# Efficient and effective methods for variant selection

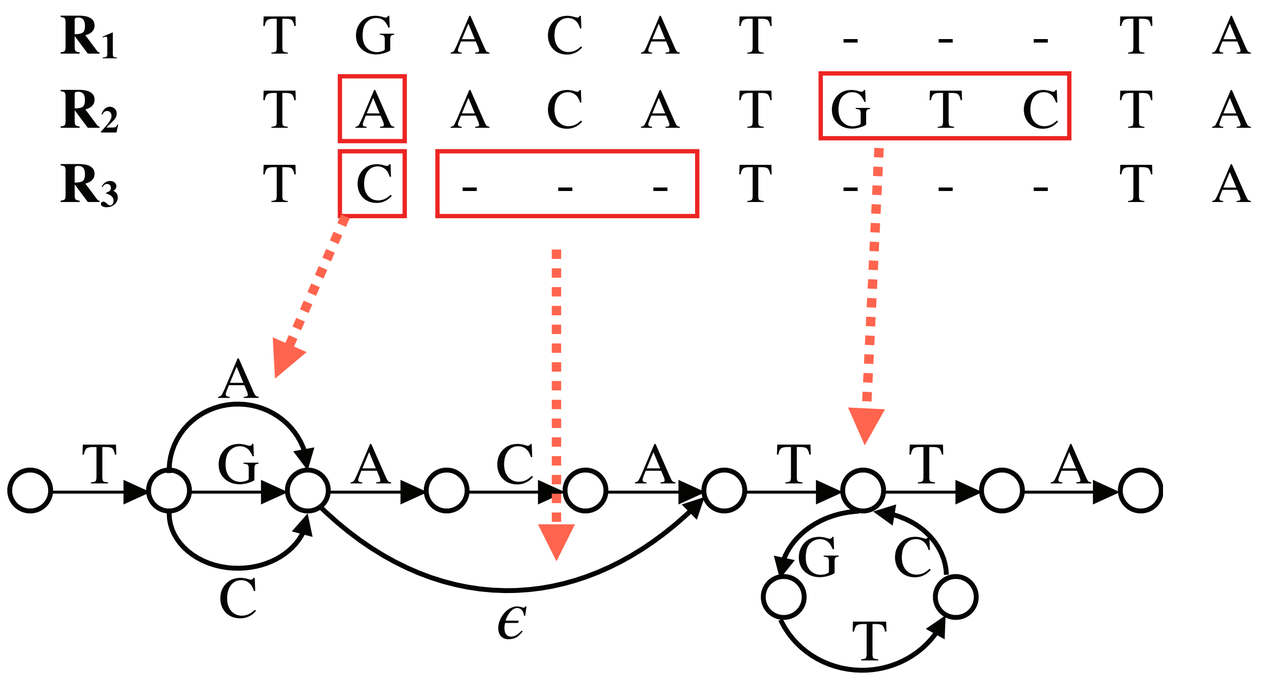
Srinivas Aluru (IDEaS, GaTech), Anne Benoit[1] (LIP, ENS de Lyon, IUF France)
**Bora Uçar**[1] (CNRS and LIP, ENS de Lyon)
(1) while visiting IDEaS





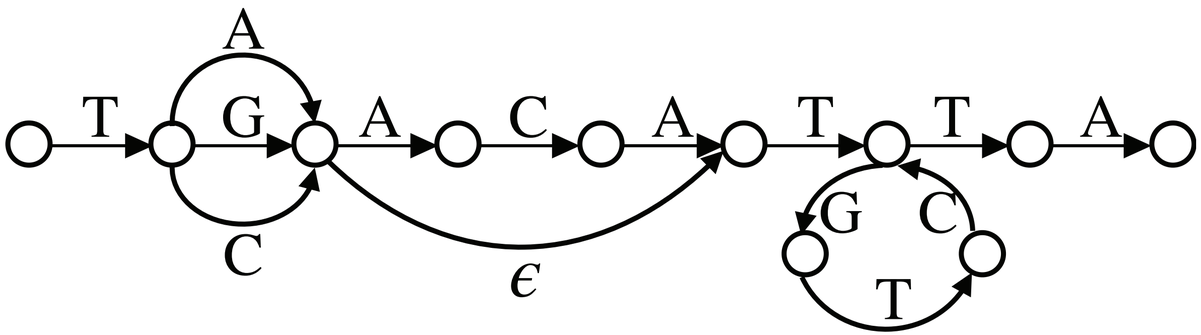
# I. Introduction

- Rapid sequencing of individuals enables representation of multiple observed haplotypes, catalog genetic diversity
- Led to graph representations (genome graphs) instead of a linear representation (string) to reduce reference bias and increase accuracy
- **Variation graphs**: a subclass that captures variations among individuals within a species or a target population



