A Simple FPTAS for Counting Edge Covers

Chengyu Lin[1] Jingcheng Liu[1] Pinyan Lu[2]

[1]Shanghai Jiao Tong University

[2]Microsoft Research Asia

ACM-SIAM Symposium on Discrete Algorithms, 2014

Overview

1. Introduction
2. Our Result

Edge cover

Definition

For an undirected input graph $G = (V, E)$, an **edge cover** of G is a set of edges C covering all vertices.

Example

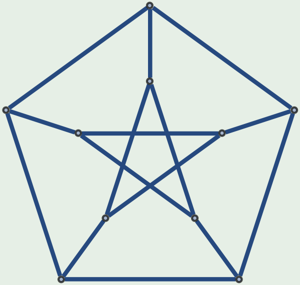


Figure: An edge cover for Petersen graph

Edge cover

Definition

For an undirected input graph $G = (V, E)$, an **edge cover** of G is a set of edges C covering all vertices.

Example

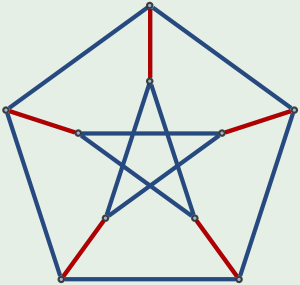


Figure: An edge cover for Petersen graph, with edges chosen being highlighted in red. Note that this is also a perfect matching.

Edge cover

Edge cover is related to many other problems such as:

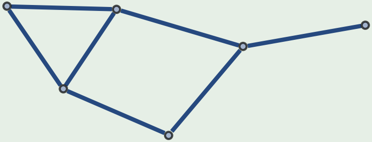
- Matching problem.
- Rtw-Mon-CNF. (read twice monotone CNF)
- Holant problem.
- . . . .

Relation to Matching

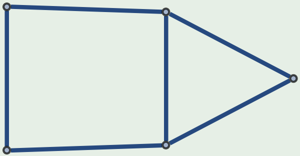
The minimal edge cover could be found via a greedy algorithm based on a maximum matching.

Example

Find edge covers by maximal matching?



(a) G has a perfect matching.



(b) G doesn't have a perfect matching.

Remark

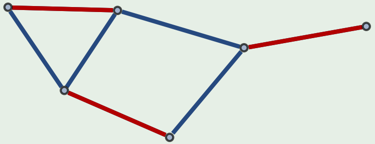
For a graph with a perfect matching, enumerating (sampling) perfect matchings is equivalent to enumerating (sampling) minimum edge covers.

Relation to Matching

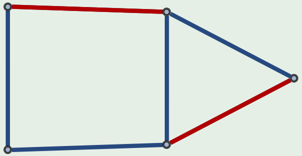
The minimal edge cover could be found via a greedy algorithm based on a maximum matching.

Example

Find edge covers by maximal matching?



(c) G has a perfect matching.



(d) G doesn't have a perfect matching.

Remark

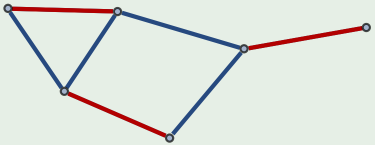
For a graph with a perfect matching, enumerating (sampling) perfect matchings is equivalent to enumerating (sampling) minimum edge covers.

Relation to Matching

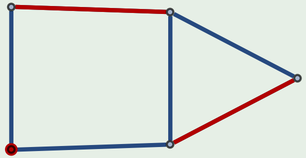
The minimal edge cover could be found via a greedy algorithm based on a maximum matching.

Example

Find edge covers by maximal matching?



(e) G has a perfect matching.



(f) G doesn't have a perfect matching.

Remark

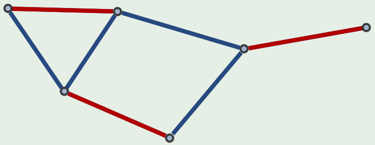
For a graph with a perfect matching, enumerating (sampling) perfect matchings is equivalent to enumerating (sampling) minimum edge covers.

Relation to Matching

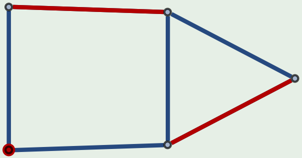
The minimal edge cover could be found via a greedy algorithm based on a maximum matching.

Example

Find edge covers by maximal matching?



(g) G has a perfect matching.



(h) G doesn't have a perfect matching.

Remark

For a graph with a perfect matching, enumerating (sampling) perfect matchings is equivalent to enumerating (sampling) minimum edge covers.

Relation to Rtw-Mon-CNF

Definition

A formula is **read twice** if every variables appears at most twice.
A formula is **monotone** if every variables appears positively.

