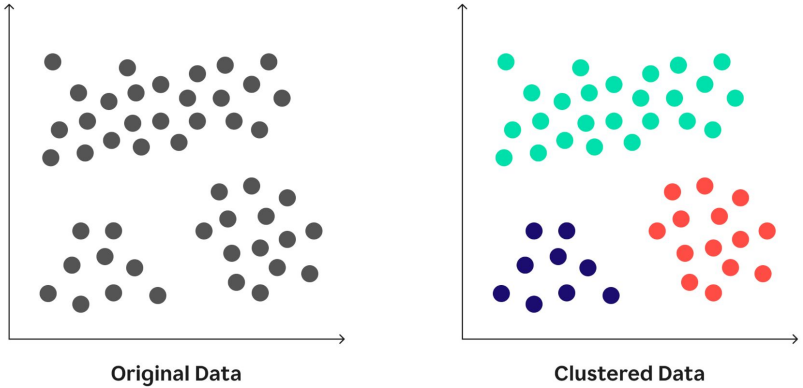
Implementation of a GA for gene clustering

Alberto **Catalano**, Alessandro **Delle Site**, Sofia **Pietrini**

Gene clustering

- Gene clustering aims to group genes with **similar expression patterns,** helping to uncover shared biological functions or regulatory mechanisms
- **Traditional clustering** methods (like K-means) can struggle with complex, high-dimensional gene expression data
- **GA** can be a robust alternative thanks to the ability to **navigate complex** search spaces

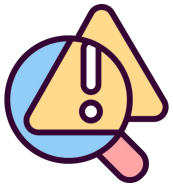


Evolution of the work & issues encountered

- Compare GA results with other bio-inspired AI techniques (ACO)
- Test the GA on several benchmark datasets

- Initial definition of the evolutionary operators
- Datasets: Khan, Yeast and Serum
- Choice of the distance

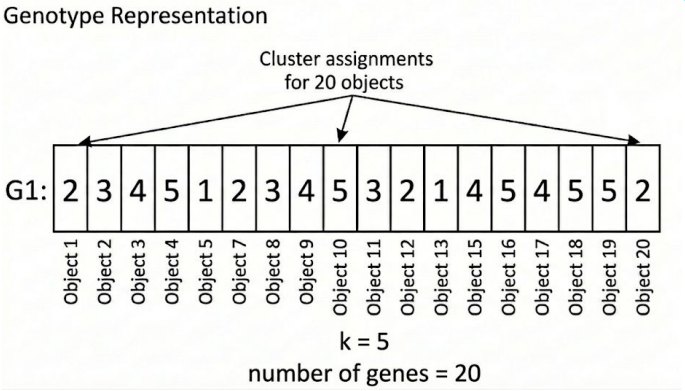
- Optimal GA parameters and strategies
- Final datasets: filtered TCGA with corresponding signature genes data
- Comparison with K-means



Many studies do not provide details on dataset creation, so they **cannot be used as a benchmark**, which was the initial plan

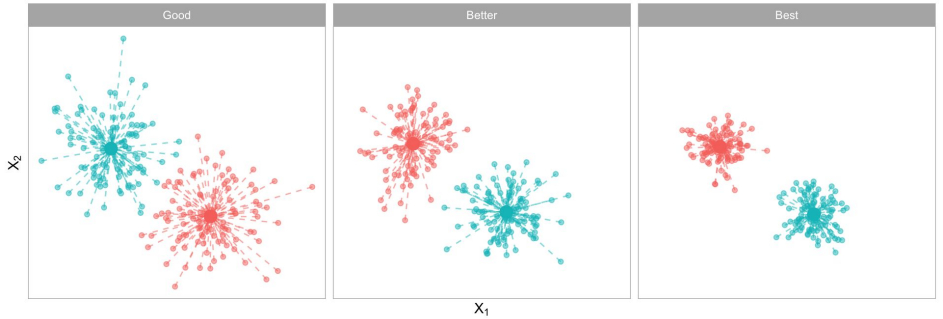
GA implementation

- Genetic representation: **integer array**
- Fitness function based on **TWCV**
- Similarity metric: **Pearson correlation**
- **Legality check**: ensures no empty clusters



