The Function-Inversion Problem

Juraj Belohorec

Key Concepts and Objectives

Main objective In 1980, Hellman introduced time-space tradeoffs as a tool for cryptanalysis and presented a black-box preprocessing algorithm that inverts a function $f : [N] \to [N]$ using only $S = \tilde{O}(N^{2/3})$ bits of advice and online time $T = \tilde{O}(N^{2/3})$. Is it possible to improve upon Hellman's time-space trade-off?

Theorem 19 (Hellman). There exists a black-box algorithm for inverting permutations $\pi : [N] \to [N]$ that, for any $S, T \in \mathbb{Z}_{>0}$ satisfying $ST \geq 2N\lceil \log N + 1 \rceil$, uses T queries and S bits of advice and is T-round adaptive.

Algorithm

- There are at most $N/(T+1)$ cycles of length greater than T.
- For each such cycle, we store a sequence of "checkpoints" in the order they appear in the cycle, ensuring that every point on the cycle is at a distance of at most T points from the previous checkpoint.
- If the i-th cycle has length i, then that cycle requires $\lceil i/T \rceil + 1$ checkpoints to cover its first $i - T$ points and one checkpoint to cover the remaining $i \mod T$ points.
- In the worst case, if all cycles have length $T + 1$, we have $N/(T + 1)$ cycles, each requiring two checkpoints. This results in a total of $2N/(T + 1)$ checkpoints.
- Therefore, we can store these checkpoints as a list of lists using $2N\lceil \log N \rceil / T$ bits. Additionally, we add one extra bit to each checkpoint (at most $2N/T$ bits total) to indicate whether the checkpoint belongs to the same cycle as the prior one.

In the online phase, given a point $y \in [N]$ as input and the list of checkpoints as an advice string, we perform the following steps:

- Set $y_0 = y$.
- Iteratively compute $y_{i+1} = \pi(y_i)$ until either:
  - $y_i = y$, at which point we output y_{i-1} as the preimage of y, or
  - y_i is one of the stored checkpoints. In this case, set y_{i+1} to be the previous checkpoint in the list on the same cycle and continue iterating.

Theorem 3 If an explicit operator F_n with $n \in \mathbb{Z}_{>0}$ has fan-in-two Boolean circuits of size $O(n)$ and depth $O(\log n)$, then for every $\varepsilon > 0$, this operator admits a non-adaptive systematic data structure of size $O(n \log \log n)$ and query complexity $O(n)$.

Theorem 6. If, for some $\varepsilon > 0$, every family of strongly non-adaptive black-box algorithms for inverting functions $f : [N] \to [N]$ that uses $O(N)$ queries requires $\omega(N \log N \log \log N)$ bits of advice, then there exists an explicit operator that cannot be computed by fan-in-two Boolean circuits of size $O(n)$ and depth $O(\log n)$.

Definition (Multiparty pointer-jumping problem) In the *pointer-jumping problem* $\mathrm{MPJ}^{\mathrm{perm}}_{N,k}$, there are k computationally unbounded players, denoted $P_0, P_1, \ldots, P_{k-1}$, and each has an input "written on her forehead."

The first player P_0 has a point $x \in [N]$ written on her forehead, the last player P_{k-1} has a Boolean mapping $\pi_{k-1} : [N] \to \{0,1\}$ written on her forehead, and each remaining player P_i, for $i = 1, \ldots, k-2$, has a permutation $\pi_i : [N] \to [N]$ written on her forehead. Each player can see all $k-1$ inputs except the one written on her own forehead.

The goal of the players is to compute the value $\pi_{k-1}(\pi_{k-2}(\ldots \pi_1(x) \ldots))$, which loosely corresponds to "following a trail of pointers" defined by the permutations, starting from x. The players can communicate by writing messages on a public blackboard. The communication complexity of a protocol is the total number of bits written on the blackboard for a worst-case input.

Theorem 10. $\mathrm{CC}^1(\mathrm{MPJ}^{\mathrm{perm}}_{N,k}) \le O((Nk+N)\log N)$.

Lemma 11. $\mathrm{CC}^1(\mathrm{MPJ}^{\mathrm{perm}}_{N,k}) \le \mathrm{CC}^1(\hat{\mathrm{MPJ}}^{\mathrm{perm}}_{N,k}) + \lceil \log N \rceil$.

Lemma 12. If there exists a $(k-2)$-round adaptive algorithm for inverting permutations $\pi : [N] \to [N]$ that uses advice S and time T, then $\mathrm{CC}^1(\hat{\mathrm{MPJ}}^{\mathrm{perm}}_{N,k}) \le S + T\lceil \log N \rceil$.

Definition (Systematic substring-search problem) We are given a bitstring of length N (the "text") and a bitstring of length $P \ll N$ (the "pattern"). If the pattern appears in the text, we must output an index $i \in [N]$ into the text at which the pattern begins. We take the pattern length to be $P = \Theta(\log N)$.

An algorithm for systematic substring search is a two-part algorithm $A = (A_0, A_1)$:

- The *preprocessing algorithm* A_0 takes as input only the text, may perform arbitrary computation on it, and then outputs an S-bit "index" into the text.
- The *online algorithm* A_1 takes as input the index and the pattern, queries T bits of the text, and then outputs the location of the pattern in the text, if one exists.

Theorem 14. For any integer $N \in \mathbb{Z}_{>0}$ and integral constant $c > 2$, if there is an algorithm for systematic substring search on texts of length $cN\lceil \log N \rceil$ with pattern length $c\lceil \log N \rceil$ that uses an S-bit index and reads T-bits of the text in its online phase, then there exists a black-box algorithm for inverting functions $f : [N] \to [N]$ that uses S bits of advice and makes T online queries.

For any integer $N \in \mathbb{Z}_{>0}$, if there is a black-box algorithm for inverting functions $f : [2N] \to [2N]$ that uses S bits of advice and T queries, then, for any integral constant $c > 1$, there exists an algorithm for systematic substring search on texts of length N with pattern length $c\lceil \log N \rceil$ that uses an $\tilde{O}(S)$-bit index and reads $\tilde{O}(T)$ bits of the text in its online phase.