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

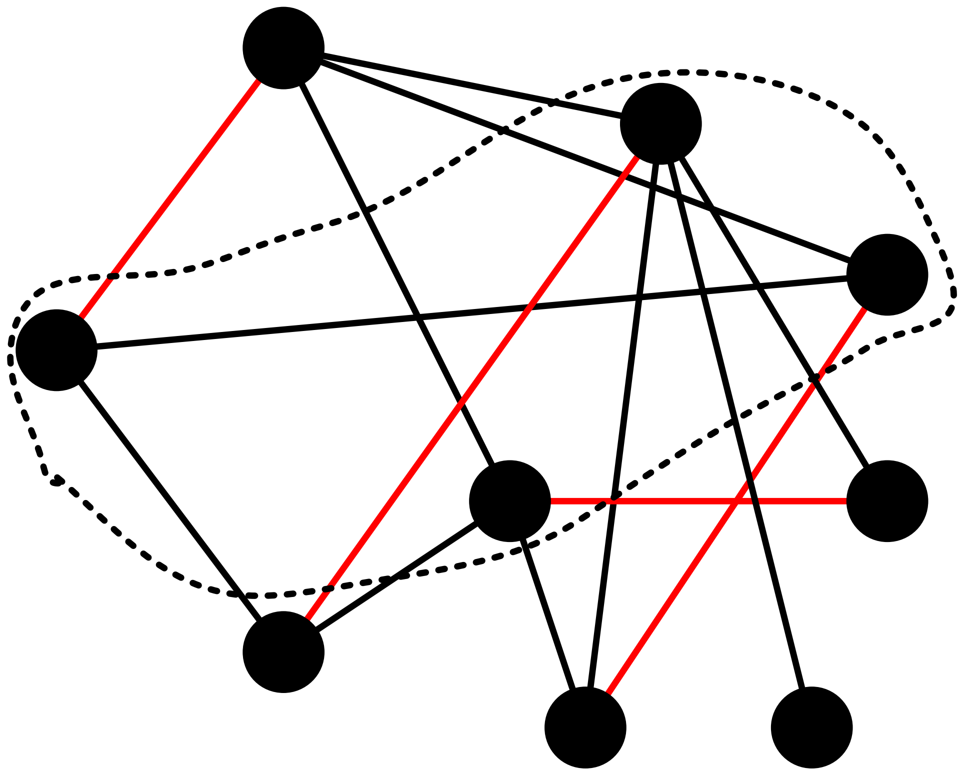
- It is easy to see that C is a node cover.
- C contains $|\,C\,|/2$ edges.[a]
- No two edges of C share a node.[b]
- *Any* node cover C' must contain at least one node from *each* of the edges of C.
  - If there is an edge in C both of whose ends are outside C', then C' is not a cover.

---

[a] The edges deleted in Line 3.

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

Analysis (continued)



Analysis (concluded)

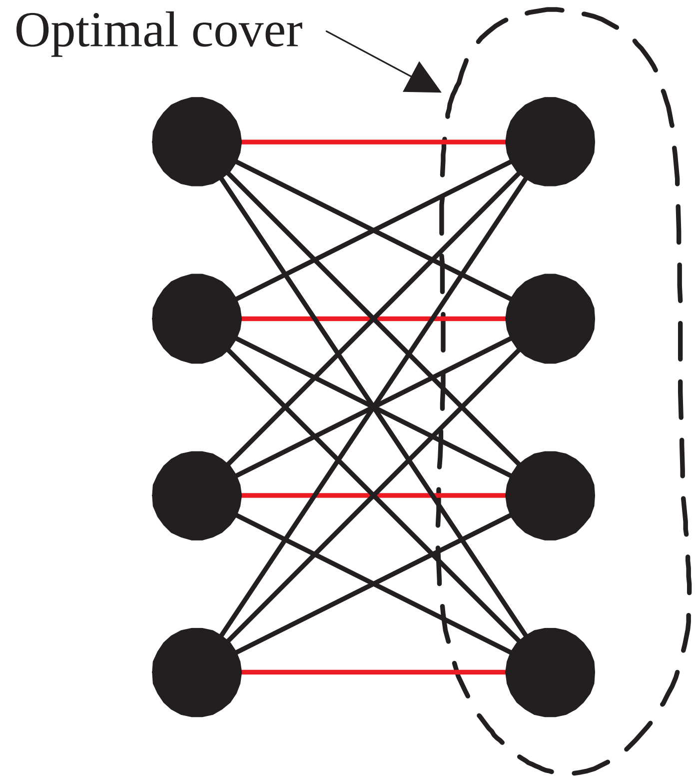
- This means that $\text{OPT}(G) \geq |\, C \,|/2$.
- The approximation ratio is hence

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

- 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. 365), 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. 754.
- 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 at least k *distinct* literals,

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

- 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.[b]
- All the results hold even if clauses are weighted.

