.
  - See `www.wired.com/wired/archive/1.05/atm_pr.html` for a famous ATM fraud in the U.S.

Knowledge in Proofs (concluded)

- Suppose I always give Alice random bits.
- Alice extracts no knowledge from me by any measure, but I prove nothing.
- Question 1: Can we design a protocol to convince Alice (the knowledge) of a secret without revealing anything extra?
- Question 2: How to define this idea rigorously?

Zero Knowledge Proofs[a]

An interactive proof protocol (P, V) for language L has the **perfect zero-knowledge** property if:

- For every verifier V', there is an algorithm M with expected polynomial running time.
- M on any input $x \in L$ generates the same probability distribution as the one that can be observed on the communication channel of (P, V') on input x.

[a]Goldwasser, Micali, & Rackoff (1985).

Comments

- Zero knowledge is a property of the prover.
  - It is the robustness of the prover against attempts of the verifier to extract knowledge via interaction.
  - The verifier may deviate arbitrarily (but in polynomial time) from the predetermined program.
  - A verifier cannot use the transcript of the interaction to convince a third-party of the validity of the claim.
  - The proof is hence not transferable.

Comments (continued)

- Whatever a verifier can "learn" from the specified prover P via the communication channel could as well be computed from the verifier alone.
- The verifier does not learn anything except "$x \in L$."
- Zero-knowledge proofs yield no knowledge in the sense that they can be constructed by the verifier who believes the statement, and yet these proofs do convince him.

Comments (continued)

- The "paradox" is resolved by noting that it is *not* the transcript of the conversation that convinces the verifier.
- But the fact that this conversation was held "on line."
- *Computational* zero-knowledge proofs are based on complexity assumptions.
  - M only needs to generate a distribution that is computationally indistinguishable from the verifier's view of the interaction.

Comments (concluded)

- If one-way functions exist, then zero-knowledge proofs exist for every problem in NP.[a]
- If one-way functions exist, then zero-knowledge proofs exist for every problem in PSPACE.[b]
- The verifier can be restricted to the honest one (i.e., it follows the protocol).[c]
- The coins can be public.[d]
- The digital money Zcash (2016) is based on zero-knowledge proofs.

---

[a] Goldreich, Micali, & Wigderson (1986).
[b] Ostrovsky & Wigderson (1993).
[c] Vadhan (2006).
[d] Vadhan (2006).

Quadratic Residuacity (QR)

- Let n be a product of two distinct primes.
- Assume extracting the square root of a quadratic residue modulo n is hard without knowing the factors.
- QR asks if $x \in Z_n^*$ is a quadratic residues modulo n.

A Useful Corollary of Lemma 82 (p. 701)

Corollary 83 *Let $n = pq$ be a product of two distinct primes. (1) If x and y are both quadratic residues modulo n, then $xy \in Z_n^*$ is a quadratic residue modulo n. (2) If x is a quadratic residue modulo n and y is a quadratic nonresidue modulo n, then $xy \in Z_n^*$ is a quadratic nonresidue modulo n.*

- Suppose x and y are both quadratic residues modulo n.
- Let $x \equiv a^2 \bmod n$ and $y \equiv b^2 \bmod n$.
- Now xy is a quadratic residue as $xy \equiv (ab)^2 \bmod n$.

The Proof (concluded)

- Suppose x is a quadratic residue modulo n and y is a quadratic nonresidue modulo n.
- By Lemma 82 (p. 701), $(x \,|\, p) = (x \,|\, q) = 1$ but, say, $(y \,|\, p) = -1$.
- Now xy is a quadratic nonresidue as $(xy \,|\, p) = -1$, again by Lemma 82 (p. 701).

Zero-Knowledge Proof of QR[a]

Below is a zero-knowledge proof for $x \in Z_n^*$ being a quadratic residue.

1: **for** $m = 1, 2, \ldots, \log_2 n$ **do**
2:     Peggy chooses a random $v \in Z_n^*$ and sends $y = v^2 \bmod n$ to Victor;
3:     Victor chooses a random bit i and sends it to Peggy;
4:     Peggy sends $z = u^i v \bmod n$, where u is a square root of x; $\{u^2 \equiv x \bmod n.\}$
5:     Victor checks if $z^2 \equiv x^i y \bmod n$;
6: **end for**
7: Victor accepts x if Line 5 is confirmed every time;

---

[a] Goldwasser, Micali, & Rackoff (1985).

Analysis

- Suppose x is a quadratic residue.
  - Then x's square root u can be computed by Peggy.
  - Peggy can answer all challenges.
  - Now,

$$z^2 \equiv \left(u^i\right)^2 v^2 \equiv \left(u^2\right)^i v^2 \equiv x^i y \bmod n.$$

  - So Victor will accept x.

Analysis (continued)

- Suppose x is a quadratic nonresidue.
  - Corollary 83 (p. 728) says if a is a quadratic residue, then xa is a quadratic nonresidue.
  - As y is a quadratic residue, $x^i y$ can be a quadratic residue (see Line 5) only when $i = 0$.
  - Peggy can answer only one of the two possible challenges, when $i = 0$.[a]
  - So Peggy will be caught in any given round with probability one half.

---

[a]Line 5 ($z^2 \equiv x^i y \bmod n$) cannot equate a quadratic residue z^2 with a quadratic nonresidue $x^i y$ when $i = 1$.

Analysis (continued)

- How about the claim of zero knowledge?
- The transcript between Peggy and Victor when x is a quadratic residue can be generated *without* Peggy!
- Here is how.
- Suppose x is a quadratic residue.[a]
- In each round of interaction with Peggy, the transcript is a triplet (y, i, z).
- We present an efficient Bob that generates (y, i, z) with the same probability *without* accessing Peggy's power.

[a]There is no zero-knowledge requirement when $x \notin L$.

Analysis (concluded)

1: Bob chooses a random $z \in Z_n^*$;

2: Bob chooses a random bit i;

3: Bob calculates $y = z^2 x^{-i} \bmod n$;[a]

4: Bob writes (y, i, z) into the transcript;

---

[a] Recall Line 5 on p. 730: Victor checks if $z^2 \equiv x^i y \bmod n$.

Comments

- Assume x is a quadratic residue.
- For (y, i, z), y is a random quadratic residue, i is a random bit, and z is a random number.
- Bob cheats because (y, i, z) is *not* generated in the same order as in the original transcript.
  - Bob picks Peggy's answer z first.
  - Bob then picks Victor's challenge i.
  - Bob finally patches the transcript.

Comments (concluded)

- So it is not the transcript that convinces Victor, but that *conversation with Peggy is held "on line."*
- The same holds even if the transcript was generated by a cheating Victor's interaction with (honest) Peggy.
- But we skip the details.[a]
- What if Victor always chooses $i = 1$ in the protocol, the harder case?[b]

---

[a] Or apply Vadhan (2006).

[b] Contributed by Mr. Chih-Duo Hong (R95922079) on December 13, 2006, Mr. Chin-Luei Chang (D95922007) on June 16, 2008, and Mr. Han-Ting Chen (R10922073) on December 30, 2021.