# I-Introduction: The variant selection problem

- In a **variation graph**, each haplotype is present as a path




- Combinatorial explosion: paths from two different haplotypes can be stitched together
  - yielding haplotypes that were not observed: diminishing returns and decline in accuracy as more and more variants are incorporated
  - sequence-to-graph alignement become costlier (with increased complexity of the graphs)

- Need to take/represent only a subset of known variations into account

➡The variant selection problem

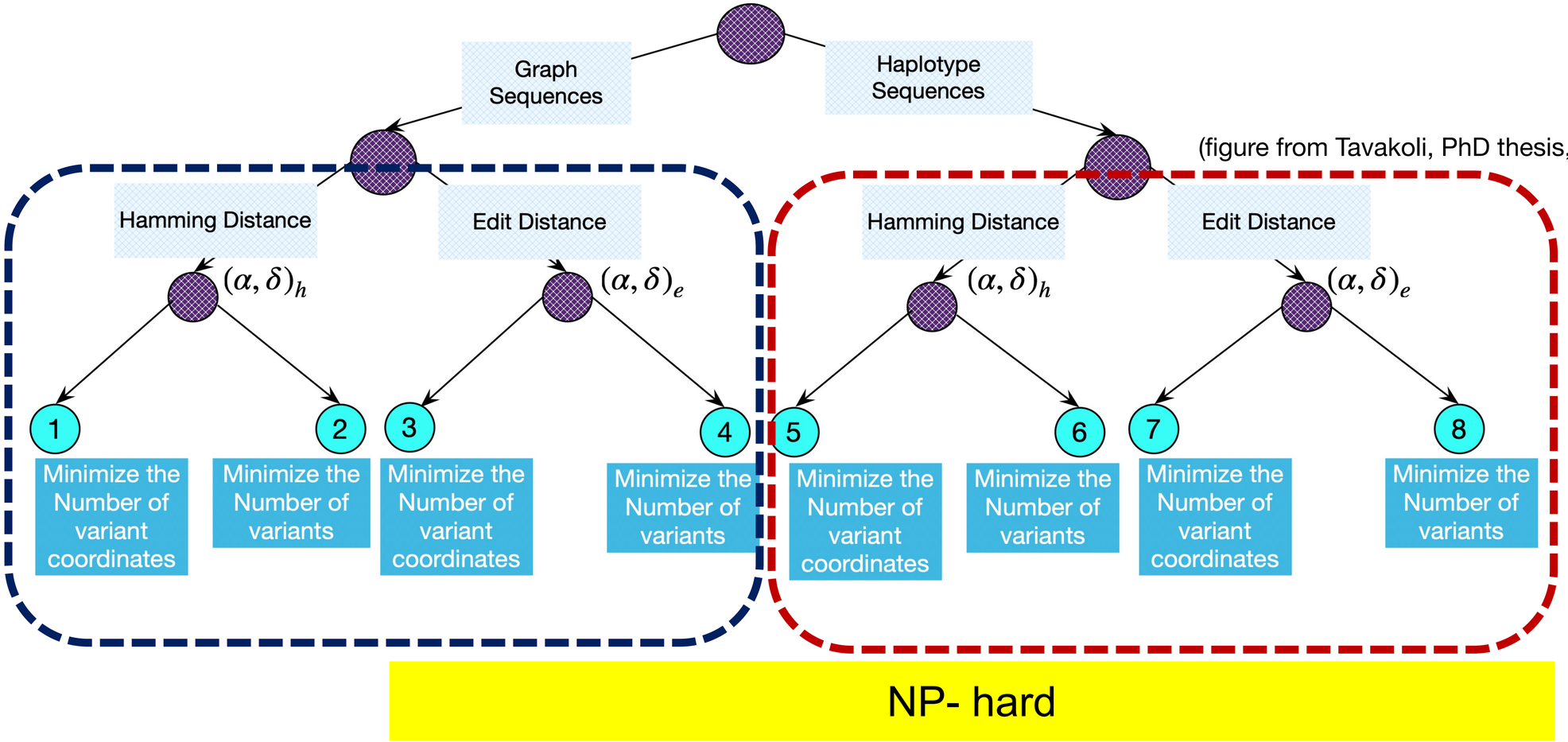
# I-What was available?

