Online Convex Optimization & Complexity
Problem Set 6 – CS 6515/4540 (Fall 2025)

Solutions to HW6, Q26

26 Fine-Grained Complexity

Deterministic reduction B $\Rightarrow$ A. This direction is trivial: given an algorithm for A, to solve B just form the union $S := V \cup W$ (of size n), run the algorithm for A on S, and check whether the returned orthogonal pair (if any) is a cross pair. However, that simple approach may return a pair completely inside V or inside W. We avoid that by a tiny deterministic embedding that forbids same-set orthogonality while preserving cross orthogonality.

Construct for each $v \in V$ the vector

$$v' := (\, v,\ 1,\ 0\,) \in \{0,1\}^{d+2},$$

and for each $w \in W$ the vector

$$w' := (\, w,\ 0,\ 1\,) \in \{0,1\}^{d+2}.$$

Let $S' = \{v'_i\} \cup \{w'_j\}$ (size n) and run the Problem A algorithm on S'.

For any two distinct $v'_i, v'_{i'}$ (both from V) their inner product contains the contribution $1 \cdot 1$ from the $(d+1)$-th coordinate, hence $v'_i \cdot v'_{i'} \geq 1$ so they cannot be orthogonal. Similarly any two $w'_j, w'_{j'}$ have inner product ≥ 1 due to the $(d+2)$-th coordinate. But for a cross pair v'_i, w'_j the last two coordinates contribute $1 \cdot 0 + 0 \cdot 1 = 0$, so $v'_i \cdot w'_j = v_i \cdot w_j$. Thus v'_i and w'_j are orthogonal iff v_i and w_j are orthogonal. Therefore the A-algorithm on S' returns a cross orthogonal pair exactly when one exists in the original B instance.

Since the dimension increased only by 2, so an $O(n^{2-\delta} \cdot d)$-time algorithm for A still yields an $O(n^{2-\delta} \cdot (d+2)) = O(n^{2-\delta} \cdot d)$-time algorithm for B.

Randomized reduction A $\Rightarrow$ B. Given an algorithm for B, we show how to solve A. Let the input be $S = \{s_1, \ldots, s_n\}$.

We will randomly partition S into two equal-sized parts V and W (each of size $n/2$), run the B-algorithm on (V, W), and repeat the random partition a constant number of times. A true orthogonal pair in S is separated across the two parts with probability 1/2, so a constant number of repetitions suffices to get constant success probability.

1. Repeat constant number of times (e.g. 10 repetitions ensures a failure probability $\leq 2^{-10}$):
   (a) Choose a uniformly random partition of S into two sets V and W each of size $n/2$.
   (b) Run the B-algorithm on (V, W). If it returns an orthogonal cross pair, output it and halt.
2. If no repetition found a cross orthogonal pair, output "no orthogonal pair".

If S contains some orthogonal pair (s_i, s_j), then when we partition uniformly at random the pair is separated (one endpoint in V, the other in W) with probability exactly 1/2. Thus each repetition independently finds the pair with probability at least 1/2 (because conditioned on separation, the B-algorithm will detect it deterministically). After c repetitions the failure probability is at most 2^{-c}. Each repetition runs the B-algorithm on sets of size $n/2$, so the time per repetition is

$$T_B\big(n/2, d\big) = O\big((n/2)^{2-\delta} \cdot d\big) = O\big(n^{2-\delta} \cdot d\big),$$

and doing $O(1)$ repetitions preserves the $O(n^{2-\delta} \cdot d)$ running time.