Daisies and Their Applications

Oded Lachish

Based on joint works with Eldar Fischer, Tom Gur and Yadu Vadusev

Setting

- Sublinear algorithms
- Complexity parameter: Query complexity
  - Property testing
  - (relaxed) Locally decodable codes

Querying versus Sampling

Querying – "smart" selection of queries that depends on the goal.

Sampling – every bit is sampled independently with the same probability.

Querying versus Sampling

Querying –

“smart” selection of queries that depends on the goal.

Result - **optimal use of queries**, but

queries are not guaranteed to be reusable!

Sampling –

every bit is sampled independently with the same probability.

Result - **wasteful use of queries**, but

queries are reusable!

We are interested in converting Querying algorithms to sampling algorithms

Converting Querying to Sampling

Implications (mostly due to reusability):

- GL'19 - Lower bounds on relaxed locally decodable codes
- FLV'14 – for every testable property there exists a non-trivial tester:
  - Multi-testing – can use o(n) samples for testing >>> *n* testable properties
  - Privacy – query oracle can't tell which property is tested
  - Union of very a large number of testable properties is non-trivially testable

Conversion: naïve idea

Setting:

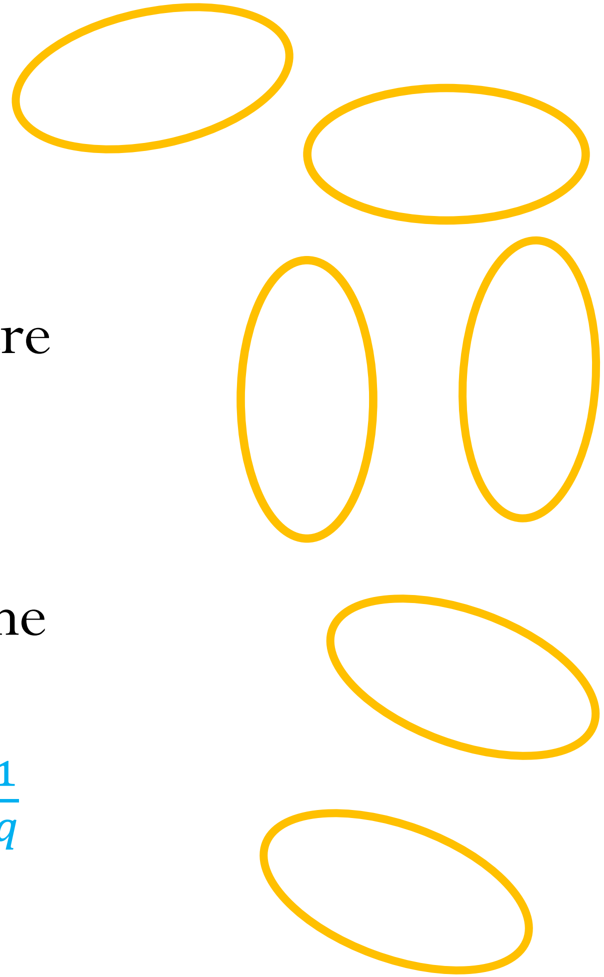
- Input alphabet is $\{0,1\}$
- Querying algorithm is non-adaptive and can be viewed as selecting a set of queries from a distribution over sets of queries of size q

Todo:

- Prove a volume lemma or two – the union of sets in the support that are "good" is large (their union is linear in the input size n)

- Prove that, with high probability, a set of samples contains a "good" set of queries