Zero-Knowledge Proof of 3 Colorability[a]

for $i = 1, 2, \ldots, |E|^2$ **do**

  Peggy chooses a random permutation π of the 3-coloring ϕ;

  Peggy samples encryption schemes randomly, commits[b] them, and sends $\pi(\phi(1)), \pi(\phi(2)), \ldots, \pi(\phi(|V|))$ *encrypted* to Victor;

  Victor chooses at random an edge $e \in E$ and sends it to Peggy for the coloring of the endpoints of e;

  if $e = (u, v) \in E$ **then**

    Peggy reveals the colors $\pi(\phi(u))$ and $\pi(\phi(v))$ and "proves" that they correspond to their encryptions;

  else

    Peggy stops;

  end if

---

[a] Goldreich, Micali, & Wigderson (1986).

[b] Contributed by Mr. Ren-Shuo Liu (`D98922016`) on December 22, 2009.

10:     **if** the “proof” provided in Line 6 is not valid **then**

11:         Victor rejects and stops;

12:     **end if**

13:     **if** $\pi(\phi(u)) = \pi(\phi(v))$ or $\pi(\phi(u)), \pi(\phi(v)) \notin \{ 1, 2, 3 \}$ **then**

14:         Victor rejects and stops;

15:     **end if**

16: **end for**

17: Victor accepts;

Analysis

- If the graph is 3-colorable and both Peggy and Victor follow the protocol, then Victor always accepts.
- Suppose the graph is not 3-colorable and Victor follows the protocol.
- Let e be an edge that is *not* colored legally.
- Victor will pick it with probability $1/m$ per round, where $m = |\, E \,|$.
- Then however Peggy plays, Victor will reject with probability at least $1/m$ per round.

Analysis (concluded)

- So Victor will accept with probability at most

$$\left(1 - m^{-1}\right)^{m^2} \leq e^{-m}.$$

- Thus the protocol is a valid IP protocol.
- This protocol yields no knowledge to Victor as all he gets is a bunch of random pairs.
- The proof that the protocol is zero-knowledge to *any* verifier is intricate.[a]

[a] Or simply cite Vadhan (2006).

Comments

- Each $\pi(\phi(i))$ is encrypted by a different cryptosystem in Line 3.[a]
  - Otherwise, the coloring will be revealed in Line 6.
- Each edge e must be picked randomly.[b]
  - Otherwise, Peggy will know Victor's game plan and plot accordingly.

[a] Contributed by Ms. Yui-Huei Chang (R96922060) on May 22, 2008

[b] Contributed by Mr. Chang-Rong Hung (R96922028) on May 22, 2008

Approximability

All science is dominated by
the idea of approximation.
— Bertrand Russell (1872–1970)

Just because the problem is NP-complete
does not mean that
you should not try to solve it.
— Stephen Cook (2002)

Tackling Intractable Problems

- Many important problems are NP-complete or worse.
- **Heuristics** have been developed to attack them.
- They are **approximation algorithms**.
- How good are the approximations?
  - We are looking for theoretically *guaranteed* bounds, not "empirical" bounds.
- Are there NP problems that cannot be approximated *well* (assuming NP $\neq$ P)?
- Are there NP problems that cannot be approximated *at all* (assuming NP $\neq$ P)?

Some Definitions

- Given an **optimization problem**, each problem instance x has a set of **feasible solutions** $F(x)$.
- Each feasible solution $s \in F(x)$ has a cost $c(s) \in \mathbb{Z}^+$.
  - Here, cost refers to the quality of the feasible solution, not the time required to obtain it.
  - It is our **objective function**: total distance, number of satisfied clauses, cut size, etc.

Some Definitions (concluded)

- The **optimum cost** is

$$\textsc{opt}(x) = \min_{s \in F(x)} c(s)$$

for a minimization problem.

- It is

$$\textsc{opt}(x) = \max_{s \in F(x)} c(s)$$

for a maximization problem.

Approximation Algorithms

- Let (polynomial-time) algorithm M on x returns a feasible solution.
- M is an **ϵ-approximation algorithm**, where $\epsilon \geq 0$, if for all x,

$$\frac{|\, c(M(x)) - \text{OPT}(x) \,|}{\max(\text{OPT}(x), c(M(x)))} \leq \epsilon.$$

  - For a minimization problem,

$$\frac{c(M(x)) - \min_{s \in F(x)} c(s)}{c(M(x))} \leq \epsilon.$$

  - For a maximization problem,

$$\frac{\max_{s \in F(x)} c(s) - c(M(x))}{\max_{s \in F(x)} c(s)} \leq \epsilon. \tag{18}$$

Lower and Upper Bounds

- For a minimization problem,

$$\min_{s \in F(x)} c(s) \leq c(M(x)) \leq \frac{\min_{s \in F(x)} c(s)}{1 - \epsilon}.$$

- For a maximization problem,

$$(1 - \epsilon) \times \max_{s \in F(x)} c(s) \leq c(M(x)) \leq \max_{s \in F(x)} c(s). \tag{19}$$

Lower and Upper Bounds (concluded)

- ϵ ranges between 0 (best) and 1 (worst).
- For minimization problems, an ϵ-approximation algorithm returns solutions within

$$\left[\,\text{OPT}, \frac{\text{OPT}}{1-\epsilon}\,\right].$$

- For maximization problems, an ϵ-approximation algorithm returns solutions within

$$[\,(1-\epsilon)\times\text{OPT}, \text{OPT}\,].$$

Approximation Thresholds

- For each NP-complete optimization problem, we shall be interested in determining the *smallest* ϵ for which there is a polynomial-time ϵ-approximation algorithm.
- But sometimes ϵ has no minimum value.
- The **approximation threshold** is the greatest lower bound of all $\epsilon \geq 0$ such that there is a polynomial-time ϵ-approximation algorithm.
- By a standard theorem in real analysis, such a threshold exists.[a]

[a] Bauldry (2009).

Approximation Thresholds (concluded)

- The approximation threshold of an optimization problem is anywhere between 0 (approximation to any desired degree) and 1 (no approximation is possible).
- If P = NP, then all optimization problems *in NP* have an approximation threshold of 0.
- So assume P $\neq$ NP for the rest of the discussion.

Approximation Ratio

- ϵ-approximation algorithms can also be measured via the **approximation ratio**:[a]

$$\frac{c(M(x))}{\text{OPT}(x)}.$$

- For a minimization problem, the approximation ratio is

$$1 \leq \frac{c(M(x))}{\min_{s \in F(x)} c(s)} \leq \frac{1}{1-\epsilon}. \tag{20}$$

- For a maximization problem, the approximation ratio is

$$1-\epsilon \leq \frac{c(M(x))}{\max_{s \in F(x)} c(s)} \leq 1. \tag{21}$$

[a] Williamson & Shmoys (2011).

Approximation Ratio (concluded)

- Suppose there is an approximation algorithm that achieves an approximation ratio of θ.
  - For a minimization problem, it implies a $(1 - \theta^{-1})$-approximation algorithm by Eq. (20).
  - For a maximization problem, it implies a $(1 - \theta)$-approximation algorithm by Eq. (21).

NODE COVER

- NODE COVER seeks the smallest $C \subseteq V$ in graph $G = (V, E)$ such that for each edge in E, at least one of its endpoints is in C.
- A heuristic to obtain a good node cover is to iteratively move a node with the *highest degree* to the cover.
- This turns out to produce an approximation ratio of[a]

$$\frac{c(M(x))}{\text{OPT}(x)} = \Theta(\log n).$$

- So it is not an ϵ-approximation algorithm for any constant $\epsilon < 1$ (see p. 754).

