Dispersers and extractors

Alexey Vyskubov

29.09.2004

Randomized algorithms depends on *coin tosses.*

We can decrease error probability exponentially, repeating the algorithm.

But we need more random bits. In real world random bits are hard to obtain. **Can we save on the number of random bits used to achieve a certain error probability?**

We can get source of "some randomness". **Can we simulate our algorithm with bit sequence with "some randomness" instead of unbiased independent bits?**

Or: **Can we replace the randomized algorithm with a deterministic one, thus avoiding using randomness altogether?**

We concentrate on randomized algorithms solving decision problems, with a small probability of error on every instance.

We shall care much more about the number of random bits used than about other computational resources, such as time. But an exponential time is not interesting (we can run deterministic algorithm then). We accept any polynomial slowdown for saving a constant factor on the random bits used.

We shall concentrate on generic, general-purpose derandomization, without any knowledge about how the algorithm operates (except some bound on computational complexity).

Randomness-efficient amplification

The task is to obtain the same probability of error using fewer random bits, while preserving the efficiency.

1. We can obtain an error probability of $\frac{1}{r^k}$ for any desired constant k using r bits, i.e. using no additional randomness.
2. We can obtain an error probability of 2^{-R} for any desired R, using (a) $r + \mathcal{O}(R)$ random bits, and even (b) $(1+\varepsilon)(1+R)$ random bits for any desired constant $\varepsilon > 0$.
3. Given any constant $\varepsilon > 0$, we can, using R random bits, where R is a certain polynomial in r, obtain an error probability of $2^{-R+R^{\varepsilon}}$.
4. If the algorithem has one-sided error (i.e. has error probability 0 on negative instances), we can, using R random bits for any parameter $R \geq r$, obtain an error probability of $2^{-R+r+\mathrm{polylog} R}$.

Computing with weak random sources

Let's suppose we have a randomized algorithm using r random bits to achieve an error probability of $\frac{1}{10}$. Where we can get those r random bits?

The objective: to simulate the algorithm using the weak random source instead of independent, unbiased coin flips, while preserving (roughly) the efficiency and (precisely) the error probability.

We need to formalize this task before we can try to solve it. We need some measure on the amount of randomness contained in a source.

The first idea would be to use Shannon entropy. It does not work.

Let $R > r$.

Probability $\frac{99}{100}$: $(0, 0, \ldots, 0) \in \{0,1\}^R$.

Probability $\frac{1}{100}$: uniformly chosen value in $\{0,1\}^R$.

Shannon's entropy is as big as we wish, but we cannot simulate our randomized algorithm with success probability $9/10$.

We are more interested in "stable" randomness.

We say that a source has **min-entropy** at least t if any outcome of the source has probability at most 2^{-t}.

$$-\sum p_i \log p_i \geq \sum p_i t = t$$

As we can see, min-entropy is smaller than Shannon entropy.

For the previous example min-entropy is very small: $(0, 0, \ldots, 0)$ is very likely to occur.

5. For any constant $\varepsilon > 0$ and any randomized polynomial algorithm A using r bits to obtain an error probability of $1/3$, there is a randomized polynomial algorithm A' which, instead of r random bits, takes an auxiliary input $y \in \{0,1\}^R$, where R is polynomial in r. If y is sampled from any R-bit source with min-entropy at least R^ε, the error probability of the original algorithm is maintained.

Deterministic simulation

I will not talk about this today.

- **Dispersers**

  “Get few long random sequences from one short”

- **Extractors**

  “Extract randomness from weak source”

- **Expanders**

  “Graph where vertices have many neighbours”

Alice and Bob are playing card game using a deck of N cards; $\varepsilon > 0$ is a constant.

The first player, Alice, secretely chooses at least $(1-\varepsilon)N$ cards and puts a mark on the face of the cards chosen. Then she places all the cards in the deck, face down, before the second player, Bob.