**Initial approches**: make use of specifics of the particular population or the context of the study; frequency of occurrence,…[Danek et al.'14, Siren et al.'14, Pritt et al.'18…]

**Recently**: Mathematical framework [Tavakoli, Jain, Gibney & Aluru] with desirable guarantees

- Given two parameters $\alpha$ and $\delta$, minimize the number of variants selected by retaining substrings of length up to $\alpha$ with at most $\delta$ differences
- Eight different problems: which substrings (only from the given haplotypes, the whole var. graph), which variants considered (SNPs, indels); which objective sought (num. variants, num. of variant loci);

# I-Mathematical framework: The eight problems




(figure from Tavakoli, PhD thesis, GaTech 2023)

NP- hard

An exact, poly-time algorithm for Problem 1; ILP solutions to others (ILP for Problem 2 is LP), [Tavakoli, Jain, Gibney & Aluru]

# I-Our contributions

- Revisit **Problems 1-4** and develop fast algorithms with provable guarantees:
  - novel graph-theoretic formulation
  - some problems solved optimally, some others approximately, comprehensive experiments with close-to-optimal results in all cases.

**Goal**: Fast methods with high-quality solutions can be incorporated into sequence-to-graph mappers

# II-Problem definition

- $R_1, R_2, \ldots, R_m$ are $m$ input haplotype sequences
- Represented with the **variation graph** $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \sigma)$, with edge-labels $\sigma$
- Edge-labels in $\Sigma \cup \{\varepsilon\}$, where $\Sigma = \{A, C, G, T\}$ and $\varepsilon$ is the empty character
- $R_1$, edge labels of a directed chain on $|R_1| + 1$ vertices in $\mathscr{G}$; augmented with variants from others:
  - An SNP variant: $\Sigma$-labeled edge between two adjacent coordinates.
  - A deletion variant: an $\varepsilon$-labeled edge between two coordinates (deletion-length apart).
  - An insertion variant: a loop of one or more labeled edges at a coordinate.

# II-Problem setting

- A path in $\mathscr{G}$ with $\alpha$ edges = a string of length $\alpha$
- A path visits a vertex at most once; twice if there is an insertion (one insertion at a site)
- The graph contains paths that are substrings of the original haplotypes, and other genotypes that are not necessarily in the input
- **The variant reduction framework** [Jain, Tavakoli, Aluru & Tavakoli, Gibney, Aluru]:

  **Goal**: reduce the graph $\mathscr{G} = (\mathscr{V}, \mathscr{E}, \sigma)$ to $\mathscr{G}' = (\mathscr{V}, \mathscr{E}', \sigma')$ by selectively removing edges and keeping the labels of the retained edges intact.

  **Constraint**: all $\alpha$-length strings corresponding to a path in $\mathscr{G}$ are mapped to the same starting vertex in $\mathscr{G}'$ with at most a user-specified error threshold $\delta$.

# II-Problem definition

- The **error threshold** is defined in terms of Hamming (SNP-only case) or edit distance (SNP-and-indels case).

- A reduced graph $\mathcal{G}'$ is $(\alpha, \delta)_h$-compatible if all $\alpha$-length paths in $\mathcal{G}$ can be mapped to their corresponding paths in graph $\mathcal{G}'$ with Hamming distance at most $\delta$. (*The number of positions that differ between an $\alpha$-length path in $\mathcal{G}$ and its corresponding path in $\mathcal{G}'$ is less than $\delta$.*)

- $(\alpha, \delta)_e$-compatible if all $\alpha$-length paths … an edit distance at most $\delta$. (*The number of insertions, deletions, and substitutions required to change an $\alpha$ -length path in $\mathcal{G}$ to its corresponding path in $\mathcal{G}'$ is less than $\delta$.*)

# II-Four problems of focus

**Problem 1**: For a variation graph $\mathcal{G}$ where all variations are SNPs, $\alpha$ and $\delta$, find $(\alpha, \delta)_h$ -compatible reduced graph $\mathcal{G}'$ with the minimum number of coordinates containing one or more variants.