[a] Lieberherr & Specker (1981).
[b] Håstad (2001).

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.

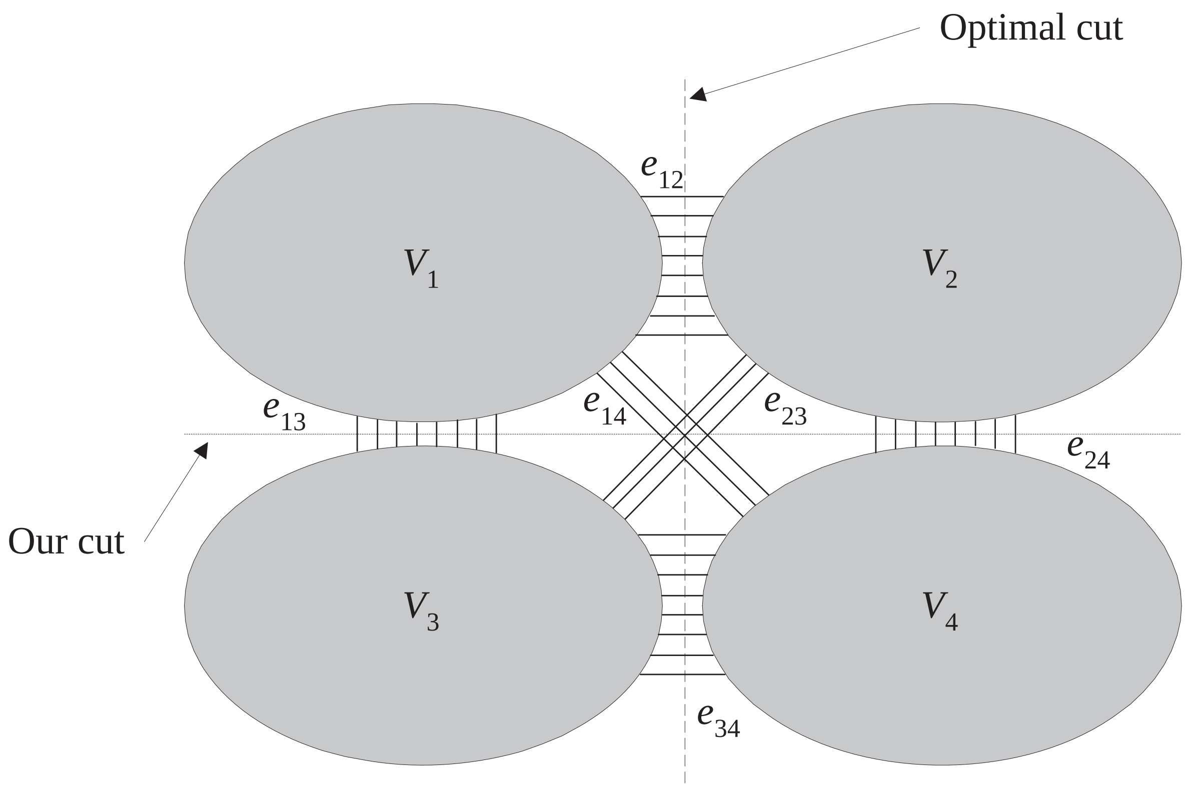
---

[a]Recall p. 400.