Consider the following Rtw-Mon-CNF formula, its satisfying assignments are exactly edge covers of its graph representation, where we write edges as variables, and vertices as clauses.

$$\phi = (e_1 \vee e_2 \vee e_3) \wedge (e_1 \vee e_4) \wedge (e_4 \vee e_5 \vee e_2) \wedge (e_3 \vee e_5).$$

Figure: Graph representation for ϕ.

Relation to Rtw-Mon-CNF

Definition

A formula is **read twice** if every variables appears at most twice.
A formula is **monotone** if every variables appears positively.

Consider the following Rtw-Mon-CNF formula, its satisfying assignments are exactly edge covers of its graph representation, where we write edges as variables, and vertices as clauses.

$$\phi = (e_1 \vee e_2 \vee e_3) \wedge (e_1 \vee e_4) \wedge (e_4 \vee e_5 \vee e_2) \wedge (e_3 \vee e_5).$$

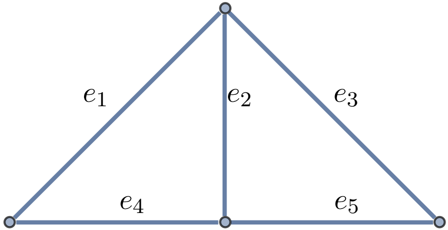


Figure: Graph representation for ϕ.

Counting Problems

A list of problems in their search, optimization, and counting versions.

Search problems:

- SAT.
- Find a (perfect) matching.
- Find an edge cover.
- . . . .

Optimizations:

- MAX-SAT.
- Find a maximum matching.
- Find a minimum edge cover.
- . . . .

Counting problems:

- #SAT.
- Counting matchings.
- Counting edge covers.
- . . . .

Counting Problems

A list of problems in their search, optimization, and counting versions.

Search problems:

- SAT.
- Find a (perfect) matching.
- Find an edge cover.
- . . . .

Optimizations:

- MAX-SAT.
- Find a maximum matching.
- Find a minimum edge cover.
- . . . .

Counting problems:

- #SAT.
- Counting matchings.
- Counting edge covers.
- . . . .

Counting Problems

A list of problems in their search, optimization, and counting versions.

Search problems:

- SAT.
- Find a (perfect) matching.
- Find an edge cover.
- ....

Optimizations:

- MAX-SAT.
- Find a maximum matching.
- Find a minimum edge cover.
- ....

Counting problems:

- #SAT.
- Counting matchings.
- Counting edge covers.
- ....

Counting Problems

A list of problems in their search, optimization, and counting versions.

Search problems:

- SAT.
- Find a (perfect) matching.
- Find an edge cover.
- . . . .

Optimizations:

- MAX-SAT.
- Find a maximum matching.
- Find a minimum edge cover.
- . . . .

Counting problems:

- #SAT.
- Counting matchings.
- Counting edge covers.
- . . . .

Counting Problems

A list of problems in their search, optimization, and counting versions.

Search problems:

- SAT.
- Find a (perfect) matching.
- Find an edge cover.
- . . . .

Optimizations:

- MAX-SAT.
- Find a maximum matching.
- Find a minimum edge cover.
- . . . .

Counting problems:

- #SAT.
- Counting matchings.
- Counting edge covers.
- . . . .

Counting Problems

A list of problems in their search, optimization, and counting versions.

Search problems:

- SAT.
- Find a (perfect) matching.
- Find an edge cover.
- ....

Optimizations:

- MAX-SAT.
- Find a maximum matching.
- Find a minimum edge cover.
- ....

Counting problems:

- #SAT.
- Counting matchings.
- Counting edge covers.
- ....

Counting Problems

A list of problems in their search, optimization, and counting versions.

Search problems:

- SAT.
- Find a (perfect) matching.
- Find an edge cover.
- . . . .

Optimizations:

- MAX-SAT.
- Find a maximum matching.
- Find a minimum edge cover.
- . . . .

Counting problems:

- #SAT.
- Counting matchings.
- Counting edge covers.
- . . . .

Counting Problems

A list of problems in their search, optimization, and counting versions.

Search problems:

- SAT.
- Find a (perfect) matching.
- Find an edge cover.
- ....

Optimizations:

- MAX-SAT.
- Find a maximum matching.
- Find a minimum edge cover.
- ....

Counting problems:

- #SAT.
- Counting matchings.
- Counting edge covers.
- ....

Counting Problems

A list of problems in their search, optimization, and counting versions.

Search problems:

- SAT.
- Find a (perfect) matching.
- Find an edge cover.
- ....