[a]Chvátal (1979).

A 0.5-Approximation Algorithm[a]

$C := \emptyset$;
while $E \neq \emptyset$ **do**
    Delete an arbitrary edge $[\,u, v\,]$ from E;
    Add u and v to C; {Add 2 nodes to C each time.}
    Delete edges incident with u or v from E;
end while
return C;

[a] Gavril (1974).

Analysis

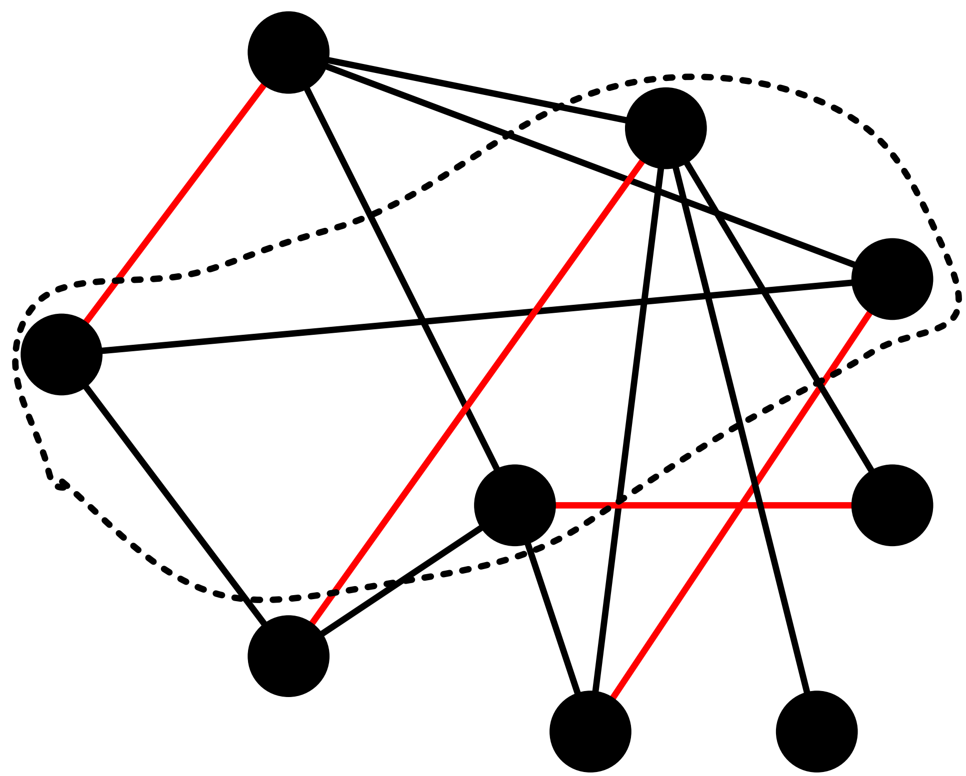
- It is easy to see that C is a node cover.
- C contains $|C|/2$ edges.[a]
- No two edges of C share a node.[b]
- *Any* node cover C' must contain at least one node from *each* of the edges of C.
  - If there is an edge in C both of whose ends are outside C', then C' will not be a cover.

---

[a] The edges deleted in Line 3.

[b] In fact, C as a set of edges is a *maximal* matching.

Analysis (continued)



Analysis (concluded)

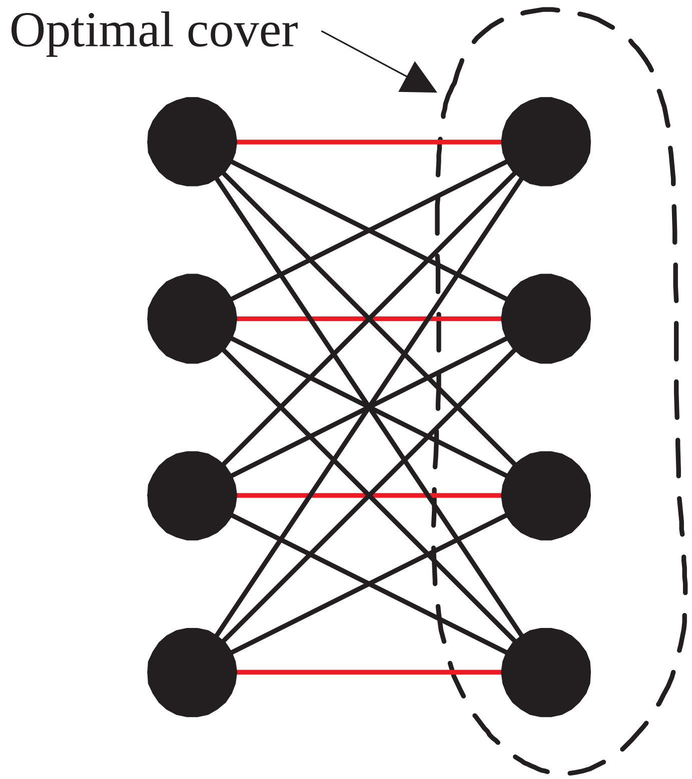
- This means that $\text{OPT}(G) \geq |\,C\,|/2$.
- The approximation ratio is hence

$$\frac{|\,C\,|}{\text{OPT}(G)} \leq 2.$$

- So we have a 0.5-approximation algorithm.[a]
- And the approximation threshold is therefore ≤ 0.5.

[a] Recall p. 754.

The 0.5 Bound Is Tight for the Algorithm[a]



[a]Contributed by Mr. Jenq-Chung Li (R92922087) on December 20, 2003. **König's theorem** says the size of a *maximum* matching equals that of a *minimum* node cover in a bipartite graph.

Remarks

- The approximation threshold is at least[a]

$$1 - \left(10\sqrt{5} - 21\right)^{-1} \approx 0.2651.$$

- The approximation threshold is 0.5 if one assumes the **unique games conjecture** (UGC).[b]
- This ratio 0.5 is also the lower bound for any "greedy" algorithms.[c]

[a] Dinur & Safra (2002).
[b] Khot & Regev (2008).
[c] Davis & Impagliazzo (2004).

Maximum Satisfiability

- Given a set of clauses, MAXSAT seeks the truth assignment that satisfies the most simultaneously.
- MAX2SAT is already NP-complete (p. 367), so MAXSAT is NP-complete.
- Consider the more general k-MAXGSAT for constant k.
  - Let $\Phi = \{ \phi_1, \phi_2, \ldots, \phi_m \}$ be a set of boolean expressions in n variables.
  - Each ϕ_i is a *general* expression involving up to k variables.
  - k-MAXGSAT seeks the truth assignment that satisfies the most expressions simultaneously.

A Probabilistic Interpretation of an Algorithm

- Let ϕ_i involve $k_i \leq k$ variables and be satisfied by s_i of the 2^{k_i} truth assignments.
- A random truth assignment $\in \{ 0, 1 \}^n$ satisfies ϕ_i with probability $p(\phi_i) = s_i / 2^{k_i}$.
  - $p(\phi_i)$ is easy to calculate as k is a constant.
- Hence a random truth assignment satisfies an average of

$$p(\Phi) = \sum_{i=1}^{m} p(\phi_i)$$

expressions ϕ_i.