Very wishful thinking



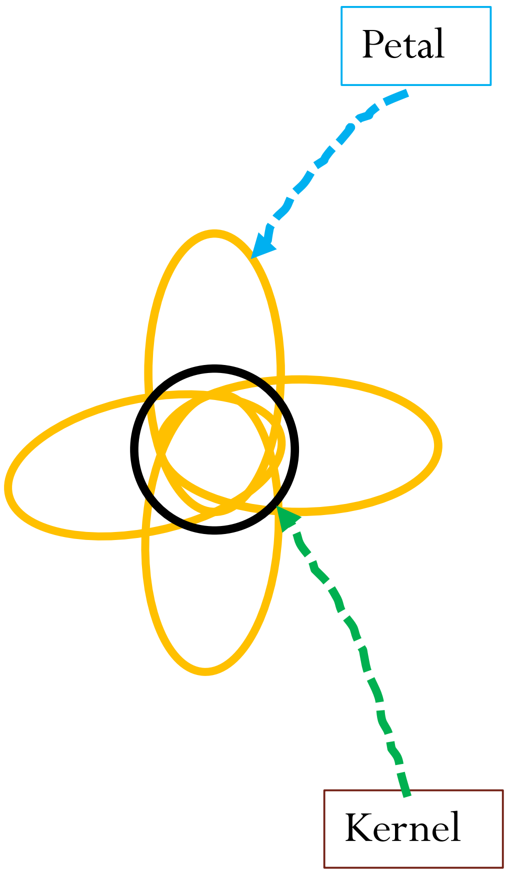
The sets in the support of the distribution are pairwise disjoint.

Sampling should work if

- The union of the "good" sets is linear in the input size
- Sampler probability is about is about $n^{-\frac{1}{q}}$

Problem: Sunflowers

- A family of sets S is a *sunflower* if there exists a set K such that the intersection of every pair of distinct sets in A,B in S is K.



What if the support of the querying algorithm is a sunflower.

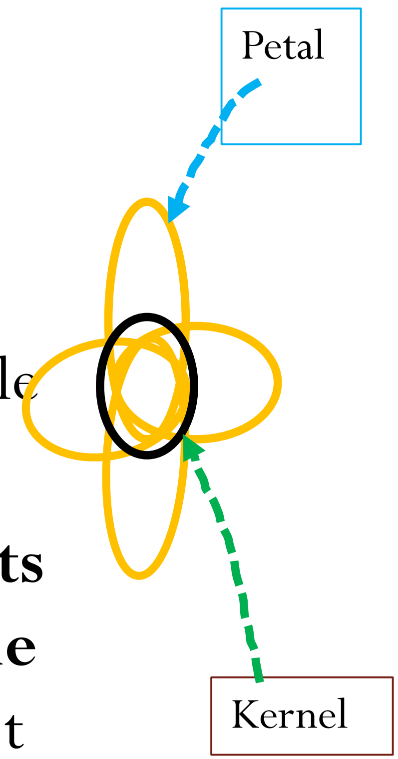
The probability of sampling the Kernel is too small. So, forget about seeing a set from the support.

Actually sunflowers are nice

What if the support of the querying algorithm is a sunflower.

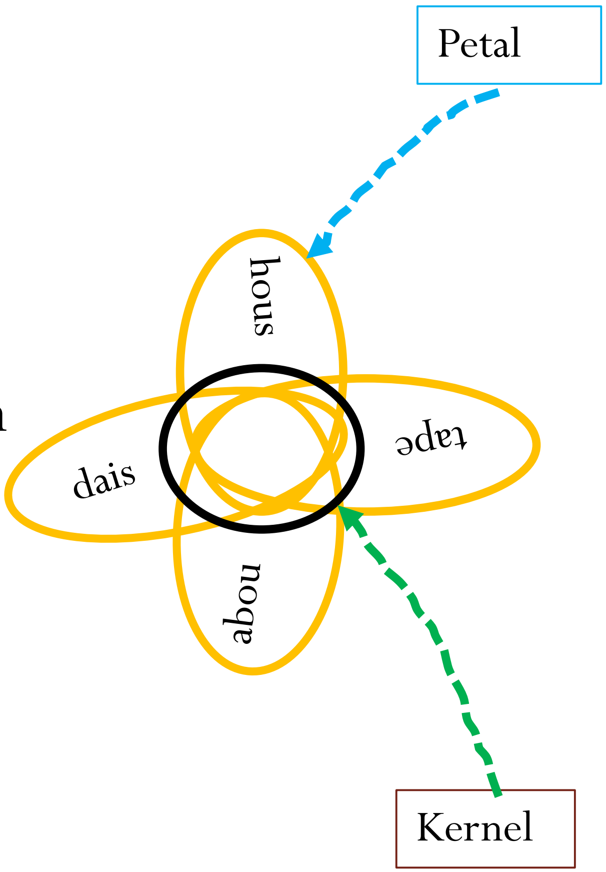
The probability of sampling the Kernel is too small. So forget about support.