Optimizations:

- MAX-SAT.
- Find a maximum matching.
- Find a minimum edge cover.
- ....

Counting problems:

- #SAT.
- Counting matchings.
- Counting edge covers.
- ....

Why counting?

Besides theoretical computer science, counting problems are also related to many problems from other discipline such as:

- Partition function of Statistical physics.
- Graph polynomials.
- Sampling, learning and inference.
- Query evaluations of probabilistic database.
- ....

Why counting?

Besides theoretical computer science, counting problems are also related to many problems from other discipline such as:

- Partition function of Statistical physics.
- Graph polynomials.
- Sampling, learning and inference.
- Query evaluations of probabilistic database.
- . . . .

Why counting?

Besides theoretical computer science, counting problems are also related to many problems from other discipline such as:

- Partition function of Statistical physics.
- Graph polynomials.
- Sampling, learning and inference.
- Query evaluations of probabilistic database.
- ....

Why counting?

Besides theoretical computer science, counting problems are also related to many problems from other discipline such as:

- Partition function of Statistical physics.
- Graph polynomials.
- Sampling, learning and inference.
- Query evaluations of probabilistic database.
- ....

Approximate Counting

Many interesting problems in the exact counting regimes, including counting edge cover, is hard (#P-complete). Instead we look for these two types of polynomial time approximation scheme:

Definition (FPTAS)

For given parameter $\varepsilon > 0$ and an instance of a particular problem class, if the algorithm outputs a number $\hat{N}$ such that $(1-\varepsilon)N \leq \hat{N} \leq (1+\varepsilon)N$, where N is the accurate answer of the problem instance, and the running time is bounded by $poly(n, 1/\varepsilon)$ with n being the size of instance, this is called the **FPTAS (fully polynomial time approximation scheme)**.

Definition (FPRAS)

A randomized relaxation of FPTAS is known as **FPRAS (fully polynomial time randomized approximation scheme)**, which uses random bits and only outputs $\hat{N}$ to the desired precision with high probability.

Approximate Counting

Many interesting problems in the exact counting regimes, including counting edge cover, is hard (#P-complete). Instead we look for these two types of polynomial time approximation scheme:

Definition (FPTAS)

For given parameter $\varepsilon > 0$ and an instance of a particular problem class, if the algorithm outputs a number $\hat{N}$ such that $(1-\varepsilon)N \leq \hat{N} \leq (1+\varepsilon)N$, where N is the accurate answer of the problem instance, and the running time is bounded by $poly(n, 1/\varepsilon)$ with n being the size of instance, this is called the **FPTAS (fully polynomial time approximation scheme)**.

Definition (FPRAS)

A randomized relaxation of FPTAS is known as **FPRAS (fully polynomial time randomized approximation scheme)**, which uses random bits and only outputs $\hat{N}$ to the desired precision with high probability.

Approximate Counting

Many interesting problems in the exact counting regimes, including counting edge cover, is hard (#P-complete). Instead we look for these two types of polynomial time approximation scheme:

Definition (FPTAS)

For given parameter $\varepsilon > 0$ and an instance of a particular problem class, if the algorithm outputs a number $\hat{N}$ such that $(1-\varepsilon)N \le \hat{N} \le (1+\varepsilon)N$, where N is the accurate answer of the problem instance, and the running time is bounded by $poly(n, 1/\varepsilon)$ with n being the size of instance, this is called the **FPTAS (fully polynomial time approximation scheme)**.

Definition (FPRAS)

A randomized relaxation of FPTAS is known as **FPRAS (fully polynomial time randomized approximation scheme)**, which uses random bits and only outputs $\hat{N}$ to the desired precision with high probability.

Main Result

Previous work:

- Only an MCMC-based FPRAS is known for counting edge covers in graphs with maximum degree 3.
- The correlation decay based FPTAS for anti-ferromagnetic 2-spins systems (e.g. counting independent sets) goes beyond the best known MCMC based FPRAS and achieves the boundary of approximability.

Our main result: a correlation decay based FPTAS for general graphs. This provides another example where the tractable range for correlation decay based FPTAS exceeds the sampling based FPRAS.

Main Result

Previous work:

- Only an MCMC-based FPRAS is known for counting edge covers in graphs with maximum degree 3.
- The correlation decay based FPTAS for anti-ferromagnetic 2-spins systems (e.g. counting independent sets) goes beyond the best known MCMC based FPRAS and achieves the boundary of approximability.

Our main result: a correlation decay based FPTAS for *g*eneral graphs. This provides another example where the tractable range for correlation decay based FPTAS exceeds the sampling based FPRAS.

