Tight Bounds for Inverting Permutations via Compressed Oracle Arguments

Ansis Rosmanis

Graduate School of Mathematics, Nagoya University, Japan

Summary

Consider the following random oracle problems.

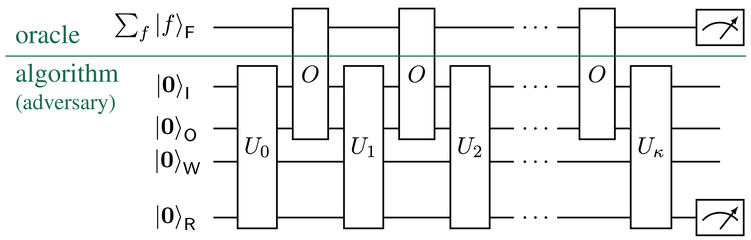
- UNSTRUCTURED SEARCH: Given a random function $f\colon \{0,1\}^m \to \{0,1\}^n$, find some x such that $f(x) = 0^n$.
- INVERTING PERMUTATION: Given a random permutation $f\colon \{0,1\}^n \to \{0,1\}^n$, find the *unique* x such that $f(x) = 0^n$.

In [Crypto 2019] Zhandry introduced the compressed oracle technique and, among other things, showed how to use it to reprove that the success probability for any κ-query algorithm for UNSTRUCTURED SEARCH is in $\mathrm{O}(\kappa^2/2^n)$. We show how an approach similar to the compressed oracle technique can be used to reprove the same bound for INVERTING PERMUTATION.

Oracle-Algorithm Interaction

If we consider any given function f, the algorithm has access to the oracle call $O_f\colon |x,y\rangle \mapsto |x, y\oplus f(x)\rangle$ and the projector on successful outcomes is $P_f := \sum_{x\colon f(x)=0^n} |x\rangle\langle x|$.

If the oracle holds a superposition over functions f, the algorithm interacts with the oracle via $O := \sum_f |f\rangle\langle f| \otimes O_f$ and the projector on successful function-outcome pairs is $P := \sum_f |f\rangle\langle f| \otimes P_f$.



The Basic Idea Behind the Bound

The state of the oracle has to undergo certain evolution in order for the algorithm to succeed, yet each oracle call cannot evolve the oracle state by much. (The quantum adversary method uses the same idea.)

Modifying Zhandry's Approach

To obtain our bound for INVERTING PERMUTATION, we first introduce a couple of modifications to Zhandry's analysis for UNSTRUCTURED SEARCH, and then we rewrite the modified analysis to be more suitable for permutations.

Modification 1 : *Zhandry used a time-efficient operation $\tilde{O}$ that, from algorithm's point of view, is perfectly indistinguishable from calling O on a uniform superposition of f. We simply use the latter.*

Independent Registers for Function Values

The oracle register F containing the function can be decomposed into 2^m registers $\mathsf{F}_0, \mathsf{F}_1, \mathsf{F}_2, \dots, \mathsf{F}_{2^m-1}$, each of dimension 2^n and containing one specific value of the function, and we can write

$$|f\rangle_{\mathsf{F}} = |f(0_{\mathsf{b}})\rangle_{\mathsf{F}_0}|f(1_{\mathsf{b}})\rangle_{\mathsf{F}_1}|f(2_{\mathsf{b}})\rangle_{\mathsf{F}_2}\cdots|f((2^m-1)_{\mathsf{b}})\rangle_{\mathsf{F}_{2^m-1}}.$$

Here y_{b} is y written in binary.

Observation 1 : *Since initially the oracle contains a uniform superposition over all 2^{n2^m} functions $f\colon \{0,1\}^m \to \{0,1\}^n$, its initial state is $|\widehat{\emptyset}\rangle_{\mathsf{F}_0}|\widehat{\emptyset}\rangle_{\mathsf{F}_1}|\widehat{\emptyset}\rangle_{\mathsf{F}_2}\cdots|\widehat{\emptyset}\rangle_{\mathsf{F}_{2^m-1}}$ where $|\widehat{\emptyset}\rangle := \sum_y |y\rangle/\sqrt{2^n}$.*

We interpret the state of the oracle register F as a "database" of the answers the oracle has so far given to the algorithm. We interpret the state of the single value register F_x as

- $|\widehat{\emptyset}\rangle$: the algorithm has not queried $f(x)$ and no input-output pair (x,y) is in the database.
- $\sum_y \beta_y|y\rangle$ orthogonal to $|\widehat{\emptyset}\rangle$: the algorithm has queried $f(x)$ and (x,y) is in the database with probability $|\beta_y|^2$; in particular, we interpret
  - $|r_0\rangle := \Big(|0^n\rangle - \frac{1}{2^n-1}\sum_{y\neq 0^n}|y\rangle\Big)\sqrt{\frac{2^n-1}{2^n}}$: the algorithm has learnt that $f(x) = 0^n$;
  - $|other\rangle$: the algorithm has learnt $f(x) \neq 0^n$.

Modification 2 : *Zhandry essentially used a dedicated extra (2^n+1)-th dimension for every F_x to indicate $(x,\cdot)$ not being in the database.*

Observation 2 : *After k oracle calls, no more than k input-output pairs are in the database.*

Observation 3 : *The probability of the algorithm finding a valid solution is at most the probability of the database containing the pair $(x,0^n)$ for some x plus the minuscule term of $\mathrm{O}(1/2^n)$.*

Low and High Success Oracle Subspaces

For every register F_x, we interpret its state being in the space $\mathcal{H}_{\mathrm{oth}}$ that is spanned by all the states orthogonal to $|r_0\rangle$ (including $|\widehat{\emptyset}\rangle$) as the algorithm having not learnt that $f(x) = 0^n$.