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;

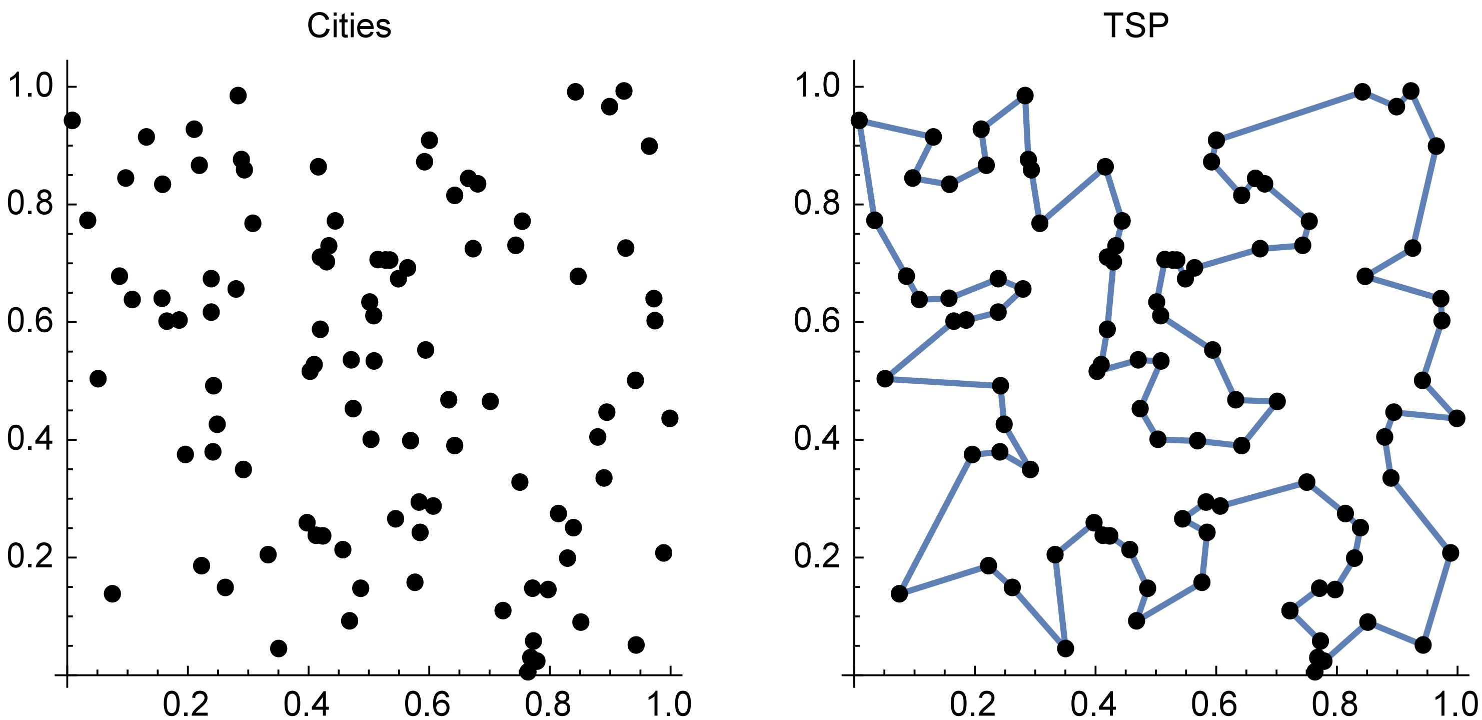
Analysis

- 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)