$$TWCV = \sum_{k=1}^{K} \sum_{x \in C_k} (1 - \text{similarity}(x, \mu_k))$$

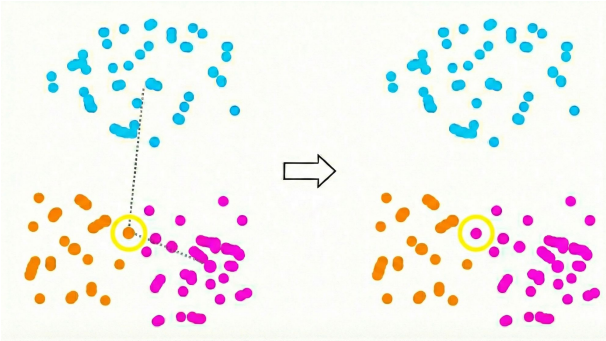
$$\text{Fitness} = 1.5 * \text{TWCV}_{\text{max}} - \text{TWCV}_{\text{individual}}$$



https://bradleyboehmke.github.io/HOML/kmeans.html

GA implementation: Mutation

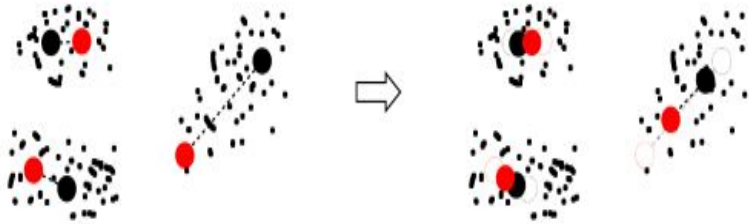
- Computes the distance between the selected point and each centroid
- The point is significantly more likely to be reassigned to a **geometrically closer cluster**



```
# Calculates distance from all the current centroids
for k in range(self.n_clusters):
    centroid = self.centroids[k]
    if centroid is None:
        # If the cluster is empty, distance = max distance
        # (1.0 for cosine similarity)
        d_vals.append(1.0)
    else:
        sim = np.dot(point, centroid)
        dist = 1.0 - sim
        d_vals.append(dist)

d_vals = np.array(d_vals)
d_max = np.max(d_vals)

# Invert distance: closer clusters get higher probability
numerator = 1.5 * d_max - d_vals + 1e-6

# Clip negative values to avoid crashes
numerator = np.maximum(numerator, 0.0)

total = np.sum(numerator)
if total == 0:
    prob_distribution = np.ones(self.n_clusters) / self.n_clusters
else:
    # Normalization to obtain the probability
    prob_distribution = numerator / total
```

GA implementation: Crossover

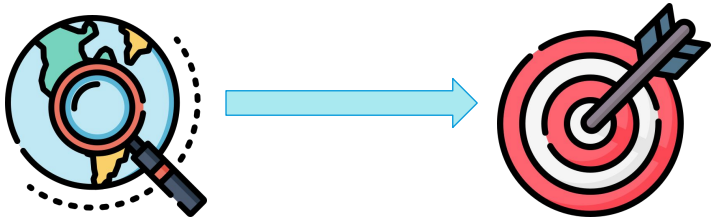
- **Geometric crossover** that works in the **centroid** space
- **Offspring** created by **linear interpolation** using a mixing parameter α and adding **gaussian noise**



```python
class GeneticClustering:
    def crossover(self, other, crossover_probability, alpha, noise_level):

        # Calculate similarity matrix between all the centroids
        mat_c1 = np.array(c1)
        mat_c2 = np.array(c2)

        sim_matrix = np.dot(mat_c1, mat_c2.T)
        # Compute distance matrix
        dist_matrix = 1.0 - sim_matrix

        # Interpolation and noise
        new_centroids = [None] * self.n_clusters
        for idx1, idx2 in matched_pairs:
            base_c = alpha * c1[idx1] + (1 - alpha) * c2[idx2]

            if noise_level > 0:
                noise = np.random.normal(0, noise_level, size=base_c.shape)
                base_c += noise
```

GA implementation: algorithm strategy

- Selection: **Roulette Wheel** Selection + **Elitism**
- Adaptive **mutation rate**: 0.01 → 0.001
- Adaptive **geometric crossover rate**: 0.85 → 0.65
- **Gaussian noise**: decreases to 0