Now Bob must turn over at most d cards. If a mark is revealed, Bob wins the game.

Interesting situation: $d \leq \varepsilon N$.

- No deterministic strategy for Bob can be good.
- Good randomized strategy for Bob is to chose d cards to turn over randomly, without replacement, yielding a loosing probability for Bob which can be bounded from above by ε^d.

Approximately $d \log N$ random bits are needed. Let's make the game more interesting: Bob is allowed to use only R random bits.

R**-bit strategy** for Bob:

a map φ from $\{0,1\}^R$ to the set of subsets of $\{1,\ldots,N\}$ of size at most d.

Bob executes the startegy by picking $x \in \{0,1\}^R$ uniformly at random and turning over the cards of $\varphi(x)$.

The general task: find $(N, \varepsilon, R, d, p)$-strategy which makes the probability for Bob to loose the game smaller than p.

Definiton. A bipartite graph on vertex sets U and V and with edges $E \subseteq U \times V$ is called an $(1-\varepsilon)$-hitting disperser with threshold T if, for all subsets S of U of size $|S| \geq T$, we have that the size of the set of neighbours of S in V, $|\Gamma(S)|$, is more than $\varepsilon|V|$.

There is an obvious correspondence between R-bit strategies and bipartite graphs: $U = \{0,1\}^R$, $V = \{1, \ldots, N\}$, edges are defined by $\Gamma(x) = \varphi(x)$ for $x \in U$, where $\Gamma(x)$ is the set of neighbours in V of x.

Proposition. A strategy is an $(N, \varepsilon, R, d, p)$-strategy if and only if the corresponding graph is a $(1-\varepsilon)$-hitting disperser with threshold $p|U|$.

Proof, part 1: the graph of a good strategy is a good disperser.

If it is not, there is a subset of the size at least of $p|U|$ in U such that it has less than $\varepsilon|V|$ neighbours. Alice marks cards which are not those neighbours and wins with the probability at least $1-p$.

Proof, part 2: the good disperser defines a good strategy.

If it does not, there is a subset S of V with the size at least $(1-\varepsilon)|V|$ such that $|\Gamma(V \setminus S)| \geq p|U|$. Contradiction with disperser definition.

An example of usage of dispersers

Let L be some language. Suppose we have a randomized algorithm for L using r random bits on instances of a certain size and with error probability at most ε (one-sided error, the answer $x \notin L$ is always right).

Let $N = 2^r$, and let Alice's cards be the possible sequences of r random coin tosses. For a positive instance $x \in L$, we let the marked cards be the coin toss sequences leading to acceptance of x.

New algorithm, using R random bits: take a random $y \in \{0,1\}^R$, put it in the disperser and run the original algorithm for all coin tosses we got from disperser. If it accepts at least once, the new algorithm accepts, otherwise the new algorithm rejects.

The error probability is at most Bob's risk of loosing the game.

But we need efficient disperser.

Definition. A family of dispersers $E_r \subseteq U_r \times V_r$ with $|V_r| = 2^r$ is explicit, if there is an algorithm which on input r and $x \in U_r$ enumerates the neigbours of x in V_r, in time polynomial in $r, \log |U_r|$ and the number of such vertices.

"Master lemma"

If there is an explicit family of 1/2-hitting dispersers $E_r \subseteq U_r \times V_r, r = 1, 2, \ldots$ with $|V_r| = 2^r, |U_r| = 2^{R(r)}$, degree $d(r)$ for each vertex in U_r, and threshold $T(r)$, then a one-sided error algorithm running it time t, using r random bits to achieve error probability 1/2 can be converted to a one-sided error algorithm, running in time $(R(r) + d(r) + t)^{\mathcal{O}(1)}$ using $R(r)$ random bits to achive error probability $T(r)/2^{R(r)}$.

$T(r)$ — at most $T(r)$ coin toss sequences make the algorithm give the wrong output.

