ON EXPLOITING PSEUDO-LOCALITY OF INTERCHANGE DISTANCE

Avivit Levy

Shenkar College of Engineering and Design

Motivation: Studying String Metrics

String metrics in computational tasks:

- Similarity search and analysis
- Text editing
- Pattern matching
- Comparative genomics

Definitions: What are String Metrics?

Example: swap distance

$dist_{swap}(abcaa, baaca)$ =2

- Set of **operators**:

  OP={op1, op2,…}

- Distance:

$$dist(s_1, s_2) = min\{cost(o) | o \ over \ OP \ converts \ s_1 \ to \ s_2\}$$

Remark: In this paper we assume **UCM** (Unit-Cost Model)

Definitions: What are String Metrics?

- Set of **operators**:

  OP={op1, op2,...}

- Distance:

$$dist(s_1, s_2) = min\{cost(o) | o \text{ over } OP \text{ converts } s_1 \text{ to } s_2\}$$

Example: interchange distance

$dist_{int}(acbaa, baaca) = 2$

$dist_{int}(acbaa, cbaaa) = 2$

Definitions: Pseudo-Locality

First defined for
Period Recovery Problem
[AELPS, TALG 2012]

A string metric under UCM is **pseudo-local** if there is constant $c \geq 1$ s.t. for every strings s1, s2, if:

$$dist(s_1, s_2) = k$$

then:

$$k \leq H(s_1, s_2) \leq c \cdot k$$

where H is Hamming distance.

Example: interchange distance (c=2)

$$dist_{int}(abc, bca) = 2$$

$$2 \leq H(abc, bca) = 3 \leq 4$$

Definitions: Strong Pseudo-Locality

A string metric under UCM is **strong pseudo-local** if there is constant $c \geq 1$ s.t. for every strings s1, s2, if:

$$dist(s_1, s_2) = k$$

then:

$$H(s_1, s_2) = c \cdot k$$

where H is Hamming distance.

Example: swap distance (c=2)

$$dist_{swap}(abaa, baaa) = 1$$

$$H(abaa, baaa) = 2$$

Interchange Distance - Background

- Operator of comparison-based sorting algorithms
- Classical distance studied by **Cayley** in 1849
- **NP-hard** to compute on general strings even for binary strings [AHKLP, SICOMP 2009]
- Linear-time to compute on permutations, where each character appears once [AABLLPSV, JCSS 2009]
- 1.5-approximation in linear time [AHKLP, SICOMP 2009]

The Scope Problems

- **Approximate Nearest Neighbor Search:**

Given *n* *DB* *d*-dimensional vectors, $\varepsilon > 0$ and query vector *q*,
a $C(\varepsilon)$-ANN(*q*) is $a \in DB$ s.t. for every $b \in DB$

$$dist(q, a) \le C(\varepsilon) \cdot dist(q, b)$$

$n \gg d$

- **Approximate Pattern Matching:**

Given *m*-length P, $\varepsilon > 0$ and T of length *n*>*m*, output a $C(\varepsilon)$-approx. distance $dist$ between P and *m*-length substring of T for each position *i*.