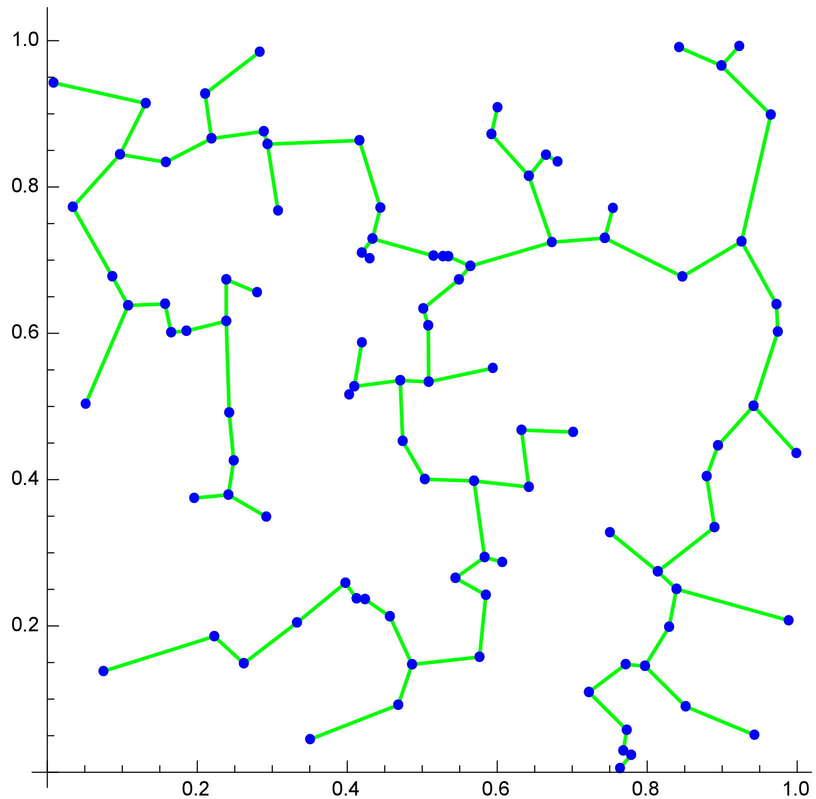
- Because of the triangular inequality, "shortcutting" does not increase the cost.
  - $(1, 2, 3, 2, 1, 4, \ldots) \to (1, 2, 3, 4, \ldots)$, a Hamiltonian cycle.
- Thus
$$c(C) \leq c(T').$$
- Combine all the inequalities to yield
$$c(C) \leq c(T') = 2c(T) \leq 2c(C_{\text{opt}}),$$
  as desired.

A 100-Node Example



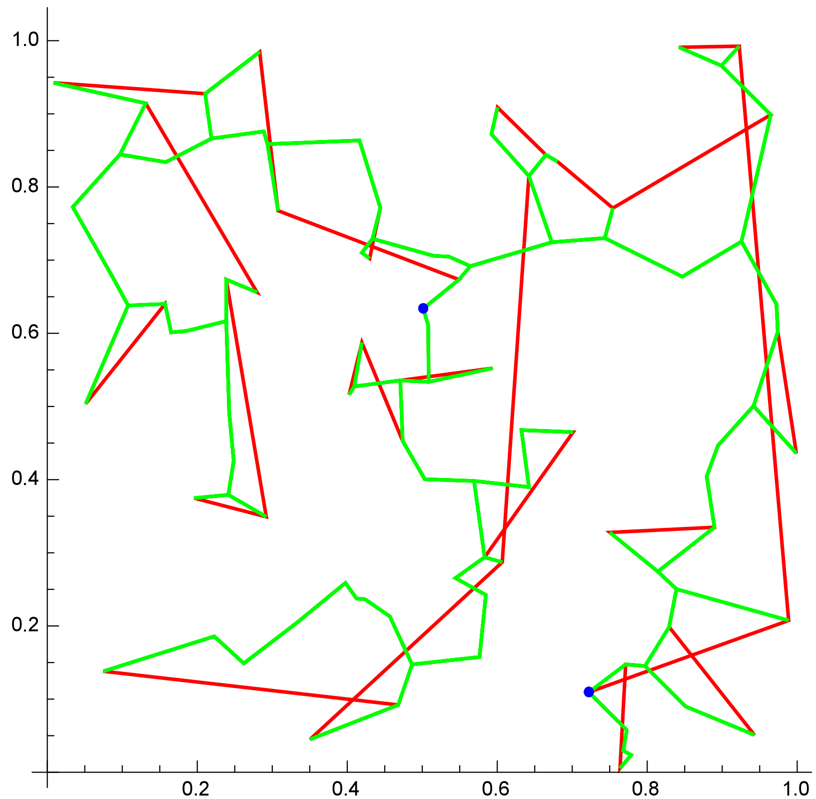
The cost is 7.72877.

