StableMDS: A Novel Gradient Descent-Based Method for Stabilizing and Accelerating Weighted Multidimensional Scaling

WASEDA University

Zhongxi Fang†, Xun Su†, Tomohisa Tabuchi†, Jianming Huang†, Hiroyuki Kasai†

†Waseda University, Tokyo, Japan

AISTATS

Problem of Interest: Weighted Multidimensional Scaling (WMDS)

- **Goal:** Given pairwise distances d_{ij} and weights w_{ij}, find low-dimensional embeddings $\mathbf{y}_1, \dots, \mathbf{y}_n \in \mathbb{R}^p$ that preserve the structure of the original space.
- **Optimization Problem:**

$$\min_{\mathbf{Y}\in\mathbb{R}^{n\times p}} \left\{ S(\mathbf{Y}) \coloneqq \sum_{i<j} \underbrace{w_{ij}\left(\|\mathbf{y}_i - \mathbf{y}_j\| - d_{ij}\right)^2}_{f_{ij}(\mathbf{Y})} \right\}$$

- **Use Cases:** Dimensionality reduction and graph drawing.
- **Key Challenges:**
  - Stability: While line search effectively addresses the step-size selection challenge and improves robustness, it comes at a significant computational cost due to multiple objective function evaluations per update: $\mathcal{O}(kn^2)$ per iteration, where k is the number of evaluations during line search.
  - Efficiency: Stress Majorization guarantees that the objective function value is non-increasing, but it is computationally expensive: $\mathcal{O}(\max(n^3, n^2p))$ per iteration.

Contributions

- Propose *StableMDS*, a novel gradient descent-based method that ensures convergence to stationary points and guarantees non-increasing stress values during optimization.
- Improve computational efficiency over Kamada-Kawai and Stress Majorization methods.
- Propose *FastMDS*, an accelerated variant of *StableMDS* that empirically achieves stable convergence on large-scale datasets.

Proposed Methodology: Key Steps & Proof Sketch

The procedure is simple, but the key challenge is Step 1: proving for the first time that the Hessian of WMDS with respect to each $\mathbf{y}_i$ admits a spectral bound. While WMDS has been well studied, this property was not previously known.

❶ **Hessian Bound.** For each pairwise term $f_{ij}(\mathbf{Y})$:

$$f_{ij}(\mathbf{Y}) \coloneqq w_{ij}\left(\|\mathbf{y}_i - \mathbf{y}_j\| - d_{ij}\right)^2,$$

we compute the second derivative with respect to $\mathbf{y}_i$:

$$\nabla^2_{\mathbf{y}_i} f_{ij} = 2w_{ij}(a\,\mathbf{I} + b\,\mathbf{v}\mathbf{v}^\top) \preceq 2w_{ij}\mathbf{I},$$

where $\mathbf{I}$ is the identity matrix, $a \coloneqq 1 - \frac{d_{ij}}{\|\mathbf{y}_i-\mathbf{y}_j\|}$, $b \coloneqq \frac{d_{ij}}{\|\mathbf{y}_i-\mathbf{y}_j\|}$, and $\mathbf{v} \coloneqq \frac{\mathbf{y}_i-\mathbf{y}_j}{\|\mathbf{y}_i-\mathbf{y}_j\|}$. Summing over $j \neq i$, we obtain the bound for the objective function:

$$\nabla^2_{\mathbf{y}_i} S(\mathbf{Y}) \preceq L\mathbf{I}, \quad L = 2\sum_{j\neq i} w_{ij}.$$

❷ **Descent Lemma.** The bound $\nabla^2_{\mathbf{y}_i} S \preceq L\mathbf{I}$ implies that for any update Δ_i:

$$S(\mathbf{Y} + \Delta_i) \le S(\mathbf{Y}) + \langle \nabla_{\mathbf{y}_i} S, \Delta_i \rangle + \frac{L}{2}\|\Delta_i\|^2.$$

This shows that the objective function decreases after each update.

❸ **Optimal Step Size.** Setting $\Delta_i = -\frac{1}{L}\nabla_{\mathbf{y}_i} S$ ensures:

$$S(\mathbf{Y}^{(t+1)}) \le S(\mathbf{Y}^{(t)}) - \frac{1}{2L}\|\nabla_{\mathbf{y}_i} S(\mathbf{Y}^{(t)})\|^2,$$

guaranteeing that the objective decreases at each step.

❹ **StableMDS Algorithm.** The update rule for each $\mathbf{y}_i$ is:

$$\mathbf{y}_i^{(t+1)} = \mathbf{y}_i^{(t)} - \frac{1}{L}\nabla_{\mathbf{y}_i} S(\mathbf{Y})\Big|_{\mathbf{y}_i=\mathbf{y}_i^{(t)}},$$

which ensures monotonic decrease in the objective function.