Results: Interchange Distance

Main tool is ***pseudo-locality***

- **Approximate Nearest Neighbor Search:**

  Known: No known ANN DS for Interchange distance

  New: $(2+\varepsilon)$-ANN search data structure

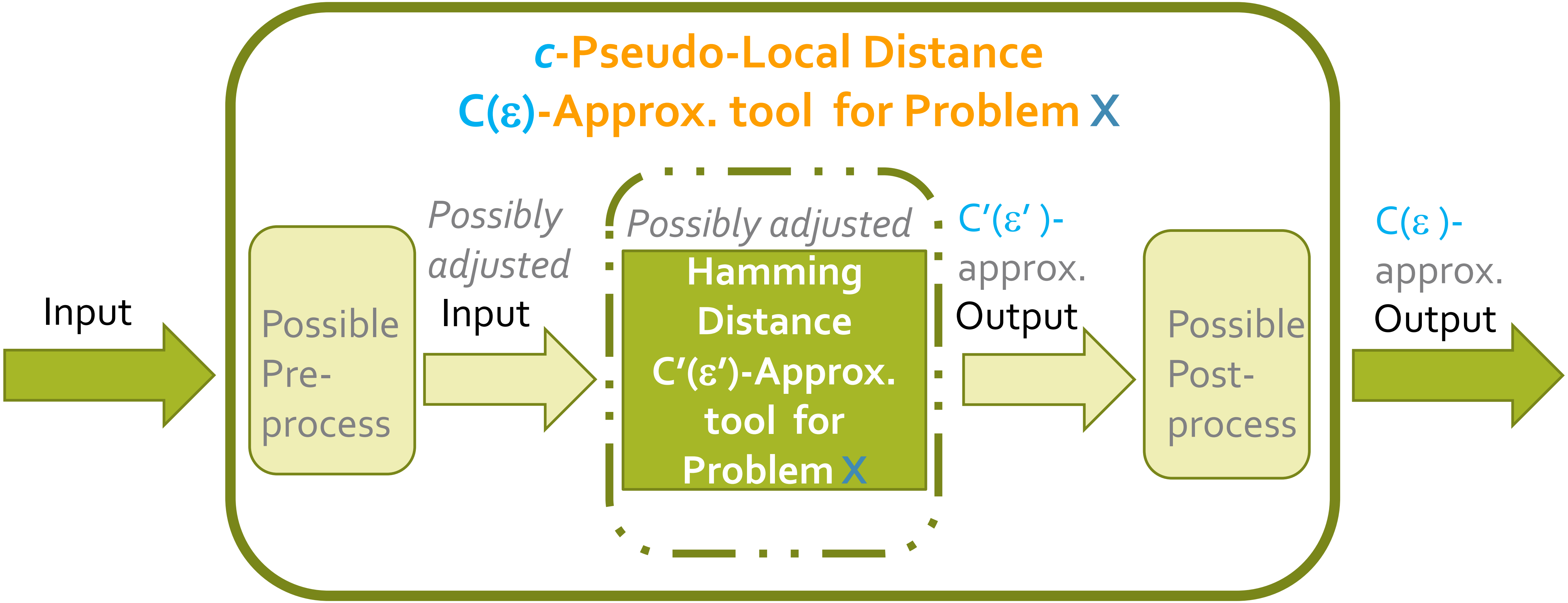
- **Approximate Pattern Matching:**

  Known: $\Theta(nm)$ algorithm giving 1.5 approximation

  New: $O(n)$ *randomized* algorithm giving $(2+\varepsilon)$-approximation

  $\tilde{O}(n)$ *determinitic* algorithm giving 2-approx. for fixed-size alphabets