The Search Procedure

- Clearly

$$p(\Phi) = \frac{p(\Phi[\,x_1 = \texttt{true}\,]) + p(\Phi[\,x_1 = \texttt{false}\,])}{2}.$$

- Select the $t_1 \in \{\,\texttt{true}, \texttt{false}\,\}$ such that $p(\Phi[\,x_1 = t_1\,])$ is the larger one.
- Note that $p(\Phi[\,x_1 = t_1\,]) \geq p(\Phi)$.
- Repeat the procedure with expression $\Phi[\,x_1 = t_1\,]$ until all variables x_i have been given truth values t_i and all ϕ_i are either true or false.

The Search Procedure (continued)

- By our hill-climbing procedure,

$$
\begin{aligned}
& p(\Phi) \\
\leq\ & p(\Phi[\, x_1 = t_1 \,]) \\
\leq\ & p(\Phi[\, x_1 = t_1, x_2 = t_2 \,]) \\
\leq\ & \cdots \\
\leq\ & p(\Phi[\, x_1 = t_1, x_2 = t_2, \ldots, x_n = t_n \,]).
\end{aligned}
$$

- So at least $p(\Phi)$ expressions are satisfied by truth assignment $(t_1, t_2, \ldots, t_n)$.

The Search Procedure (concluded)

- Note that the algorithm is *deterministic*!
- It is called **the method of conditional expectations**.[a]

[a] Erdős & Selfridge (1973); Spencer (1987).

Approximation Analysis

- The optimum is at most the number of satisfiable ϕ_is—i.e., those with $p(\phi_i) > 0$.
- The ratio of algorithm's output vs. the optimum is[a]

$$\geq \frac{p(\Phi)}{\sum_{p(\phi_i)>0} 1} = \frac{\sum_i p(\phi_i)}{\sum_{p(\phi_i)>0} 1} \geq \min_{p(\phi_i)>0} p(\phi_i).$$

- This is a polynomial-time ϵ-approximation algorithm with $\epsilon = 1 - \min_{p(\phi_i)>0} p(\phi_i)$ by Eq. (21) on p. 753.
- Because $p(\phi_i) \geq 2^{-k}$ for a satisfiable ϕ_i, the heuristic is a polynomial-time ϵ-approximation algorithm with $\epsilon = 1 - 2^{-k}$.

---

[a]Because $\sum_i a_i / \sum_i b_i \geq \min_i (a_i/b_i)$.

Back to MAXSAT

- In MAXSAT, the ϕ_i's are clauses (like $x \vee y \vee \neg z$).
- Hence $p(\phi_i) \geq 1/2$ (why?).
- The heuristic becomes a polynomial-time ϵ-approximation algorithm with $\epsilon = 1/2$.[a]
- Suppose we set each boolean variable to true with probability $(\sqrt{5} - 1)/2$, the golden ratio.
- Then follow through the method of conditional expectations to **derandomize** it.

[a] Johnson (1974).

Back to MAXSAT (concluded)

- We will obtain a $[(3-\sqrt{5})]/2$-approximation algorithm.[a]
  - Note $[(3-\sqrt{5})]/2 \approx 0.382$.
- If the clauses have k *distinct* literals,

$$p(\phi_i) = 1 - 2^{-k}.$$

- The heuristic becomes a polynomial-time ϵ-approximation algorithm with $\epsilon = 2^{-k}$.
  - This is the best possible for $k \geq 3$ unless P = NP.
- All the results hold even if clauses are weighted.

[a]Lieberherr & Specker (1981).

MAX CUT Revisited

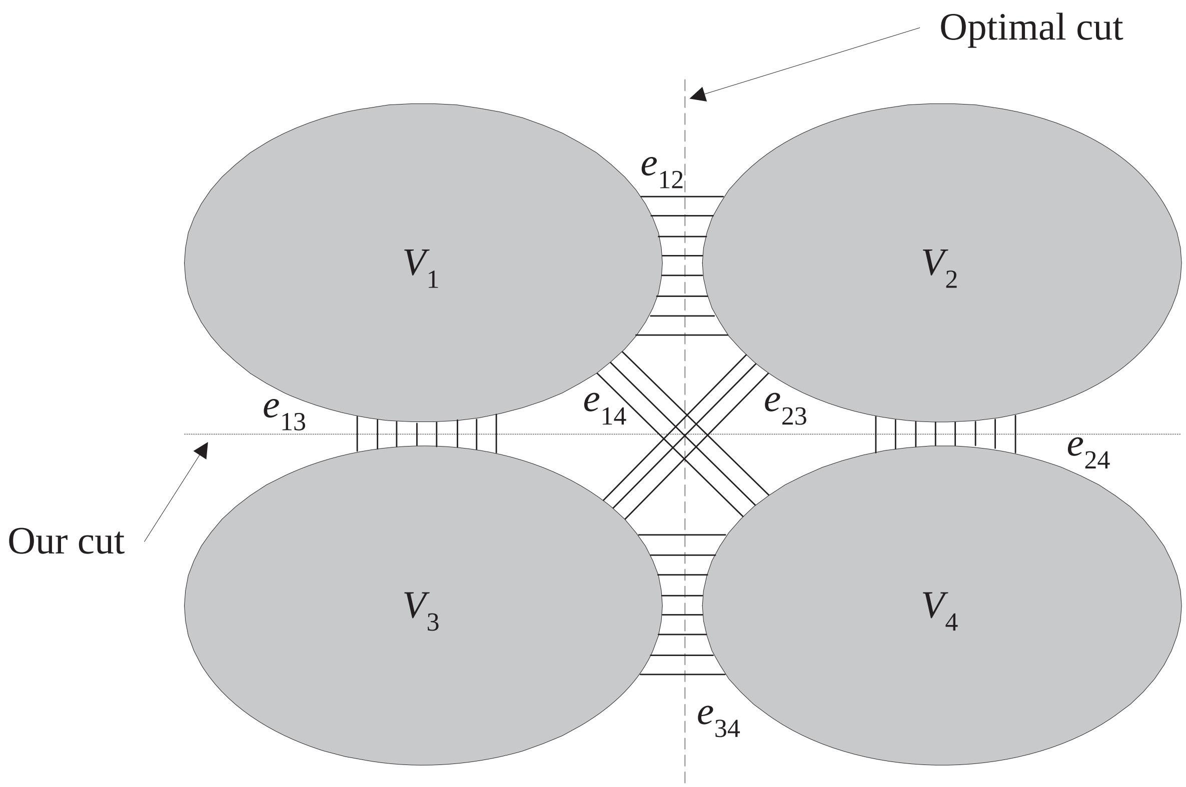
- MAX CUT seeks to partition the nodes of graph $G = (V, E)$ into $(S, V - S)$ so that there are as many edges as possible between S and $V - S$.
- It is NP-complete.[a]
- **Local search** starts from a feasible solution and performs "local" improvements until none are possible.
- Next we present a local-search algorithm for MAX CUT.

[a]Recall p. 402.