**Problem 2**: For a variation graph $\mathcal{G}$ where all variations are SNPs, $\alpha$ and $\delta$, find $(\alpha, \delta)_h$ -compatible reduced graph $\mathcal{G}'$ with the minimum number variants.

**Problem 3**: For a variation graph $\mathcal{G}$, find $(\alpha, \delta)_e$-compatible reduced graph $\mathcal{G}'$ with the minimum number of coordinates containing one or more variants.

**Problem 4**: For a variation graph $\mathcal{G}$, find $(\alpha, \delta)_e$-compatible reduced graph $\mathcal{G}'$ with the minimum number variants.

At a site, remove all variations or none of them [Jain et al.]

# III-Graph formulation

- Undirected graphs derived from variation graphs using the parameter $\alpha$
- A total ordering on the vertices (coordinates):

  $ladj(v)$: neighbors of $v$ that the are before $v$; $hadj(v)$: neighbors of $v$ that are after $v$.

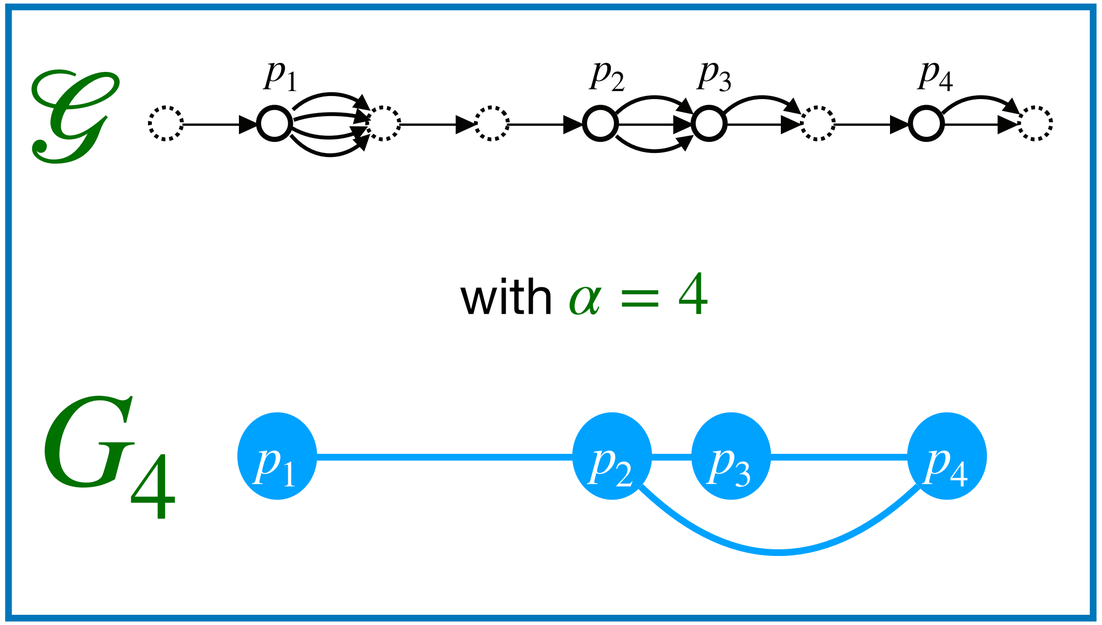
  Use with + to include $v$, for example $ladj^{+}(v)$.
- The graphs are interval graphs (is a proper subset of chordal graphs)
- Vertices have weights and another attribute called charge:
  $G_\alpha = (V, E, w, c)$ vertex weights $w(v)$ and charges $c(v)$

# III-The graph $G_\alpha = (V, E, w, c)$

Definition of $V$ and $E$; weight $w$ and charge $c$ depend on the problem

- $p_1, \ldots, p_n$ are the coordinates of the sites at which there are variants in the variation graph $\mathcal{G}$. A vertex in $G_\alpha$ for each $p_i$
- There is an edge $(v_i, v_j) \in E$, whenever $p_j$ is in $(p_i - \alpha, p_i]$ in the SNP-case

  (deletions: replace $p_i - \alpha$ by the left-most position $p_\ell$ that can reach $p_i$ with at most $\alpha$ edges with labels from $\Sigma$, see [Jain et al.])