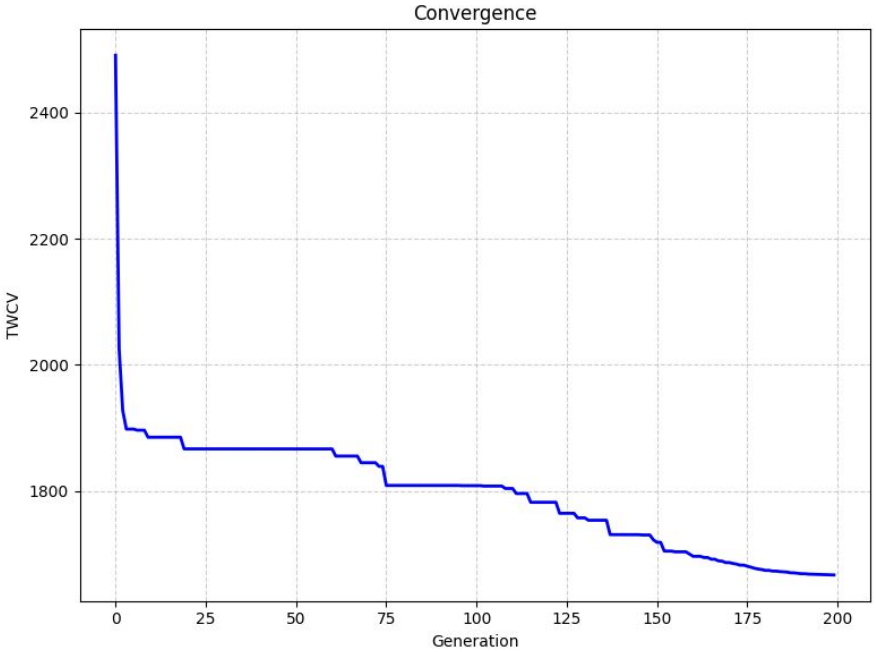
Datasets and data preprocessing

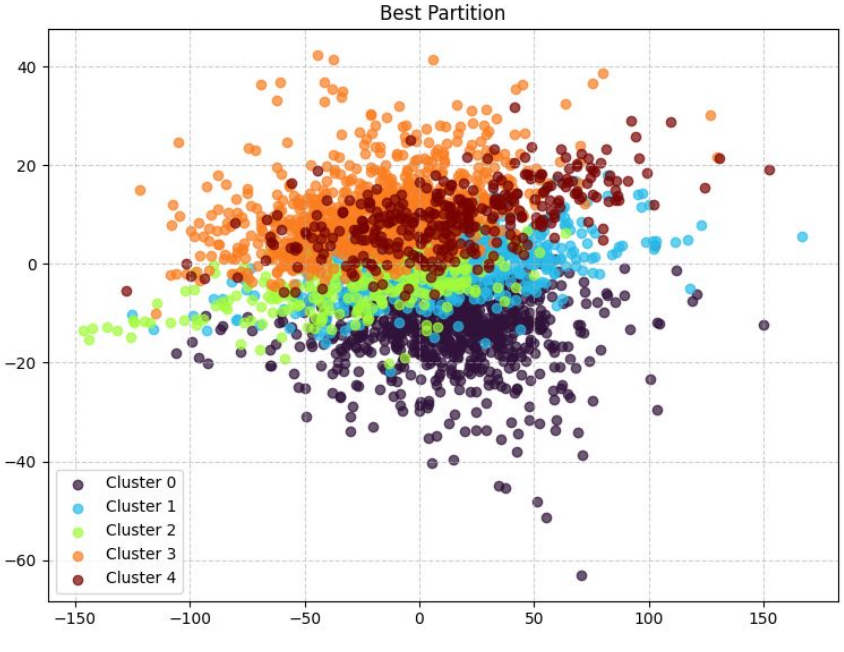
Preprocessing:

- Row-wise standardization (**Z-score**)
- **L2 normalization**

Dataset Name	Tumor Type/Tissue	Number of Signatures	Expected Number of Clusters
BRCA	Breast	PAM50 – 50	k = 5
CRC	Colorectal	CMS – 40	k = 4
GBM	Glioblastoma	Verhaak – 15	k = 3
LUAD	Lung	Wilkerson – 60	k = 3