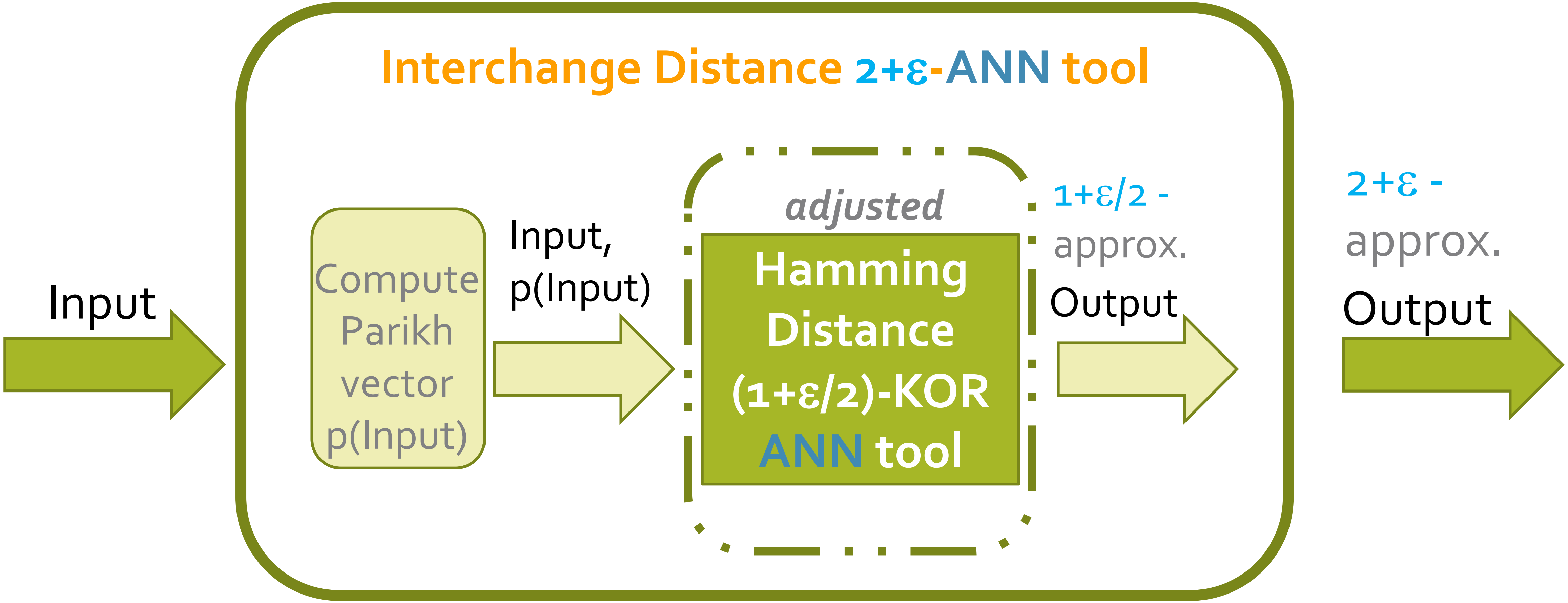
The Basic Idea



Transformation from C′(ε′)-approx. to C(ε)-approx. due to *c-pseudo-locality*

This work shows how to apply
for
Interchange Distance

Approximate Nearest Neighbor Search



KOR ANN Data Structure (binary vectors)

- **KOR Test:** β-Test τ

Randomly choose $C \subseteq \{1, \cdots, d\}$ with prob. β,

for each $i \in C$ randomly pick $r_i \in \{0,1\}$.

Define:

$$\tau(v) = \sum_{i \in C} r_i \cdot v_i \qquad (mod\ 2)$$

Property: For query q and a, b in DB s.t. $H(q,a) \leq \ell$, $H(q,b) > (1+\mathcal{E})\ell$, $\beta = \frac{1}{2\ell}$-test distinguishes between a and b with constant probability.

KOR ANN Data Structure (binary vectors)

- **KOR Data Structure:**

S has $S_1, \cdots, S_d$ substructures for each distance.

Each S_ℓ has $M = M(d, \mathcal{E}, \mu) = \tilde{O}(d)$ structures $T_1, \cdots, T_M$.

Each T_i has list $\mathrm{T} = \mathrm{T}(d, \mathcal{E}, \mu) = O(\log\log d)$ $\frac{1}{2\ell}$-tests $t_1, \cdots, t_{\mathrm{T}}$ and 2^{T}-size table for *DB* vectors results.

For a database vector *v*, its ***trace*** is the vector

$$t(v) = t_1(v), \cdots, t_{\mathrm{T}}(v) \in \{0,1\}^T.$$

KOR ANN Data Structure (binary vectors)

- **KOR Search Algorithm:** Given query q, binary search min distance ℓ, s.t. random T_i in S_ℓ has a *DB* point in table entry

$$t(q) = t_1(q), \cdots, t_{\mathrm{T}}(q).$$

If exists – search smaller ℓ, if not – search larger ℓ.

Property:

1. Prob. search uses structure T_i that fails at query at most μ.
2. If search doesn't fail at q, a in DB returned has $H(q,a) \leq (1+\mathcal{E})\Delta_H$, where $\Delta_H = \min_{v \in DB} H(q,v)$.

Adjusting KOR ANN Data Structure (take $\varepsilon' = \frac{\varepsilon}{c}$)

<u>Pseudo-local Query Condition</u>:

Let q be a query s.t. $\forall v \epsilon DB, dist(q, v) < \infty$.
If search doesn't fail at q,

a in DB returned has $dist(q, a) \leq (c + \mathcal{E})\Delta$,

where $\Delta = \min_{v \epsilon DB} dist(q, v)$.

Adjusting KOR ANN Data Structure (take $\varepsilon' = \frac{\varepsilon}{c}$)

Pseudo-local Query Condition proof:

Since $\forall v \epsilon DB, dist(q, v) < \infty$,

then by pseudo-locality $\Delta \leq \Delta_H \leq c \cdot \Delta$.

If $\ell < \Delta/(1+\varepsilon)$ then $\ell < \Delta/(1+\varepsilon') \leq \Delta_H/(1+\varepsilon')$

$\Rightarrow$ No *DB* point $\Rightarrow$ search on ℓ fails.

On the other hand, if $\ell \geq \Delta_H$ then search step on ℓ succeeds.