A 100-Node Example (continued)



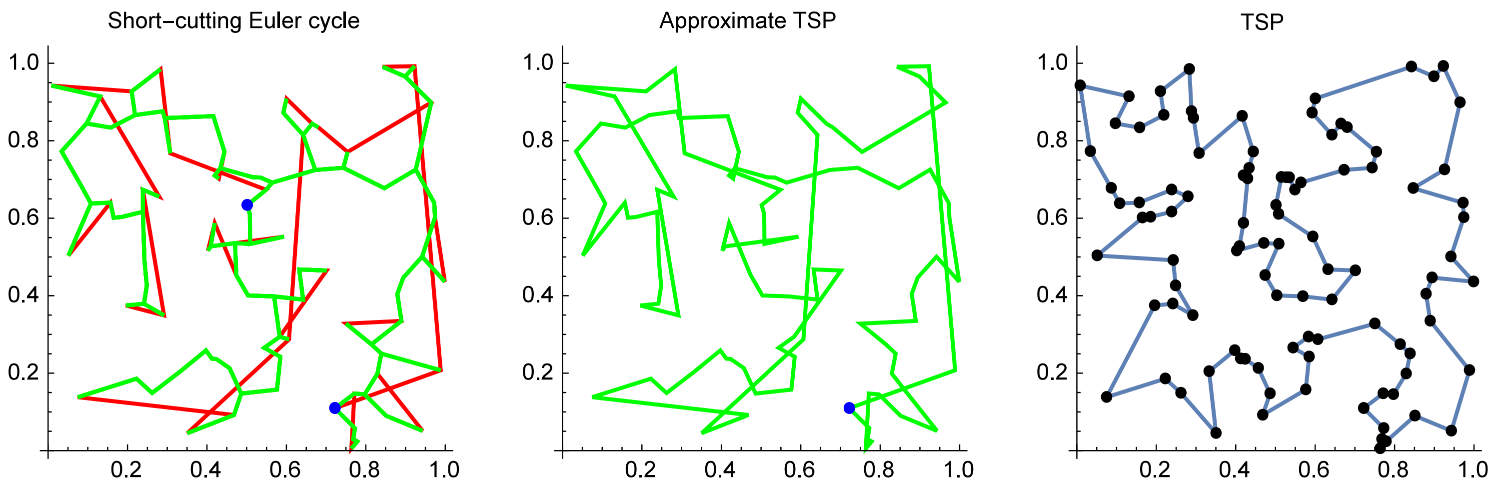
The minimum spanning tree T.

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

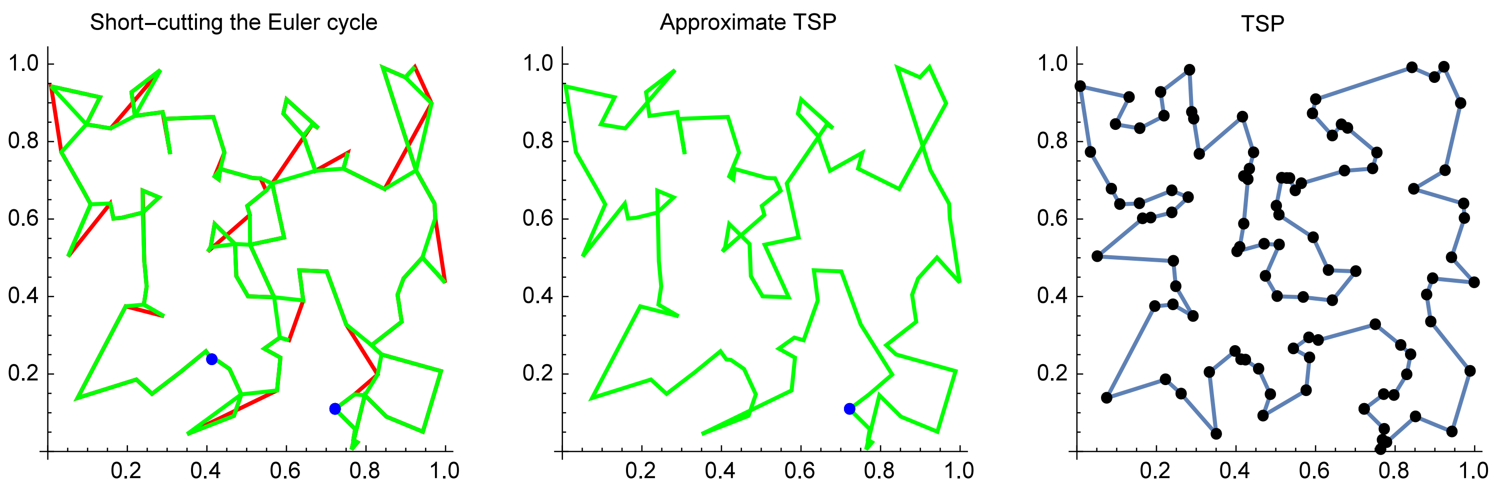
$$\frac{c(M(x))}{\text{OPT}(x)} \leq \frac{3}{2}$$