# III-The problems on the graph $G_\alpha = (V, E, w, c)$

**Problems 1'-4'**: Given $G_\alpha = (V, E, w, c)$ and $\delta$, find a set of vertices $\mathcal{I}$ with the **maximum** weight $w(\mathcal{I})$, while the charge of $\mathcal{I}$ around each $v$, $c(ladj^+(v) \cap \mathcal{I})$, is at most $\delta$.

**Problem 1'**: $w(v_i) = c(v_i) = 1$.

**Problem 2'**: $w(v_i)$ is degree of $p_i$, $c(v_i) = 1$;

**Problems 3'-4'**: $w(v_i)$ as, resp., in Probs 1'-2'. $c(v_i)$ is an estimated upper bnd. on the edit distance if the insertion/deletion variants are removed at site $p_i$ (indep. from $p_j$).

Maximize to remove, translates to minimizing the retained variants in Probs 1-4.

# III-The nature of the graph problems

**Problems 1'-4'**: Given $G_\alpha = (V, E, w, c)$ and $\delta$, find a set of vertices $\mathcal{I}$ with the maximum weight $w(\mathcal{I})$, while the charge of $\mathcal{I}$ around each $v$, $c(ladj^+(v) \cap \mathcal{I})$, is at most $\delta$.

- For $\delta = 1$:
  - **Problem 1'**($w(v_i) = c(v_i) = 1$) is the maximum independent set (MIS) problem
  - **Problem 2'** ($w(v_i) \geq 1$, $c(v_i) = 1$) is the maximum weight independent set (MWIS) problem
- For $\delta > 1$: **Problems 1'- 2'** generalize, respectively, the MIS and MWIS problems
- **Problems 3'-4'** add another extension to the MIS and MWIS problems
- A solution is called a $\delta$-independent set

# IV-Algorithms

- Algorithms for **Problems 1'** and **2'** (poly-time solvable) and show their properties.

- Adapt them for **Problems 3'** and **4’**.

# IV-Greedy [Jain, Tavakali, Aluru]

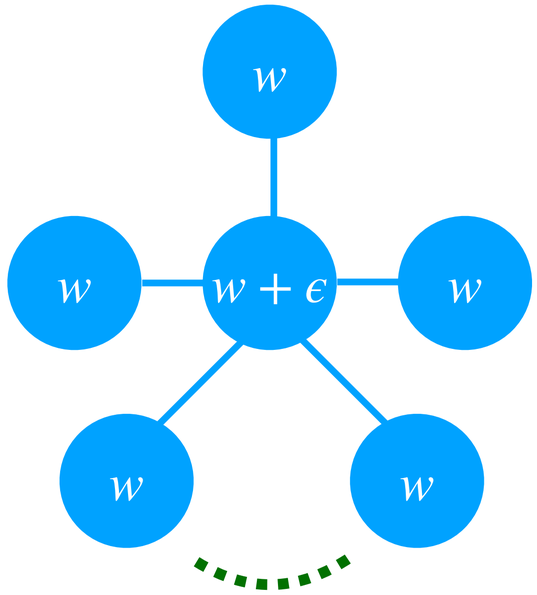
- **Greedy** for **Problem 1'**: visits vertices in the order of their intervals' starting position, and includes a vertex in the $\delta$-independent set $\mathcal{I}$ whenever doing so is feasible. (feasibility check $c(ladj(v) \cap \mathcal{I}) < \delta$).

# IV-GreedySort

- Inspired by the heuristics used for the MWIS problem

- **GreedySort** visits the vertices in a nonincreasing order of weights, breaking ties by the order of the vertices (smaller first), and includes a vertex into solution if doing so is feasible

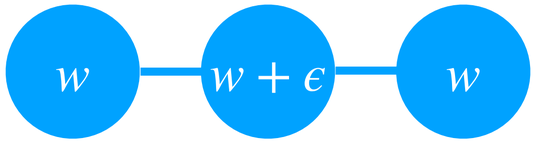
For MWIS in general, **GreedySort** is bad (**Problem 2'**, with $\delta = 1$)



