On the Efficient Generation Of Language Instances

Kartik Kurupaswamy
212123027 | kartik.kurupaswamy@iitg.ac.in

April 24, 2023

Introduction

Consider a well-known language in NP, such as the set of Hamiltonian graphs in a suitable language encoding. Given a length, it is straightforward to determine whether or not there exists a string in this language having that particular length. Also it is quite easy to construct such a string if it exists: after determining n and m such that a graph with n vertices and m edges has an encoding with the given length, and such that $m \leq n$, simply construct a Hamiltonian cycle and then add extra edges if necessary. With a little more thought it is also possible to come up with an efficient (polynomial-time) nondeterministic or probabilistic procedure that can output all Hamiltonian graphs having a given encoding length (or equivalently, having a given number of vertices and edges), although this procedure will probably include repetitions and will not necessarily output the graphs with uniform probabilities. Here we will investigate whether such efficient construction or generation procedures exist for all languages in P or NP.

Some Definitions

Acceptor

A Turing machine that (when it halts) either accepts or rejects its input is called an *acceptor*.

A Turing machine that, in addition to the input and working tapes, has an output tape is called a *transducer*.

Categorical TM

A nondeterministic Turing machine acceptor M is *categorical* if for each x there exists at most one computation of M on input x that leads to acceptance.

The class of languages in NP that are accepted by categorical NP machines is called UP.

Polynomial-time Detector

A *polynomial-time detector (PTD)* for a language L is a *deterministic* Turing machine transducer that runs in polynomial time and that on input 1^n, outputs 1 if there exist strings of length n in L and outputs 0 otherwise.

Polynomial-time Constructor

A *polynomial-time constructor (PTC)* for a language L is a *deterministic* Turing machine transducer that runs in polynomial time and that on input 1^n, outputs a string in L of length n, if such a string exists, and outputs Λ otherwise.

Polynomial-time Generator

A *polynomial-time generator (PTG)* for a language L is a *nondeterministic* Turing machine transducer that runs in polynomial time and that on input 1^n, outputs a string in L of length n, if such a string exists, and outputs Λ otherwise. Moreover, for each string $x \in L$ of length n there should exist some computation of the generator on input 1^n that outputs x.

Categorical PTG

A *categorical PTG* for a language L is a PTG for L such that, for each string $x \in L$ of length n, there is exactly one computation of the generator on input 1^n that outputs x.

Preliminaries III

Lemma

1. *If a language has a PTG it has a PTC; if a language has a PTC it has a PTD.*
2. *If a language has a PTG it is in NP.*
3. *A language in NP has a PTC if and only if it has a PTG.*

Results on PTG I

Theorem

1. *If $P \neq NP$, then there exists a PTG for a language in NP-P.*
2. *If $P \neq UP$, then there exists a categorical PTG for a language in UP-P.*

Definition

An NP machine for a language L is *traceable* if there exists a polynomial-time procedure that on input $x \in L$ outputs an accepting computation of the machine on input x.

Results on PTG II

Theorem

1. *$P = NP$ if and only if all PTGs are traceable.*
2. *$P = UP$ if and only if all categorical PTGs are traceable.*
3. *$P = UP \cap$ co-UP if and only if all categorical PTGs for languages in P are traceable.*
4. *If $P \neq NP \cap$ co-NP, then there exists a PTG for a language in P that is not traceable.*

Conclusion

Thanks man

Thank You

for your attention.

Do you have any question?