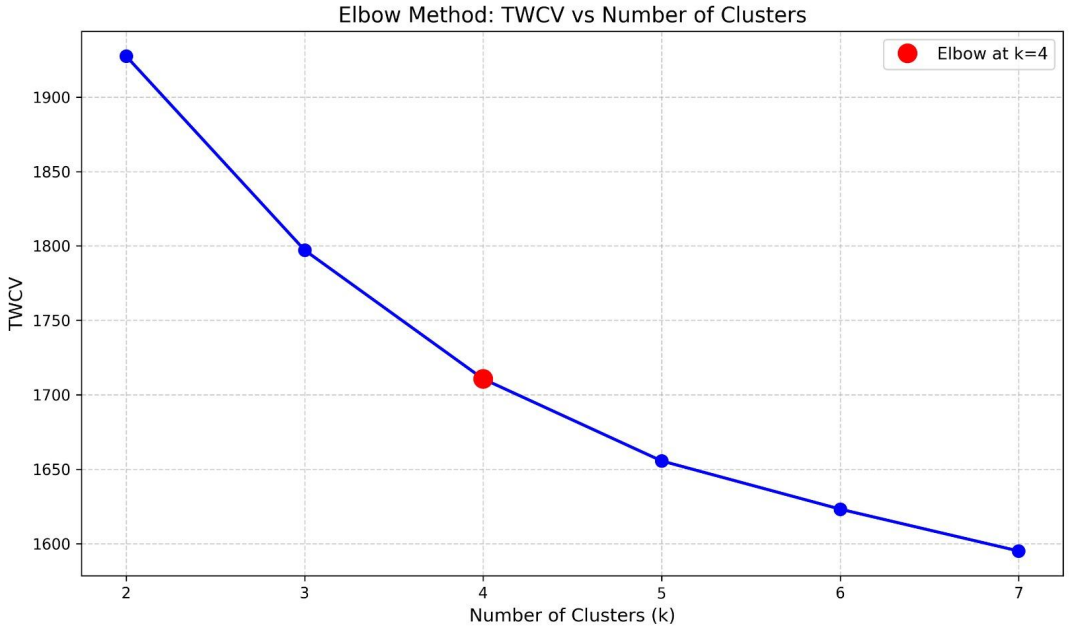
BRCA clustering results



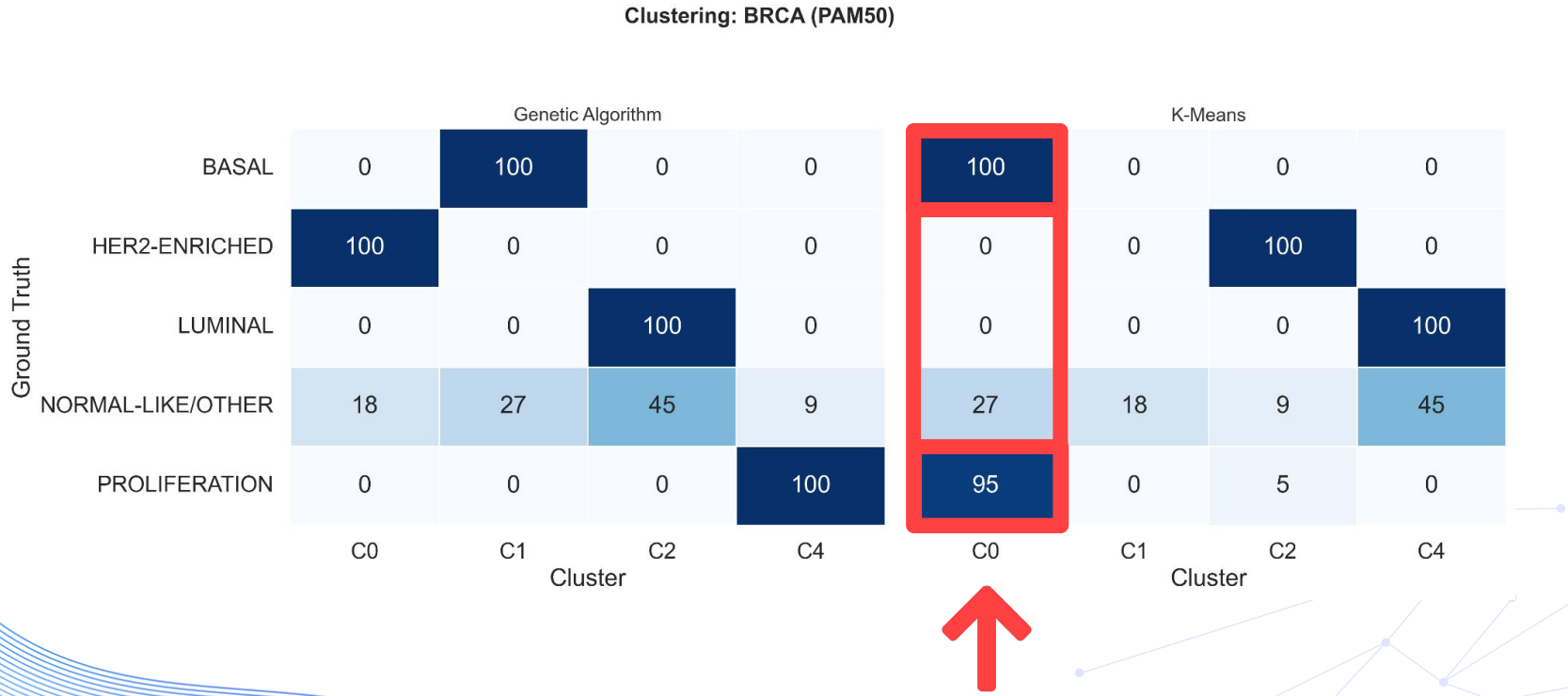


Elbow method

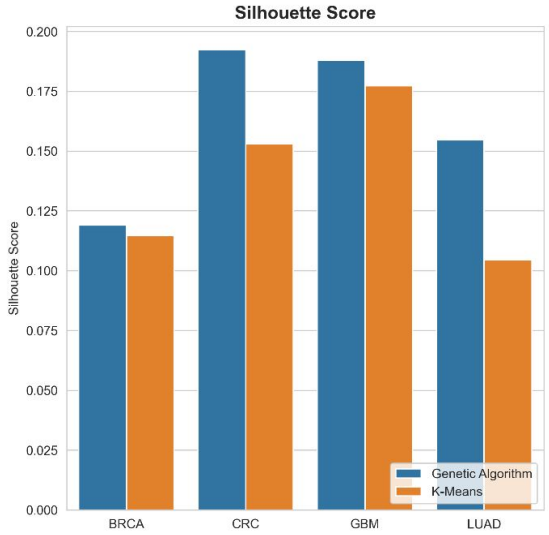
- Elbow method predicted a mathematical optimum of k=4, despite the fixed biological target of k=5
- This could be due to one of the PAM50 subtype lacking distinct cohesion (*Normali-like/Other)*

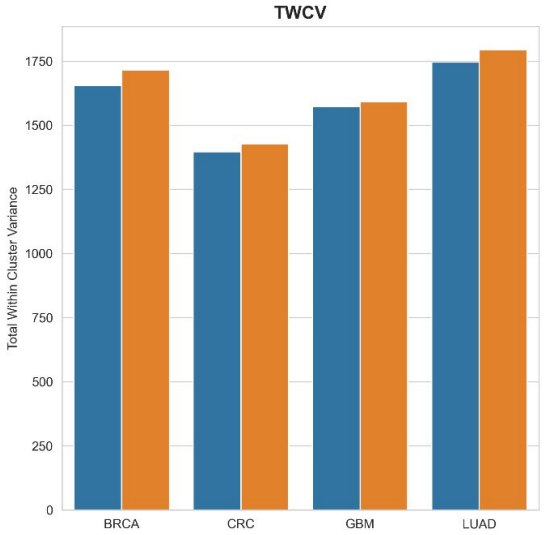


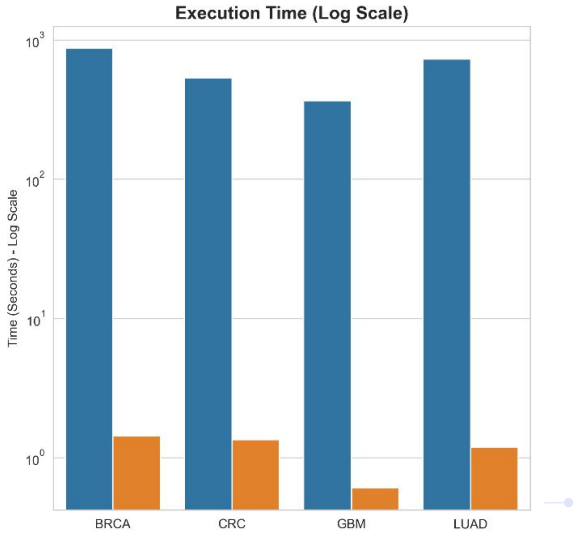
GA vs K-means: biological fidelity



GA vs K-means: metrics







Conclusion

- Developed and validated a GA designed to handle the challenges of clustering **high-dimensional** gene expression data
- The GA achieved perfect classification purity for **four out of the five** PAM50 molecular subtypes, with similar performance across other datasets tested.
- The **high precision** comes at a significant **computational cost**, with runtimes that are orders of magnitude longer than those of K-means

Future ideas

- Updating the framework to a **hybrid model**, using PSO to further optimize the algorithm's hyperparameters
- Transition to **single-cell analysis** to study tumor heterogeneity
- Shift to **multi objective optimization**
  - TWCV → intra-cluster compactness
  - Silhouette → inter-cluster separation

Thank you!