Thus, search ends with $\Delta_H/(1+\varepsilon') \leq \ell \leq \Delta_H$.

By **KOR-property**, a in DB returned has $H(q, a) \leq (1+\mathcal{E}')\Delta_H$

Thus, by pseudo-locality

$$dist(q, a) \leq c(1+\mathcal{E}')\Delta \leq (c+\mathcal{E})\Delta.$$

Adjusting KOR ANN Data Structure (take $\varepsilon' = \frac{\varepsilon}{c}$)

Infinite distances:
The condition - q is a query s.t. $\forall v \epsilon DB, dist(q, v) < \infty$ is crucial!

Example: Let DB = {a,b,c}, a = (0,1,0,1,0,0), b = (0,0,0,1,0,1), c = (0,0,0,0,1,0), and let q = (0,0,1,0,1,0).
Then, $\Delta_{swap} = d_{swap}(q, a) = d_{swap}(q, b) = 2$ and $H(q,a) = H(q,b) = 4$.
However, $\Delta_H = H(q, c) = 1$ and $d_{swap}(q, c) = \infty$.

Solution: Monitor infinite distances!

Problem: No guarantee on returned point!

Adjusting KOR ANN Data Structure (take $\varepsilon' = \frac{\varepsilon}{c}$)

Parikh Vector of a word (string or vector) over $\Sigma = \{a_1, \cdots, a_k\}$ is

$$p(w) = \left(\left|w_{a_1}\right|, \left|w_{a_2}\right|, \cdots, \left|w_{a_k}\right|\right),$$

where $\left|w_{a_i}\right|$ is the number of occurrences of the letter a_i in the word w.

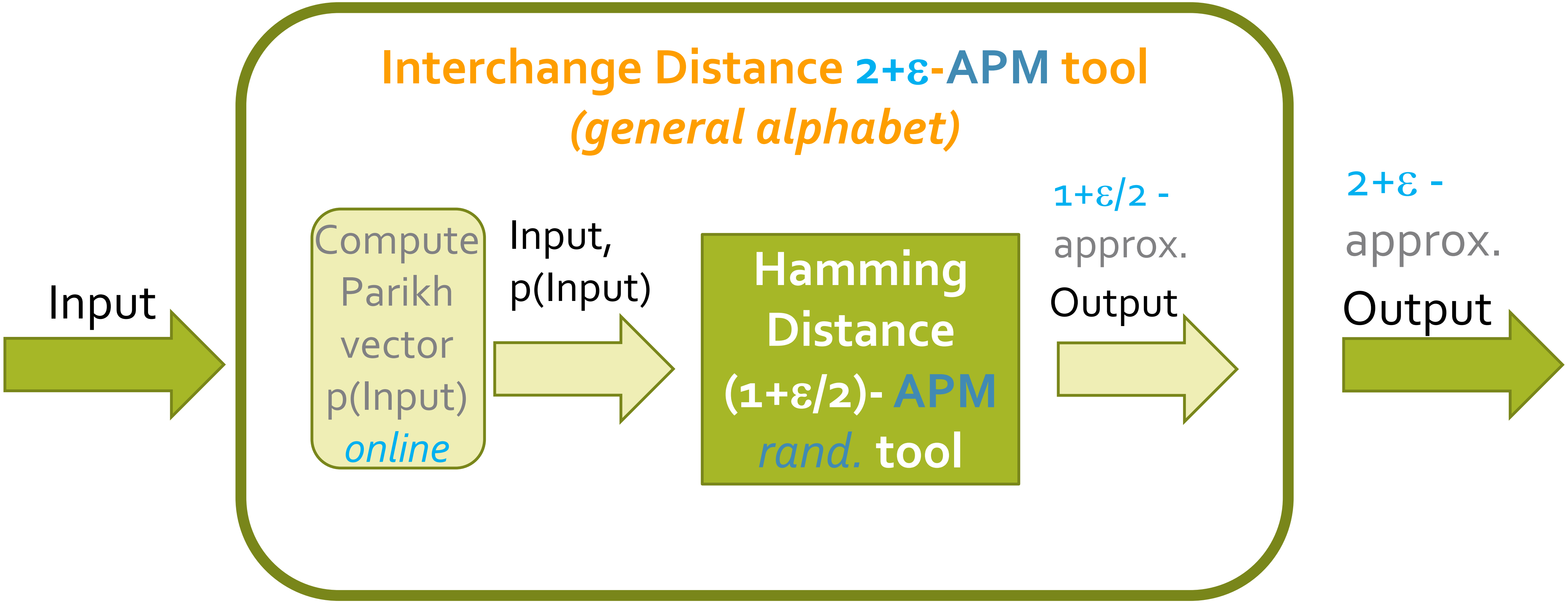
Infinity Check for Interchange Distance:

$d_{int}(a, b) < \infty$ **if and only** if $p(a) = p(b)$.