A 0.5-Approximation Algorithm for MAX CUT

```
S := ∅;
while ∃v ∈ V whose switching sides results in a larger cut do
    Switch the side of v;
end while
return S;
```

Analysis



Analysis (continued)

- Partition $V = V_1 \cup V_2 \cup V_3 \cup V_4$, where
  - Our algorithm returns $(V_1 \cup V_2, V_3 \cup V_4)$.
  - The optimum cut is $(V_1 \cup V_3, V_2 \cup V_4)$.
- Let e_{ij} be the number of edges between V_i and V_j.
- Our algorithm returns a cut of size

$$e_{13} + e_{14} + e_{23} + e_{24}.$$

- The optimum cut size is

$$e_{12} + e_{34} + e_{14} + e_{23}.$$

Analysis (continued)

- For each node $v \in V_1$, its edges to $V_3 \cup V_4$ cannot be outnumbered by those to $V_1 \cup V_2$.
  - Otherwise, v would have been moved to $V_3 \cup V_4$ to improve the cut.
- Considering all nodes in V_1 together, we have

$$2e_{11} + e_{12} \leq e_{13} + e_{14}.$$

  - $2e_{11}$, because each edge in V_1 is counted twice.
- The above inequality implies

$$e_{12} \leq e_{13} + e_{14}.$$

Analysis (concluded)

- Similarly,

$$\begin{aligned} e_{12} &\leq e_{23} + e_{24} \\ e_{34} &\leq e_{23} + e_{13} \\ e_{34} &\leq e_{14} + e_{24} \end{aligned}$$

- Add all four inequalities, divide both sides by 2, and add the inequality $e_{14} + e_{23} \leq e_{14} + e_{23} + e_{13} + e_{24}$ to obtain

$$\text{OPT} = e_{12} + e_{34} + e_{14} + e_{23} \leq 2(e_{13} + e_{14} + e_{23} + e_{24}).$$

- The above says our solution is at least half the optimum.[a]

[a] Corrected by Mr. Huan-Wen Hsiao (`B90902081`, `D08922001`) on January 14, 2021.

Remarks

- A 0.12-approximation algorithm exists.[a]
- 0.059-approximation algorithms do not exist unless NP = ZPP.[b]

---

[a] Goemans & Williamson (1995).
[b] Håstad (1997).

Approximability, Unapproximability, and Between

- Some problems have approximation thresholds less than 1.
  - KNAPSACK has a threshold of 0 (p. 792).
  - NODE COVER (p. 759), BIN PACKING, and MAXSAT[a] have a threshold larger than 0.
- The situation is maximally pessimistic for TSP (p. 778) and INDEPENDENT SET,[b] which cannot be approximated
  - Their approximation threshold is 1.

---

[a]Williamson & Shmoys (2011).

[b]See the textbook.

Unapproximability of TSP[a]

Theorem 84 *The approximation threshold of* TSP *is 1 unless $P = NP$.*

- Suppose there is a polynomial-time ϵ-approximation algorithm for TSP for some $\epsilon < 1$.
- We shall construct a polynomial-time algorithm to solve the NP-complete HAMILTONIAN CYCLE.
- Given any graph $G = (V, E)$, construct a TSP with $|V|$ cities with distances

$$d_{ij} = \begin{cases} 1, & \text{if } [\, i, j \,] \in E, \\ \frac{|V|}{1-\epsilon}, & \text{otherwise.} \end{cases}$$

[a]Sahni & Gonzales (1976).

The Proof (continued)

- Run the alleged approximation algorithm on this TSP instance.
- Note that if a tour includes edges of length $|V|/(1-\epsilon)$, then the tour costs more than $|V|$.
- Note also that no tour has a cost less than $|V|$.
- Suppose a tour of cost $|V|$ is returned.
  - Then every edge on the tour exists in the *original* graph G.
  - So this tour is a Hamiltonian cycle on G.

The Proof (concluded)

- Suppose a tour that includes an edge of length $|V|/(1-\epsilon)$ is returned.
  - The total length of this tour exceeds $|V|/(1-\epsilon)$.[a]
  - Because the algorithm is ϵ-approximate, the optimum is at least $1-\epsilon$ times the returned tour's length.
  - The optimum tour has a cost exceeding $|V|$.
  - Hence G has no Hamiltonian cycles.

[a] So this reduction is **gap introducing**.

METRIC TSP

- METRIC TSP is similar to TSP.
- But the distances must satisfy the triangular inequality:

$$d_{ij} \leq d_{ik} + d_{kj}$$

  for all i, j, k.
- Inductively,

$$d_{ij} \leq d_{ik} + d_{kl} + \cdots + d_{zj}.$$

A 0.5-Approximation Algorithm for METRIC TSP[a]

- It suffices to present an algorithm with the approximation ratio of

$$\frac{c(M(x))}{\text{OPT}(x)} \leq 2$$

(see p. 754).

[a]Choukhmane (1978); Iwainsky, Canuto, Taraszow, & Villa (1986); Kou, Markowsky, & Berman (1981); Plesník (1981).

A 0.5-Approximation Algorithm for METRIC TSP (concluded)

1: $T :=$ a minimum spanning tree of G;

2: $T' :=$ duplicate the edges of T plus their cost; {Note: T' is an Eulerian *multigraph*.}

3: $C :=$ an Euler cycle of T';

4: Remove repeated nodes of C; {"Shortcutting."}

5: **return** C;

Analysis

- Let C_{opt} be an optimal TSP tour.
- Note first that

$$c(T) \leq c(C_{\text{opt}}). \tag{22}$$

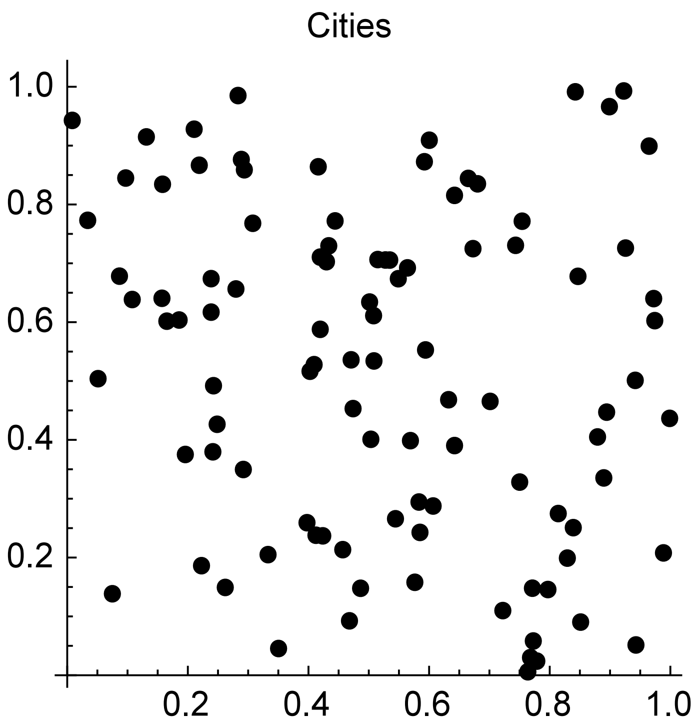
  - C_{opt} is a spanning tree after the removal of one edge.
  - But T is a *minimum* spanning tree.
- Because T' doubles the edges of T,