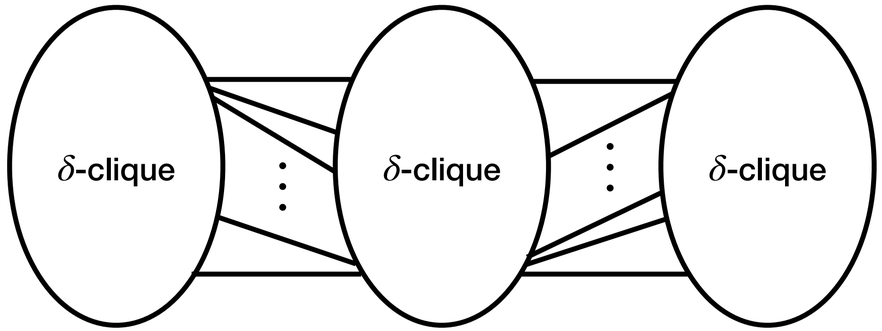
# IV-GreedySort

- Visits the vertices in a nonincreasing order of weights, breaking ties by the order of the vertices, and includes a vertex into solution if feasible.

- In our case (interval graph), **GreedySort** is 1/2 approximate for **Problem 2'** when $\delta = 1$




- **GreedySort** is 1/2 approximate for $\delta \geq 2$ for **Problem 2'**, and this bound is tight



# IV-hMWISd: A heuristic for max weight $\delta$-independent set

- A linear-time algorithm based on an existing algorithm [Frank'75] for the MWIS problem in chordal graphs
- Exact for **Problem 1'** (any $\delta$)
- Exact for **Problem 2'**, for $\delta = 1$

**Algorithm 1:** $\mathcal{I} = \text{hMWISd}(G, w, \delta)$

**Input:** $G$, an interval graph, vertices are presented in their order
**Input:** $w$, the weight of vertices of $G$
**Output:** $\mathcal{I}$, a maximum weighted $\delta$-independent set in $G$

```
ω ← w        /* copy w to ω, which is modified */
I ← ∅
mark(v) ← 0 for each vertex v
for v = 1 to n do                    /* in the given order */
    if ω(v) > 0 then
        mark(v) ← 1
        for u ∈ hadj(v) do
            ω(u) ← ω(u) − ω(v)/δ
neighsInI(v) ← 0 for all v
for v = n downto 1 do        /* in the reverse order */
    if mark(v) = 1 then
        if max_{u∈hadj(v)} (neighsInI(u)) < δ then
            I ← I ∪ {v}
            for u ∈ {v} ∪ hadj(v) do
                neighsInI(u) ← neighsInI(u) + 1
```

# IV-Other algorithms

- Another algorithm **hMWISdDec** that uses **hMWISd** $\delta$ times, by finding MWIS at a time
- A dynamic programming formulation, $O(n^{\delta})$; small $n$ (num vertices) or $\delta$ can be used for **Problems 1'-4'**.
- For **Problems 3'-4'**: update the feasibility check for including a vertex in the solution.

# V-Experiments

- We use the framework by [Jain, Tavakoli, & Aluru] which uses vcftools [Danecek et al.] to parse SNPs and indels from the 1000 Genomes Project data.

- The graphs of all 22 chromosomes (#vertices ~1.0M in chr22 to ~6.8M in chr1)

- We tested the algorithms with:
  - $\alpha = 150, 1000, 5000$, and $10000$ (average $|ladj|$= ~5.5, 30, 150, 300 in the SNP-only setting, slightly larger in the SNP-and-indels case)
  - $\delta = 0.01\alpha, 0.05\alpha, 0.1\alpha$ for each $\alpha$

  useful for short and long-read sequencing and low-to-high error tolerance [Jain et al.]