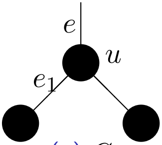
Dangling instance

Definition

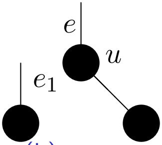
A **dangling edge** $e = (u, \_)$ of a graph is such singleton edge with exactly one end-point vertex u, as shown in the Figure 3a.

$$G - e \triangleq (V, E \setminus e)$$

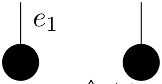
$$G - u \triangleq (V \setminus u, E - u)$$



(a) G



(b) $e_1 - u$



(c) $G - e - u \triangleq (G - e) - u$

Figure: Dangling edges examples.

Counting v.s. Marginal Probability

Recall our counting problem:

Problem

Let $EC(G)$ be the set of edge covers.
Goal: estimate $|EC(G)|$.

Let X be an edge cover sampled uniformly from $EC(G)$, consider the following marginal probability:
for an edge e, we write $P(G, e) \triangleq \Pr(e \notin X)$.

Solution: estimate $P(G, e)$.

Counting v.s. Marginal Probability

Recall our counting problem:

Problem

Let $EC(G)$ be the set of edge covers.
Goal: estimate $|EC(G)|$.

Let X be an edge cover sampled uniformly from $EC(G)$, consider the following marginal probability:
for an edge e, we write $P(G, e) \triangleq \Pr(e \notin X)$.

Solution: estimate $P(G, e)$.

Why $P(G, e)$?

Recall that the set of all edges E is an edge cover. For a randomly sampled edge cover X, what is $\Pr(X = E)$?

Let $E = \{e_i\}$, and $e_i = (u_i, v_i)$.

$$\Pr(X = E) = \frac{1}{|EC(G)|}$$

Therefore,

$$\frac{1}{|EC(G)|} = \prod_i (1 - P(G_i, e_i)) .$$

Why $P(G, e)$?

Recall that the set of all edges E is an edge cover. For a randomly sampled edge cover X, what is $\Pr(X = E)$?
Let $E = \{e_i\}$, and $e_i = (u_i, v_i)$.

$$\Pr(X = E) = \frac{1}{|EC(G)|}$$

Therefore,

$$\frac{1}{|EC(G)|} = \prod_i (1 - P(G_i, e_i)) .$$

Why $P(G, e)$?

Recall that the set of all edges E is an edge cover. For a randomly sampled edge cover X, what is $\Pr(X = E)$?
Let $E = \{e_i\}$, and $e_i = (u_i, v_i)$.

$$\Pr(X = E) = \frac{1}{|EC(G)|}$$

$$\Pr(X = E) = \Pr(\forall i, e_i \in X)$$

Therefore,

$$\frac{1}{|EC(G)|} = \prod_i (1 - P(G_i, e_i)) .$$

Why $P(G, e)$?

Recall that the set of all edges E is an edge cover. For a randomly sampled edge cover X, what is $\Pr(X = E)$?
Let $E = \{e_i\}$, and $e_i = (u_i, v_i)$.

$$\Pr(X = E) = \frac{1}{|EC(G)|}$$

$$\begin{aligned}\Pr(X = E) &= \Pr(\forall i, e_i \in X) \\ &= \Pr(e_1 \in X) \Pr(e_2 \in X \mid e_1 \in X) \cdots\end{aligned}$$

Therefore,

$$\frac{1}{|EC(G)|} = \prod_i (1 - P(G_i, e_i)) .$$

Why $P(G, e)$?

Recall that the set of all edges E is an edge cover. For a randomly sampled edge cover X, what is $\Pr(X = E)$?
Let $E = \{e_i\}$, and $e_i = (u_i, v_i)$.

$$\Pr(X = E) = \frac{1}{|EC(G)|}$$

$$\begin{aligned}\Pr(X = E) &= \Pr(\forall i, e_i \in X) \\ &= \Pr(e_1 \in X)\Pr(e_2 \in X \mid e_1 \in X)\cdots \\ &= \prod_i \Pr(e_i \in X \mid \{e_j\}_{j=1}^{i-1} \subseteq X)\end{aligned}$$

Therefore,

$$\frac{1}{|EC(G)|} = \prod_i (1 - P(G_i, e_i)) .$$

Why $P(G, e)$?