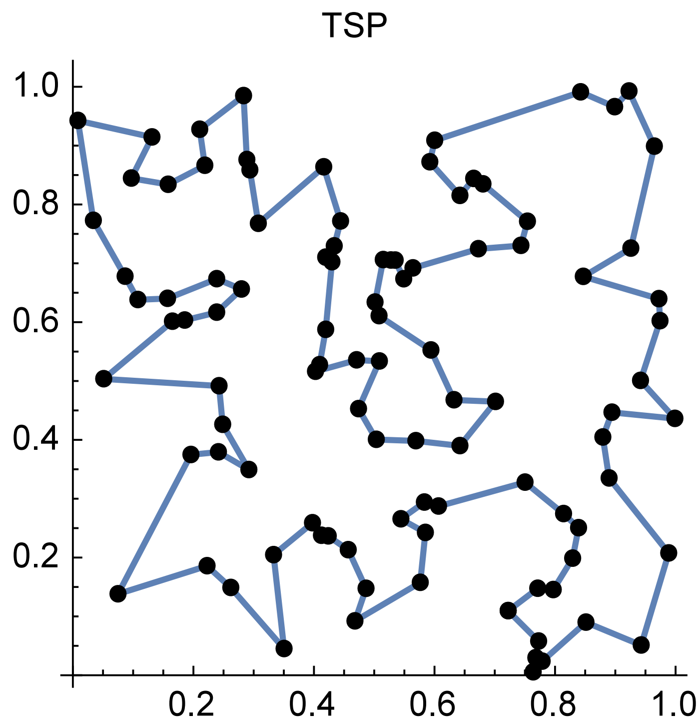
$$c(T') = 2c(T).$$

Analysis (concluded)

- Because of the triangular inequality, "shortcutting" does not increase the cost.
  - $(1, 2, 3, 2, 1, 4, \ldots) \to (1, 2, 3, 4, \ldots)$, a Hamiltonian cycle.
- Thus
$$c(C) \leq c(T').$$
- Combine all the inequalities to yield
$$c(C) \leq c(T') = 2c(T) \leq 2c(C_{\text{opt}}),$$
as desired.

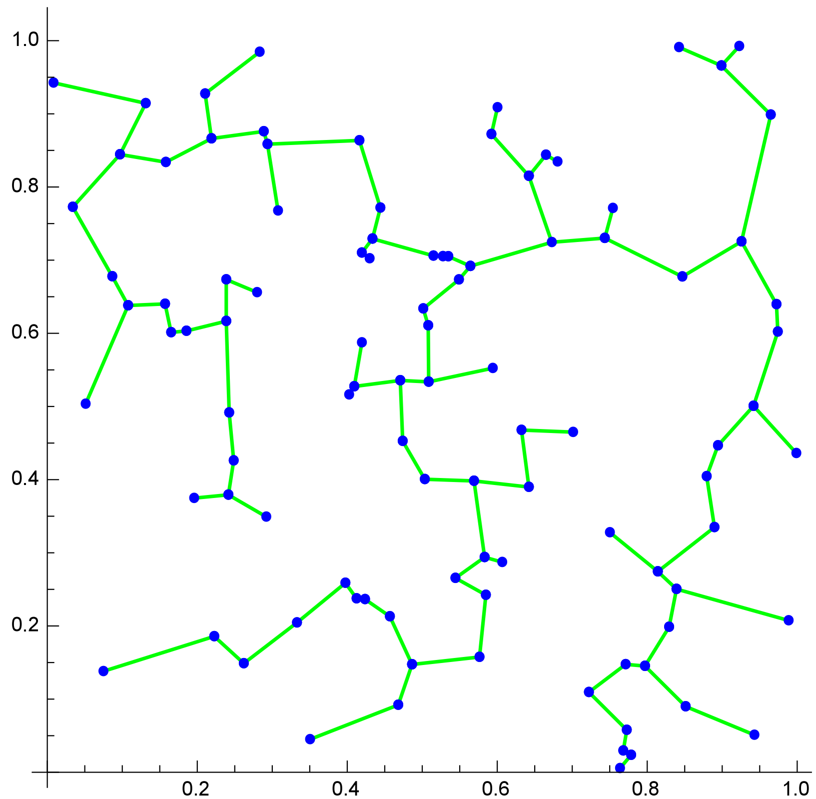
A 100-Node Example





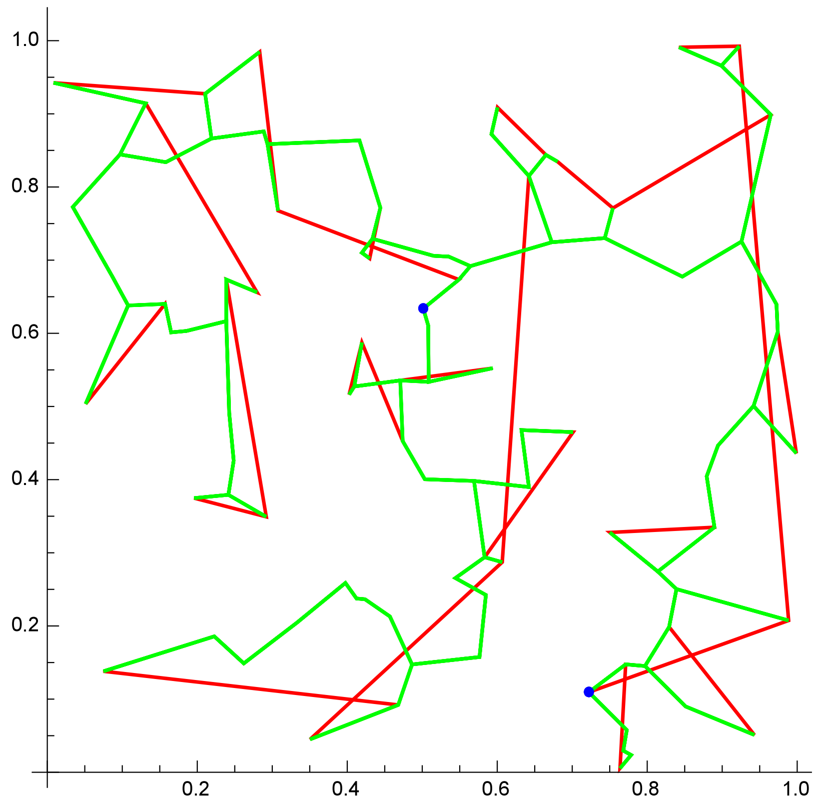
The cost is 7.72877.

A 100-Node Example (continued)