For the algorithm to succeed, it suffices that for even one x the content of F_x is not in $\mathcal{H}_{\mathrm{oth}}$. Thus we define

- low success subspace $\mathcal{H}^{\mathrm{low}}$: $\mathcal{H}^{\mathrm{low}} = \mathcal{H}_{\mathrm{oth}} \otimes \mathcal{H}_{\mathrm{oth}} \otimes \ldots \otimes \mathcal{H}_{\mathrm{oth}}$;
- high success subspace $\mathcal{H}^{\mathrm{high}}$: the orthogonal complement of $\mathcal{H}^{\mathrm{low}}$.

Lower Bound for UNSTRUCTURED SEARCH

Let $|\psi_k\rangle$ be the state of the overall system after k queries.

Let $\alpha_k = \|(\Pi^{\mathrm{high}} \otimes I_{\mathsf{A}})|\psi_k\rangle\|$, where Π^{high} is the projector on $\mathcal{H}^{\mathrm{high}}$.

We get

- from the initial state: $\alpha_0 = 0$;
- from the oracle call: $\alpha_k \leq \alpha_{k-1} + \mathrm{O}\big(1/\sqrt{2^n}\big)$;
- from the final measurement: $p_{\mathrm{succ}} \leq \|\alpha_\kappa\|^2 + \mathrm{O}(1/2^n)$.

Alternative Definitions of $\mathcal{H}^{\mathrm{low}}$ and $\mathcal{H}^{\mathrm{high}}$

Define vectors $|v^{y_1,y_2,\ldots,y_k}_{x_1,x_2,\ldots,x_k}\rangle := \sum_{f\colon \forall i\ f(x_i)=y_i} |f\rangle$, and spaces

$$\begin{aligned}
\mathcal{A}_0 &:= \mathrm{span}\{|v_\emptyset\rangle\}, & \overline{\mathcal{H}}_0^{\mathrm{low}} &:= \mathcal{A}_0,\\
\mathcal{B}_1 &:= \mathrm{span}\{|v^{0^n}_{x_1}\rangle\colon x_1\}, & \overline{\mathcal{H}}_1^{\mathrm{high}} &:= \mathcal{B}_1 \cap \mathcal{A}_0^\perp,\\
\mathcal{A}_1 &:= \mathrm{span}\{|v^{y_1}_{x_1}\rangle\colon x_1, y_1\}, & \overline{\mathcal{H}}_1^{\mathrm{low}} &:= \mathcal{A}_1 \cap (\mathcal{B}_1 + \mathcal{A}_0)^\perp,\\
\mathcal{B}_2 &:= \mathrm{span}\{|v^{y_1,0^n}_{x_1,x_2}\rangle\colon x_1, y_1, x_2\}, & \overline{\mathcal{H}}_2^{\mathrm{high}} &:= \mathcal{B}_2 \cap \mathcal{A}_1^\perp,\\
\mathcal{A}_2 &:= \mathrm{span}\{|v^{y_1,y_2}_{x_1,x_2}\rangle\colon x_1, y_1, x_2, y_2\}, & \overline{\mathcal{H}}_2^{\mathrm{low}} &:= \mathcal{A}_2 \cap (\mathcal{B}_2 + \mathcal{A}_0)^\perp,\\
&\vdots & &\vdots
\end{aligned}$$

We accumulate the low and the high success subspaces as

$$\mathcal{H}_k^{\mathrm{low}} := \bigoplus_{k'=0}^{k} \overline{\mathcal{H}}_{k'}^{\mathrm{low}} \quad\text{and}\quad \mathcal{H}_k^{\mathrm{high}} := \bigoplus_{k'=1}^{k} \overline{\mathcal{H}}_{k'}^{\mathrm{high}},$$

and we have $\mathcal{H}_{2^m}^{\mathrm{low}} = \mathcal{H}^{\mathrm{low}}$ and $\mathcal{H}_{2^m}^{\mathrm{high}} = \mathcal{H}^{\mathrm{high}}$.

Adaptations for INVERTING PERMUTATION

Define vectors $|v^{y_1,y_2,\ldots,y_k}_{x_1,x_2,\ldots,x_k}\rangle$ as above, except summing only over permutations, and in turn define all the spaces $\mathcal{A}_k$, $\mathcal{B}_k$, $\overline{\mathcal{H}}_k^{\mathrm{low}}$, $\overline{\mathcal{H}}_k^{\mathrm{high}}$, $\mathcal{H}_k^{\mathrm{low}}$, $\mathcal{H}_k^{\mathrm{high}}$ as above. However, it is not useful to consider values of k close to the maximum, and thus we do not define $\mathcal{H}^{\mathrm{low}}$ and $\mathcal{H}^{\mathrm{high}}$.

Now define $\alpha_k = \|(\Pi_k^{\mathrm{high}} \otimes I_{\mathsf{A}})|\psi_k\rangle\|$, and we have

- $\alpha_0 = 0$;
- $\alpha_k \leq \alpha_{k-1} + \mathrm{O}\big(1/\sqrt{2^n - 4k}\big)$;
- $((\Pi_k^{\mathrm{low}} + \Pi_k^{\mathrm{high}}) \otimes I_{\mathsf{A}})|\psi_k\rangle = |\psi_k\rangle$;
- $p_{\mathrm{succ}} \leq \|\alpha_\kappa\|^2 + \mathrm{O}(1/(2^n - 2k))$.