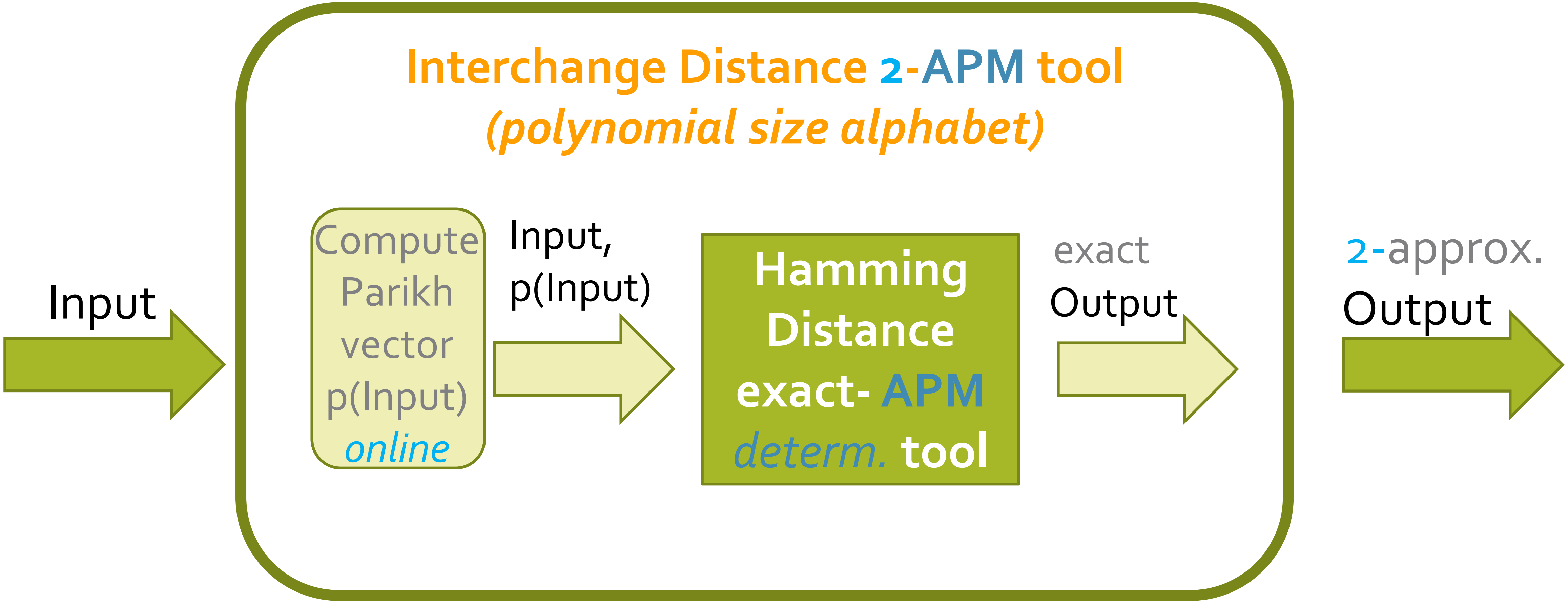
Monitoring infinite distances:

Idea – Split *DB* points by ***Parikh vector*** value and search only within *DB* points with the same Parikh vector value!

Approximate Pattern Matching



Approximate Pattern Matching



Space Efficient Histogram Online Computation

Easy: online computation of histogram in $\tilde{O}(m)$ additional space.

Impossible: online (rand.) comp. of histogram in $\tilde{o}(m)$ add. space.

New: online comp. of *highly accurate* histogram in $\tilde{O}\left(m^{2/3}\right)$ add. space.

Open Problems

- Can *LSH-based ANN* for Hamming distance be exploited to improve our ANN result?
- Can an infinite-distance check for *swap / parallel-interchange* be achieved to allow ANN DS?
- Can pseudo-locality be exploited for deriving new solutions in *other problems*?

Thank You !