(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]

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

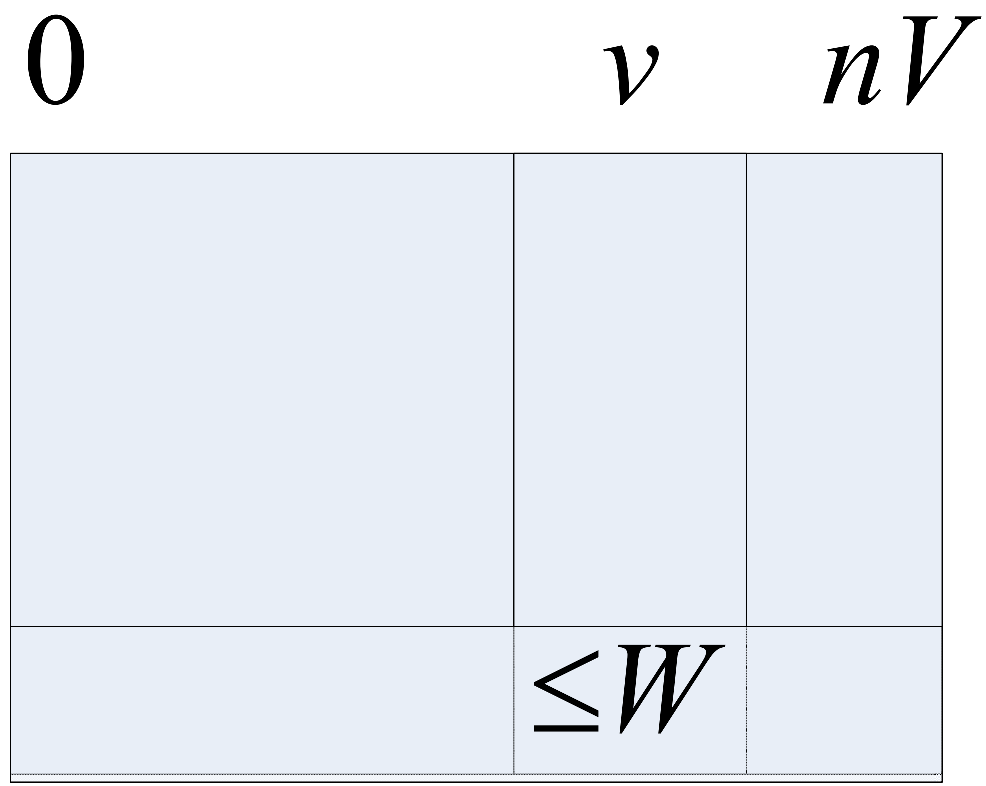
- Finally, pick the largest v such that $W(n, v) \leq W$.[c]

---

[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 $O(n^2 V)$ 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 $(n+1) \times (n\lfloor V/2^b \rfloor + 1)$ 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. 391).
- 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).

On P vs. NP

If 50 million people believe a foolish thing,
it's still a foolish thing.
— George Bernard Shaw (1856–1950)

Exponential Circuit Complexity for NP-Complete Problems

- We shall prove exponential lower bounds for NP-complete problems using *monotone* circuits.
  - Monotone circuits are circuits without $\neg$ gates.[a]
- Note that this result does *not* settle the P vs. NP problem.

---

[a] Recall p. 329.

The Power of Monotone Circuits

- Monotone circuits can only compute monotone boolean functions.
- They are powerful enough to solve a P-complete problem: MONOTONE CIRCUIT VALUE (p. 330).
- There are NP-complete problems that are not monotone; they cannot be computed by monotone circuits at all.
- There are NP-complete problems that are monotone; they can be computed by monotone circuits.
  - HAMILTONIAN PATH and CLIQUE.

$\text{CLIQUE}_{n,k}$

- $\text{CLIQUE}_{n,k}$ is the boolean function deciding whether a graph $G = (V, E)$ with n nodes has a clique of size k.
- The input gates are the $\binom{n}{2}$ entries of the adjacency matrix of G.
  - Gate g_{ij} is set to true if the associated undirected edge $\{ i, j \}$ exists.
- $\text{CLIQUE}_{n,k}$ is a monotone function.
- Thus it can be computed by a monotone circuit.
- Of course, this does not rule out that *non*monotone circuits for $\text{CLIQUE}_{n,k}$ may use *fewer* gates.