In the previous lemma we actually do not need real random source. We just want to make sure we are not hitting bad sequences (at most $T(r)$) too often. If the source has min-entropy at least $\log T(r) + 2^k$, each of those sequences will be given with probability at most $2^{-k}/T(r)$.

"Master lemma" for weak random source

If there is an explicit family of 1/2-hitting dispersers $E_r \subseteq U_r \times V_r, r = 1, 2, \ldots$ with $|V_r| = 2^r, |U_r| = 2^{R(r)}$, degree $d(r)$ for each vertex in U_r, and threshold $T(r)$, then a one-sided error algorithm running it time t, using r random bits to achieve error probability 1/2 can be simulated by an algorithm running in time $(R(r) + d(r) + t)^{\mathcal{O}(1)}$ and using any $R(r)$-bit weak random source with min-entropy at least $\log T(r) + k$ to achive error probability 2^{-k}.

What about two-sided error algorithms?

We modify the game: Bob wins if more the half of the cards he opened are marked by Alice. Such strategy will allow to distinguish between a situation where Alice marks at least $1 - \varepsilon$ fraction of the cards from a situation where she marks at most an ε fraction of the cards.

It is easy to modify the definition of dispenser for this case. This would lead to *majority disperser.* We will use so-called *extractors*, not majority disperses; they are better suited to being used as building blocks in other constructions.

Choosing cards uniformly at random is a good strategy for Bob.

Why? Chernoff bound: if X_i are independent 0-1 random variables, $X = \sum X_i, \mu = E[X]$

$$\Pr(X < (1 - \delta)\mu) < e^{-\mu\delta^2/2}$$

We will start with defining and analyzing a closeness measure for probability distributions.

Definition We say that two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are ε-close if their L_1-distance $||\mathcal{D}_1 - \mathcal{D}_2||$ is at most ε:

$$\sum_i |p_{1,i} - p_{2,i}| \leq \varepsilon$$

Lemma Two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ on domain S are ε-close if and only if, for any event $T \subseteq S$,

$$|\mathrm{Pr}_1(x \in T) - \mathrm{Pr}_2(x \in T)| \leq \varepsilon/2.$$

One can think about T as about a statistical test.

"If collisions are rare, distribution is close to uniform."

Lemma Let $\mathcal{D}$ be a probability distribution on a domain of size s. If collision probability of $\mathcal{D}$ is at most $(1+\varepsilon)/s$, then $\mathcal{D}$ is $\sqrt{\varepsilon}$-close to uniform.

"If distribution is not close to uniform, it is predictable."

$\mathcal{D}$ on $\{0,1\}^m$ is ε-predictable if there is a function (the predictor) $f : \{0,1\}^{i-1} \to \{0,1\}$:

$$\Pr(f(x_{1\ldots(i-1)} = x_i) \geq 1/2 + \varepsilon.$$

Lemma If $\mathcal{D}$ on $\{0,1\}^m$ is not ε-close to uniform, then it is $e/(2m)$-predictable.

Definition A bipartite graph on vertex sets U and V and with edges $E \subseteq U \times V$ so that every vertex x in U has exactly d neighbours is called an extractor with error ε and min-entropy threshold t if the following holds: If $x \in U$ is chosen accordingly to any distribution with min-entropy at least t, and a vertex $y \in V$ is then chosen uniformly at random from the neigbours of x, the resulting distribution of $y \in V$ is ε-close to uniform.

Notice: exctrator needs weak random source **and** some small number of truly random bits.

Explicit families of extractors are defined in obvious way.

Proposition An extractor with error strictly less than ε and min-entropy threshold t is also an $\varepsilon/2$-hitting disperser with threshold 2^t.

Extractors are also majority dispersers.