- However, there is a good chance of sampling a whole petal, and
- in the settings of our interest, **changing a few bits in the input doesn't change the results of the algorithms by much** (or at least nothing we can't handle)



Sunflowers

- Suppose the problem was checking whether a crossword puzzle is filled correctly or far from that.
- Every set is supposed to be a natural language word.
- If it is far from being filled correctly, for every guess of the letter in the kernel, with high probability, the sample is going to contain a petal that rules it out.



The PROBLEM with sunflowers

- The support may not be a sunflower.
- Ideally, we would like to partition the family of sets into *poly(q)* disjoint sunflowers.

Solution: look for other flowers

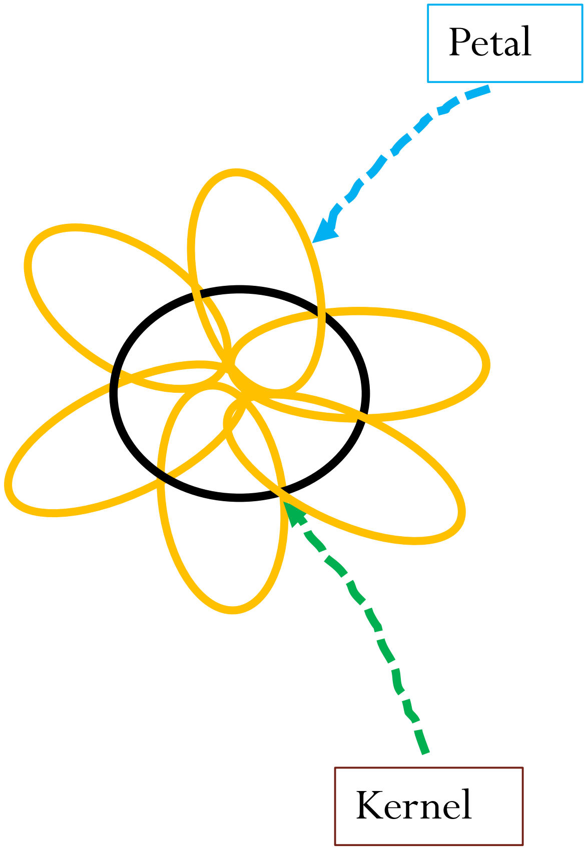
Daisies (Wikipedia)

- “The species habitually colonises lawns”, and
- “is difficult to eradicate by mowing – hence the term 'lawn daisy'. Wherever it appears it is often considered an invasive weed.”
- “The flower heads are composite”

Simple Daisy

- A family of sets S is a *simple daisy* if there exists a set K such that the intersection of every pair of distinct sets in A, B in S is a **subset** of K.

- Same ideas as before work if there are enough petals.



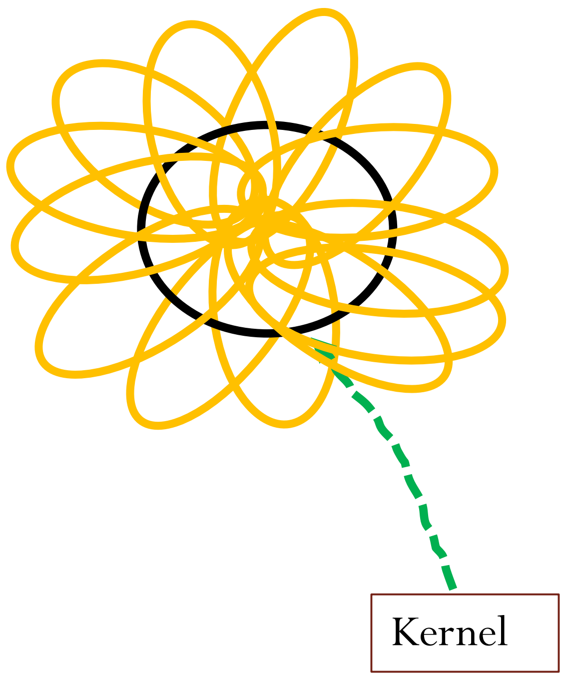
Problem: finding simple daisies.

t-daisy

- A family of sets S is a *t-daisy* if there exists a set K such that any x outside is in at most t petals.