Crude Circuits

- One possible circuit for CLIQUE$_{n,k}$ does the following.
  1. For each $S \subseteq V$ with $|\,S\,| = k$, there is a circuit with $O(k^2)$ $\wedge$-gates testing whether S forms a clique.
  2. We then take an OR of the outcomes of all the $\binom{n}{k}$ subsets $S_1, S_2, \ldots, S_{\binom{n}{k}}$.
- This is a monotone circuit with $O(k^2\binom{n}{k})$ gates, which is exponentially large unless k or $n - k$ is a constant.
- A **crude circuit** $\mathrm{CC}(X_1, X_2, \ldots, X_m)$ tests if there is an $X_i \subseteq V$ that forms a clique.[a]
  - The above-mentioned circuit is $\mathrm{CC}(S_1, S_2, \ldots, S_{\binom{n}{k}})$.

[a] Consider the empty set a clique.

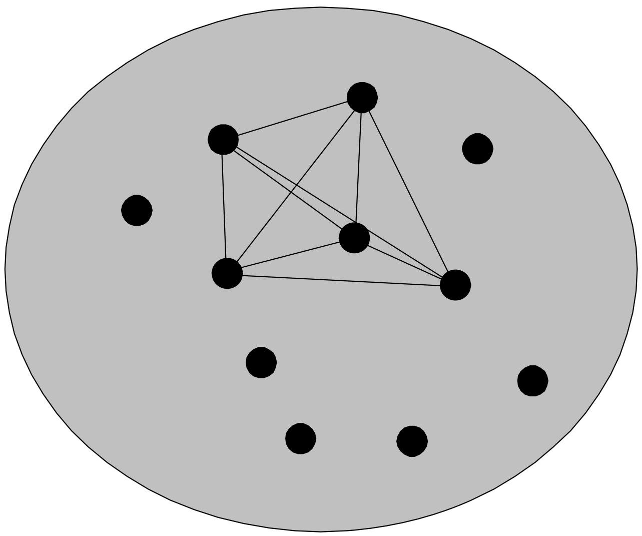
The Proof: Positive Examples

- Analysis will be applied to only the following **positive examples** and **negative examples** as input graphs.
- A positive example is a graph that has $\binom{k}{2}$ edges connecting k nodes in all possible ways.
- There are $\binom{n}{k}$ such graphs.
- $\textsc{clique}_{n,k}$ should output true on them.

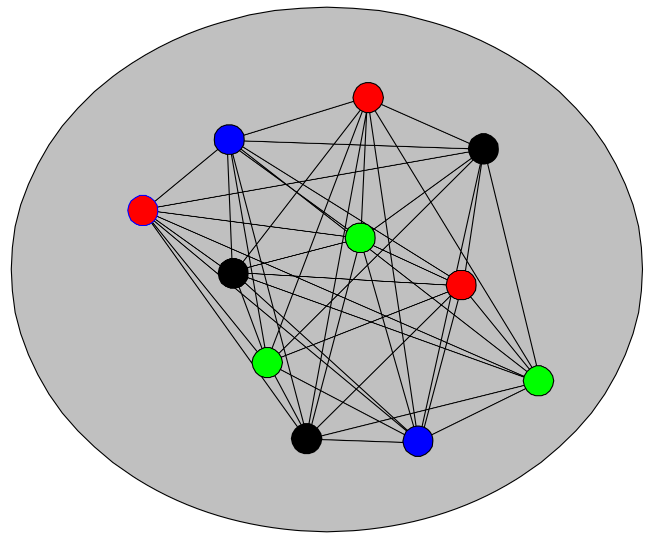
The Proof: Negative Examples

- Color the nodes with $k-1$ different colors and join by an edge any two nodes that are colored differently.
- There are $(k-1)^n$ such graphs.
- $\text{CLIQUE}_{n,k}$ should output false on them.
  - Each set of k nodes must have 2 identically colored nodes; hence there is no edge between them.

Positive and Negative Examples with $k = 5$



A positive example



A negative example