Recall that the set of all edges E is an edge cover. For a randomly sampled edge cover X, what is $\Pr(X = E)$?
Let $E = \{e_i\}$, and $e_i = (u_i, v_i)$.

$$\Pr(X = E) = \frac{1}{|EC(G)|}$$

$$\begin{aligned}\Pr(X = E) &= \Pr(\forall i, e_i \in X) \\ &= \Pr(e_1 \in X) \Pr(e_2 \in X \mid e_1 \in X) \cdots \\ &= \prod_i \Pr(e_i \in X \mid \{e_j\}_{j=1}^{i-1} \subseteq X) \\ &= \prod_i (1 - P(G_i, e_i)),\end{aligned}$$

where $G_1 = G, G_{i+1} = G_i - e_i - u_i - v_i$.
Therefore,

$$\frac{1}{|EC(G)|} = \prod_i (1 - P(G_i, e_i)).$$

Why $P(G, e)$?

Recall that the set of all edges E is an edge cover. For a randomly sampled edge cover X, what is $\Pr(X = E)$?
Let $E = \{e_i\}$, and $e_i = (u_i, v_i)$.

$$\Pr(X = E) = \frac{1}{|EC(G)|}$$

$$\begin{aligned}\Pr(X = E) &= \Pr(\forall i, e_i \in X) \\ &= \Pr(e_1 \in X)\Pr(e_2 \in X \mid e_1 \in X)\cdots \\ &= \prod_i \Pr(e_i \in X \mid \{e_j\}_{j=1}^{i-1} \subseteq X) \\ &= \prod_i (1 - P(G_i, e_i)),\end{aligned}$$

where $G_1 = G, G_{i+1} = G_i - e_i - u_i - v_i$.
Therefore,

$$\frac{1}{|EC(G)|} = \prod_i (1 - P(G_i, e_i)).$$

Why $P(G, e)$?

Recall that the set of all edges E is an edge cover. For a randomly sampled edge cover X, what is $\Pr(X = E)$?
Let $E = \{e_i\}$, and $e_i = (u_i, v_i)$.

$$\Pr(X = E) = \frac{1}{|EC(G)|}$$

$$\begin{aligned}
\Pr(X = E) &= \Pr(\forall i, e_i \in X) \\
&= \Pr(e_1 \in X) \Pr(e_2 \in X \mid e_1 \in X) \cdots \\
&= \prod_i \Pr(e_i \in X \mid \{e_j\}_{j=1}^{i-1} \subseteq X) \\
&= \prod_i \left(1 - P(G_i, e_i)\right),
\end{aligned}$$

where $G_1 = G, G_{i+1} = G_i - e_i - u_i - v_i$.
Therefore,

$$\frac{1}{|EC(G)|} = \prod_i \left(1 - P(G_i, e_i)\right).$$

Estimating Marginal Probability

Overall work-flow:

- Derive a computation tree recursion for $P(G, e)$ from that of smaller instances. TBA.
- Truncate the computation tree at depth L to get $P(G, e, L)$, for some notion of tree depth L.
- Show exponential correlation decay with respect to that tree depth:

$$|P(G, e, L) - P(G, e)| \leq exp(-\Omega(L))$$

Computation Tree Recursion

We focus on e is dangling.

$$P(G,e) = \frac{1 - \prod_{i=1}^{d} P(G_i, e_i)}{2 - \prod_{i=1}^{d} P(G_i, e_i)},$$

where $G_1 \triangleq G - e - u$, and $G_{i+1} \triangleq G_i - e_i$.

Truncate with a modified recursion depth:

$$P(G,e,L) = \begin{cases} \frac{1}{2}, & \text{if } L \leq 0; \\ \frac{1-\prod_{i=1}^{d} P(G_i,e_i,L-\lceil \log_6 (d+1) \rceil)}{2-\prod_{i=1}^{d} P(G_i,e_i,L-\lceil \log_6 (d+1) \rceil)}, & \text{otherwise.} \end{cases}$$

Computation Tree Recursion

We focus on e is dangling.

$$P(G,e) = \frac{1 - \prod_{i=1}^{d} P(G_i, e_i)}{2 - \prod_{i=1}^{d} P(G_i, e_i)},$$

where $G_1 \triangleq G - e - u$, and $G_{i+1} \triangleq G_i - e_i$.
Truncate with a modified recursion depth:

$$P(G,e,L) = \begin{cases} \frac{1}{2}, & \text{if } L \le 0; \\ \frac{1-\prod_{i=1}^{d} P(G_i,e_i,L-\lceil\log_6{(d+1)}\rceil)}{2-\prod_{i=1}^{d} P(G_i,e_i,L-\lceil\log_6{(d+1)}\rceil)}, & \text{otherwise.} \end{cases}$$

Correlation Decay

Proposition

Given graph G, edge e and depth L,

$$|P(G, e, L) - P(G, e)| \leq 3 \cdot (\frac{1}{2})^{L+1}$$

Thank you!