The advantages of t-daisies.

We can actually partition the support of the query algorithm into daisies and we can extract simple daises from them.



t-daisy partition lemma

(Important – the sets are the sets in the support that the querying algorithm uses, we assume there number is *cn c- constant, n size of input*)

Let S be the support.

The kernel of the first daisy K_1, is the

set of every x that is in at least $cn^{\frac{1}{q}}$

sets from S.

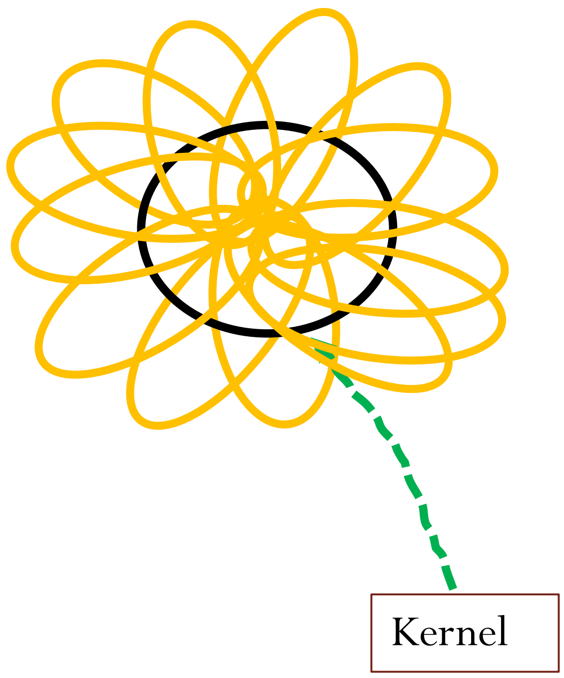
n - is the size of the input,

C - is a constant

The daisies sets are the sets of S

That have an intersection

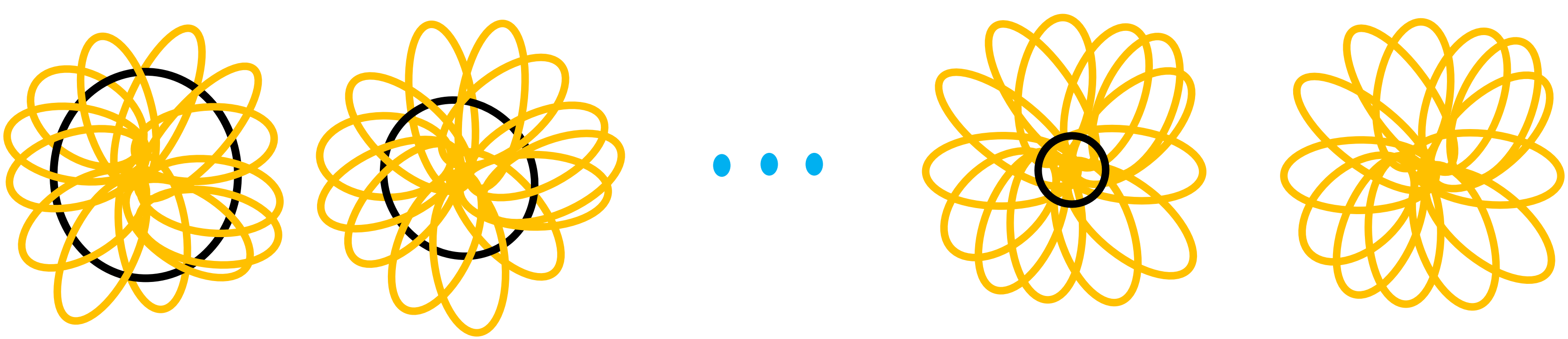
of size $q-1$ or more with K.



t-daisy partition lemma

- Remove the sets of daisy *i-1* from *S*.

  The kernel of the i'th daisy K_i, is the set of every x that is in at least $cn^{\frac{i}{q}}$ sets from S.
- The daisies sets are the sets of *S* that have an intersection of size exactly *q-i* with *K*.




Thank You