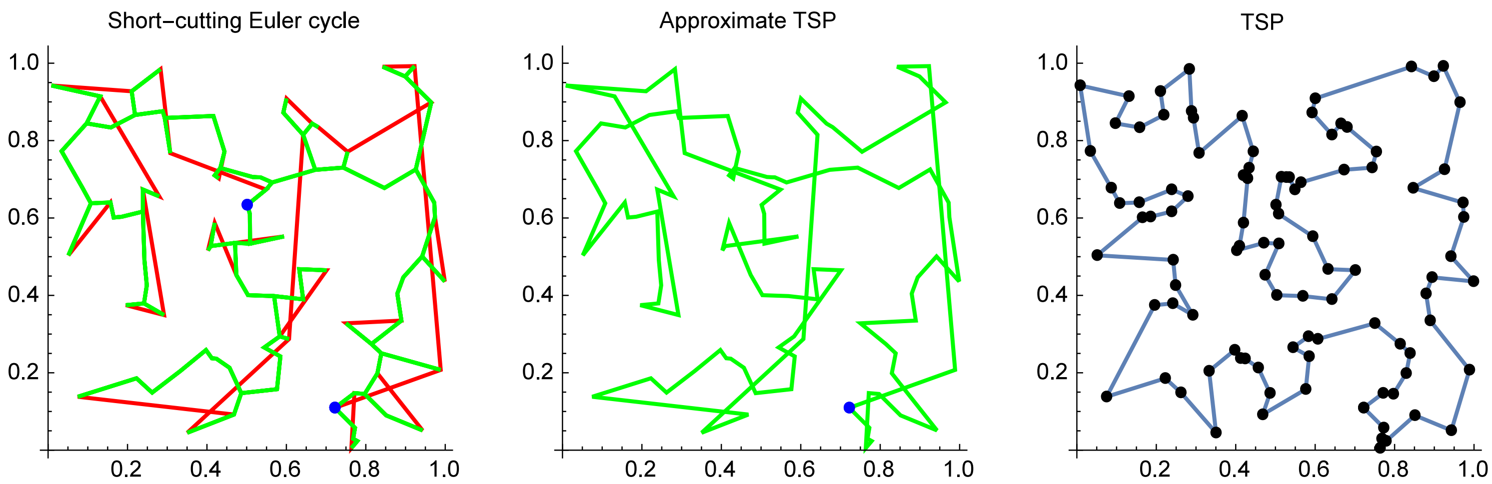
The minimum spanning tree T.

A 100-Node Example (continued)



"Shortcutting" the repeated nodes on the Euler cycle C.

A 100-Node Example (concluded)



The cost is $10.5718 \leq 2 \times 7.72877 = 15.4576$.

A (1/3)-Approximation Algorithm for METRIC TSP[a]

- It suffices to present an algorithm with the approximation ratio of

$$\frac{c(M(x))}{\text{OPT}(x)} \leq \frac{3}{2}$$

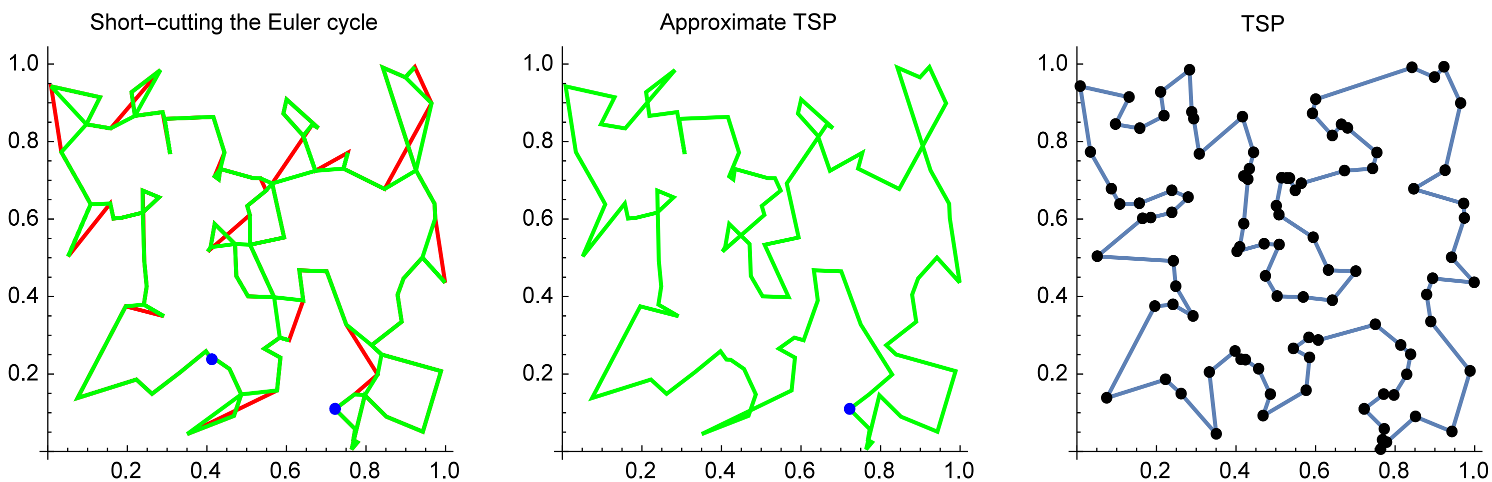
(see p. 754).

- This is the best approximation ratio for METRIC TSP as of 2016!

---

[a]Christofides (1976).

A 100-Node Example[a]



The cost is $8.74583 \leq (3/2) \times 7.72877 = 11.5932$.[b]

---

[a]Contributed by Mr. Yu-Chuan Liu (`B00507010`, `R04922040`) on July 15, 2017.

[b]In comparison, the earlier 0.5-approximation algorithm gave a cost of 10.5718 on p. 789.

KNAPSACK Has an Approximation Threshold of Zero[a]

Theorem 85 *For any ϵ, there is a polynomial-time ϵ-approximation algorithm for* KNAPSACK.

- We have n weights $w_1, w_2, \ldots, w_n \in \mathbb{Z}^+$, a weight limit W, and n values $v_1, v_2, \ldots, v_n \in \mathbb{Z}^+$.[b]
- We must find an $I \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in I} w_i \leq W$ and $\sum_{i \in I} v_i$ is the largest possible.

---

[a]Ibarra & Kim (1975). This algorithm can be used to derive good approximation algorithms for some NP-complete scheduling problems (Bansal & Sviridenko, 2006).

[b]If the values are fractional, the result is slightly messier, but the main conclusion remains correct. Contributed by Mr. Jr-Ben Tian (`B89902011`, `R93922045`) on December 29, 2004.

The Proof (continued)

- Let
$$V \stackrel{\Delta}{=} \max\{ v_1, v_2, \ldots, v_n \}.$$
- Clearly, $\sum_{i \in I} v_i \leq nV$.
- Let $0 \leq i \leq n$ and $0 \leq v \leq nV$.
- $W(i, v)$ is the minimum weight attainable by selecting only from the *first* i items[a] and with a total value of v.
  - It is an $(n+1) \times (nV+1)$ table.

---

[a] That is, items $1, 2, \ldots, i$.

The Proof (continued)

- Set $W(0, v) = \infty$ for $v \in \{ 1, 2, \ldots, nV \}$ and $W(i, 0) = 0$ for $i = 0, 1, \ldots, n$.[a]
- Then, for $0 \leq i < n$ and $1 \leq v \leq nV$,[b]

$$
\begin{aligned}
& W(i+1, v) \\
= & \begin{cases} \min\{ W(i, v), W(i, v - v_{i+1}) + w_{i+1} \}, & \text{if } v_{i+1} \leq v, \\ W(i, v), & \text{otherwise.} \end{cases}
\end{aligned}
$$