Lemma If there is an explicit family of extractors $E_r \subseteq U_r \times V_r, r = 1, 2, \ldots$ with $|V_r| = 2^r, |U_r| = 2^{R(r)}$, degree $d(r)$, min-entropy threshold $t(r)$, and error less than $1/3$, then a two-sided error algorithm running it time T, using r random bits to achieve an error probability of $1/3$ can be converted to a two-sided error algorithm, running in time $(R(r) + d(r) + T)^{\mathcal{O}(1)}$ using $R(r)$ random bits to achive error probability $2^{t(r)-R(r)}$.

Sketch of the proof

1. $R(r)$ bits are used as an input for the extractor.
2. We run the original algorithm for all possible extractor outcomes.
3. We accept if in majority of the cases the original algorithm accepts.

Suppose that at least a $2/3$ fraction of V_r corresponds to accepting computations of the original algorithm. We need to show that less than $2^{t(r)}$ choices cause reject. If it is not the case, let's take uniform distribution over that choices and show that the result is not $1/3$-close to uniform.

Similar result for weak random sources.

Lemma If there is an explicit family of extractors $E_r \subseteq U_r \times V_r, r = 1, 2, \ldots$ with $|V_r| = 2^r, |U_r| = 2^{R(r)}$, degree $d(r)$, min-entropy threshold $t(r)$, and error less than $1/3$, then a two-sided error algorithm running it time T, using r random bits to achieve an error probability of $1/3$ can be simulated by an algorithm running in time $(R(r) + d(r) + T)^{\mathcal{O}(1)}$ and using any $R(r)$-bit weak random source with min-entropy at least $t(r) + k$ to achive an error probability 2^{-k}.

Now we are going to construct dispersers. There are two ways to build them from so-called expanders.

Definition Let $G = (V, E)$ be an undirected d-regular graph. We say that G is a c-expander, for $c > 0$, if for any $V_1 \subset V$ with $|V_1| \leq |V|/2$, we have $|V_1 \cup \Gamma(V_1)| \geq (1 + c)|V_1|$.

There are families of explicit expanders with $|V| \to \infty$ and the parameters c and d being constants.

Example: 5-regular Gabber-Galil expander

$V = \{0, 1\} \times (\mathbb{Z}/m\mathbb{Z})^2$

Neighbours of (b, x, y) are $(1 - b, x, y), (1 - b, x, x + y), (1 - b, x, x + y - 1), (1 - b, x + y, y), (1 - b, x + y + 1, y)$.

This graph is a c-expander for some small constant c.

The proof is extremely complicated.

Let $U = V = V'$ and for some $l \geq 1$ let
$E = \{(u, v) | \text{there is a path of length at most } l \text{ from } u \text{ to } v\}$.

It follows directly from the definitions that, for any $\varepsilon > 0$, (U, V, E) is an $(1/2 + \varepsilon)$-hitting disperser with threshold $1/2(1 + c)^{-l}$ and degree at most d^{l+1}.

Theorem For any desired constant k, there is an explicit 1/2-hitting disperser $E_r \subset V_r \times V_r$ with $|V_r| = 2^r$, degree $r^{\mathcal{O}(1)}$ and threshold $2^r/r^k$.

(Gammer-Galil works only for odd r but we always can reduce the number of random bits used by an additive constant, by trying all possible settings of the bits removed.)

This theorem together with "master lemma" gives us the statement 1 from introduction for one-sided error algorithms; using majority dispersers it is possible to modify the proof slightly to make it work for two-sided error.

1. For each of the vertices V' of the expander we label outgoing edges $1, 2, \dots, d$ in an arbitrary way.
2. We choose some parameter l.
3. $U = V' \times \{1, \dots, d\}^l, V = V'$
4. $((u, e_1, \dots, e_l), v)$ is an edge if the walk starting in u and following edges labeled $(e_1, \dots, e_l)$ (in that order) encounters v at some point.

So we have $|U| = d^l|V|$, and degree is l. It is shown that this construction is 1/2-hitting disperser with threshold $2^{-\mathcal{O}(l)}|U|$, and a majority disperser with similar parameters.

This gives as statement 2 (a) of the introduction.