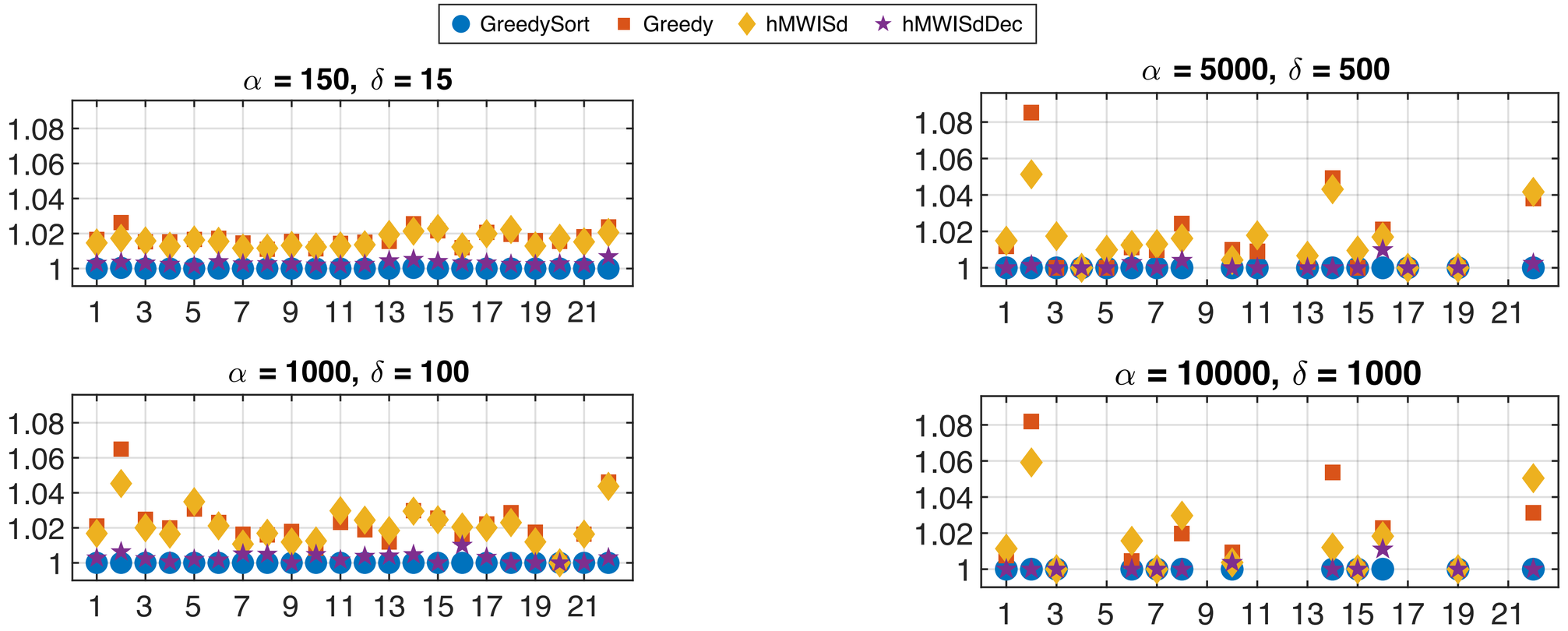
# V-Experiments (Problem 1)

**Problem 1**: Run time in seconds of various algorithms for $\alpha = 5000$ and $\delta = 100$ on the variation graphs for chromosomes 1, 11, and 22.

| Chr | Gurobi | Greedy | hMWISd | hMWISdDec | GreedySort |
|---|---|---|---|---|---|
| 1 | 3,944.554 | 0.100 | 1.985 | 11.110 | 2.386 |
| 11 | 2,948.298 | 0.063 | 1.408 | 5.646 | 1.500 |
| 22 | 963.736 | 0.018 | 0.397 | 0.982 | 0.414 |