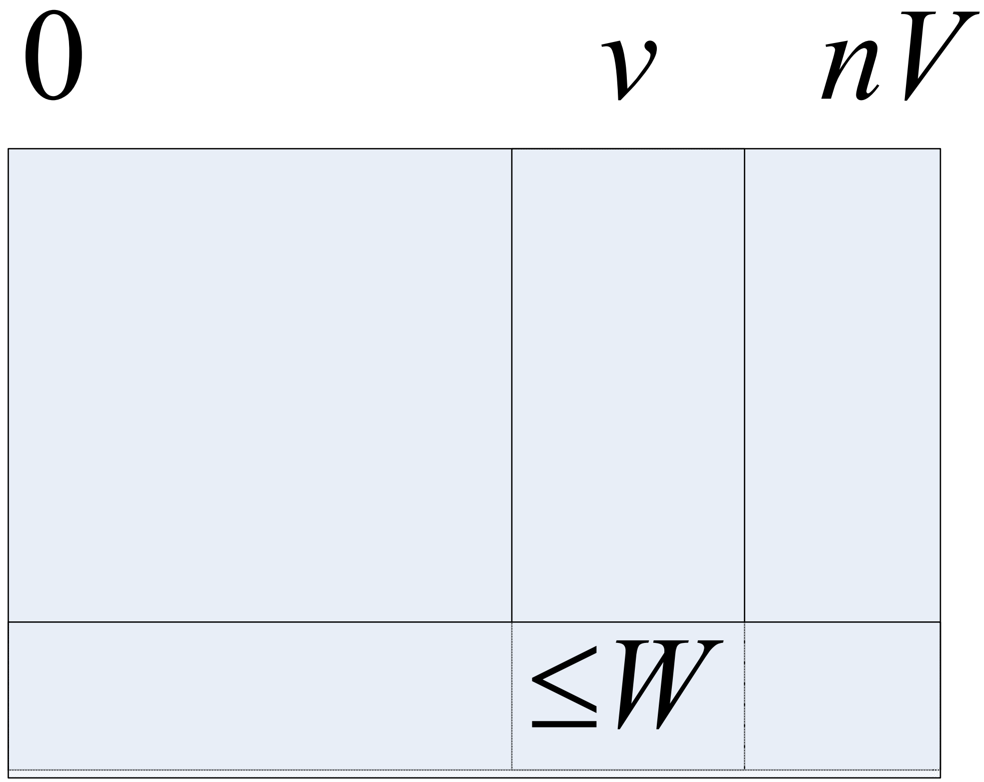
- Finally, pick the largest v such that $W(n, v) \leq W$.[c]

---

[a] Contributed by Mr. Ren-Shuo Liu (`D98922016`) and Mr. Yen-Wei Wu (`D98922013`) on December 28, 2009.

[b] The textbook's formula has an error here.

[c] Lawler (1979).


0
v
nV
≤W

The Proof (continued)

With 6 items, values $(4, 3, 3, 3, 2, 3)$, weights $(3, 3, 1, 3, 2, 1)$, and $W = 12$, the maximum total value 16 is achieved with $I = \{1, 2, 3, 4, 6\}$; I's weight is 11.

0	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
0	∞	∞	∞	3	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
0	∞	∞	3	3	∞	∞	6	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
0	∞	∞	1	3	∞	4	4	∞	∞	7	∞	∞	∞	∞	∞	∞	∞	∞
0	∞	∞	1	3	∞	4	4	∞	7	7	∞	∞	10	∞	∞	∞	∞	∞
0	∞	2	1	3	3	4	4	6	6	7	9	9	10	∞	12	∞	∞	∞
0	∞	2	1	3	3	2	4	4	5	5	7	7	8	10	10	11	∞	13

The Proof (continued)

- The running time $O(n^2V)$ is not polynomial.
- Call the problem instance

$$x = (w_1, \ldots, w_n, W, v_1, \ldots, v_n).$$

- Additional idea: Limit the number of precision bits.
- Define

$$v_i' = \left\lfloor \frac{v_i}{2^b} \right\rfloor.$$

- Note that

$$v_i - 2^b < 2^b v_i' \leq v_i. \tag{23}$$

The Proof (continued)

- Call the approximate instance

$$x' = (w_1, \ldots, w_n, W, v'_1, \ldots, v'_n).$$

- Solving x' takes time $O(n^2 V / 2^b)$.
  - Use $v'_i = \lfloor v_i / 2^b \rfloor$ and $V' = \max(v'_1, v'_2, \ldots, v'_n)$ in the dynamic programming.
  - It is now an $(n+1) \times (nV+1)/2^b$ table.
- The selection I' is optimal for x'.
- But I' may not be optimal for x, although it still satisfies the weight budget W.

The Proof (continued)

With the same parameters as p. 796 and $b = 1$: Values are $(2, 1, 1, 1, 1, 1)$ and the optimal selection $I' = \{1, 2, 3, 5, 6\}$ for x' has a *smaller* maximum value $4 + 3 + 3 + 2 + 3 = 15$ for x than I's 16; its weight is $10 < W = 12$.[a]

0	∞	∞	∞	∞	∞	∞	∞
0	∞	3	∞	∞	∞	∞	∞
0	3	3	6	∞	∞	∞	∞
0	1	3	4	7	∞	∞	∞
0	1	3	4	7	10	∞	∞
0	1	3	4	6	9	12	∞
0	1	2	4	5	7	10	13

[a] The *original* optimal $I = \{1, 2, 3, 4, 6\}$ on p. 796 has the same value 6 and but higher weight 11 for x'.

The Proof (continued)

- The value of I' for x is close to that of the optimal I as

$$
\begin{aligned}
& \sum_{i \in I'} v_i \\
\geq \ & \sum_{i \in I'} 2^b v_i' \quad \text{by inequalities (23) on p. 797} \\
= \ & 2^b \sum_{i \in I'} v_i' \geq 2^b \sum_{i \in I} v_i' = \sum_{i \in I} 2^b v_i' \\
\geq \ & \sum_{i \in I} \left(v_i - 2^b \right) \quad \text{by inequalities (23)} \\
\geq \ & \left(\sum_{i \in I} v_i \right) - n 2^b .
\end{aligned}
$$

The Proof (continued)

- In summary,

$$\sum_{i \in I'} v_i \geq \left(\sum_{i \in I} v_i \right) - n2^b.$$

- Without loss of generality, assume $w_i \leq W$ for all i.
  - Otherwise, item i is redundant and can be removed early on.
- V is a lower bound on OPT.[a]
  - Picking one single item with value V is a legitimate choice.

---

[a]Recall that $V = \max\{ v_1, v_2, \ldots, v_n \}$ (p. 793).

The Proof (concluded)

- The relative error from the optimum is:

$$\frac{\sum_{i \in I} v_i - \sum_{i \in I'} v_i}{\sum_{i \in I} v_i} \leq \frac{n 2^b}{V}.$$

- Suppose we pick $b = \lfloor \log_2 \frac{\epsilon V}{n} \rfloor$.
- The algorithm becomes ϵ-approximate.[a]
- The running time is then $O(n^2 V / 2^b) = O(n^3 / \epsilon)$, a polynomial in n and $1/\epsilon$.[b]

---

[a]See Eq. (18) on p. 748.

[b]It hence depends on the *value* of $1/\epsilon$. Thanks to a lively class discussion on December 20, 2006. If we fix ϵ and let the problem size increase, then the complexity is cubic. Contributed by Mr. Ren-Shan Luoh (`D97922014`) on December 23, 2008.

Comments

- INDEPENDENT SET and NODE COVER are reducible to each other (Corollary 46, p. 393).
- NODE COVER has an approximation threshold at most 0.5 (p. 761).
- But INDEPENDENT SET is unapproximable (see the textbook).
- INDEPENDENT SET limited to graphs with degree $\leq k$ is called k-DEGREE INDEPENDENT SET.
- k-DEGREE INDEPENDENT SET is approximable (see the textbook).