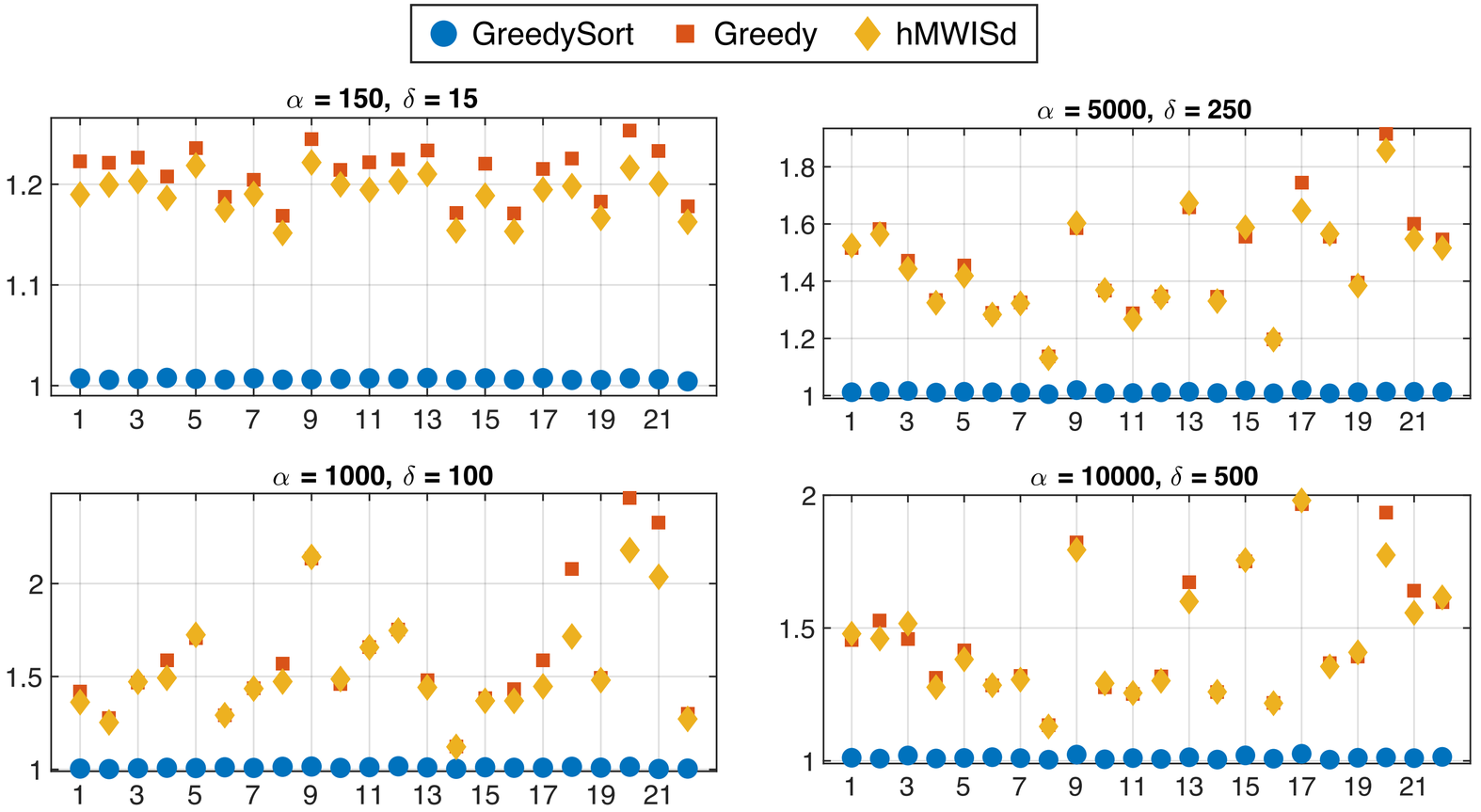
# V-Experiments (Problem 2)

$x$-axis: 22 chromosome graphs; $y$-axis: the ratio of the number of variants retained by an algorithm to that retained in an optimal solution (the lower the better performance)



# V-Experiments (Problem 4)

$x$-axis: 22 chromosome graphs; $y$-axis: the ratio of the number of variants retained by an algorithm to that retained in an optimal solution (the lower the better performance)




- On **Problem 3**, similar performance (the optimum vals are close)
- **GreedySort** 1.08 (worst case)
- Run time (secs) on the largest,
  - **GreedySort**: 11.25
  - ILP with Gurobi: 8498.1
- Challenging synth. instances:
  - **GreedySort**: 1.88
  - others goes beyond 10-20

# VI-Conclusion and future work

- Revisited four variant selection problems in a mathematical framework that allows guarantees
- Prior work: a practical heuristic without any performance guarantee, and ILP formulations
- Proposed three heuristics with provable properties for some problems, and a dyn. prog.
- Our heuristics are fast, robust, and deliver optimal or near-optimal results for all problems.

- Future work:
  - faster dynamic programming approach
  - haplotype-aware versions of the problems
  - other types of variations and genome graphs

- Codes are available:
  - https://github.com/ParBLiSS/Variant_Framework (original variant framework by Jain, Tavakoli, Aluru)
  - Our codes https://gitlab.inria.fr/bora-ucar/vf-graphs

Thank you!