❺ **Convergence Rate.** StableMDS reaches an ϵ-stationary point in $\mathcal{O}(1/\epsilon^2)$ steps.

❻ **FastMDS Extension.** By using a mini-batch $\mathcal{B}$ of size $b \ll n$, the per-step cost is reduced from $\mathcal{O}(n^2p)$ to $\mathcal{O}(bnp)$, enabling faster computation for large datasets.

Source code

Source code is available at: https://github.com/Fzx-oss/StableMDS.

Comparison of WMDS Solvers

	Kamada-Kawai Algorithm	Stress Majorization	StableMDS (**Ours**)
Computational Complexity per Iteration	$\mathcal{O}(n^2p^3)$ if $p \ge 3$ $\mathcal{O}(n^2p^2)$ if $p = 2$	$\mathcal{O}(\max(n^3, n^2p))$	$\mathcal{O}(n^2p)$
Guarantee of Non- Increasing Loss Values	✗	✓	✓
Convergence Guarantee	Requires close to a stationary point	Stationary point	Stationary point
Convergence Rate	Quadratic (Newton) or superlinear (quasi-Newton)	Unknown	Sublinear
Auxiliary Space	$\mathcal{O}(p^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(p)$

StableMDS (Alg.1)

Require: Number of iterations T, initial positions of embeddings $\mathbf{Y}^{(0)}$, distance matrix $\mathbf{D}$, weighting matrix $\mathbf{W}$, boolean variable `shuffle`.
Ensure: Updated embeddings $\mathbf{Y}^{(nT)}$

1: Initialize the indices list indices $= [1, \dots, n]$, where n is the number of points.
2: **for** $t = 1$ to T **do**
3: **if** `shuffle` **then**
4: Shuffle indices to randomize the order of updates.
5: **end if**
6: **for** index **in** indices **do**
7: Set i to the current index.
8: Compute the step size: $\eta_i = \frac{1}{\sum_{j=1,j\neq i}^{n} w_{ij}}$.
9: Compute the gradient:
$\mathbf{g}_i = \sum_{j=1,j\neq i}^{n} w_{ij}\left(\mathbf{y}_i^{(t)} - \mathbf{y}_j^{(u)}\right)\left(1 - \frac{d_{ij}}{\max(\epsilon,\|\mathbf{y}_i^{(t)}-\mathbf{y}_j^{(u)}\|)}\right)$
10: where $\mathbf{y}_j^{(u)} = \begin{cases}\mathbf{y}_j^{(t+1)} & \text{if } \mathbf{y}_j \text{ has been updated}\\ \mathbf{y}_j^{(t)} & \text{otherwise.}\end{cases}$
11: Update the embedding: $\mathbf{y}_i^{(t+1)} = \mathbf{y}_i^{(t)} - \eta_i\mathbf{g}_i$.
12: **end for**
13: **end for**

*Differences for FastMDS relative to StableMDS are highlighted in color.

FastMDS (Alg.2)

Require: Number of iterations T, initial positions of embeddings $\mathbf{Y}^{(0)}$, distance matrix $\mathbf{D}$, weighting matrix $\mathbf{W}$, sampling size b, boolean variable `shuffle`.
Ensure: Updated embeddings $\mathbf{Y}^{(nT)}$

1: Initialize the indices list indices $= [1, \dots, n]$, where n is the number of points.
2: **for** $t = 1$ to T **do**
3: **if** `shuffle` **then**
4: Shuffle indices to randomize the order of updates.
5: **end if**
6: Uniformly sample a set of data points $\mathcal{B} \subset \{1, \dots, n\}$, with a size of $b = |\mathcal{B}|$.
7: **for** index **in** indices **do**
8: Set i to the current index.
9: Compute the step size: $\eta_i = \frac{1}{\sum_{j\in\mathcal{B},j\neq i} w_{ij}}$.
10: Compute the gradient:
$\mathbf{g}_i = \sum_{j\in\mathcal{B},j\neq i} w_{ij}\left(\mathbf{y}_i^{(t)} - \mathbf{y}_j^{(u)}\right)\left(1 - \frac{d_{ij}}{\max(\epsilon,\|\mathbf{y}_i^{(t)}-\mathbf{y}_j^{(u)}\|)}\right)$,
11: where $\mathbf{y}_j^{(u)} = \begin{cases}\mathbf{y}_j^{(t+1)} & \text{if } \mathbf{y}_j \text{ has been updated}\\ \mathbf{y}_j^{(t)} & \text{otherwise.}\end{cases}$
12: Update the embedding: $\mathbf{y}_i^{(t+1)} = \mathbf{y}_i^{(t)} - \eta_i\mathbf{g}_i$.
13: **end for**
14: **end for**

Numerical Evaluations

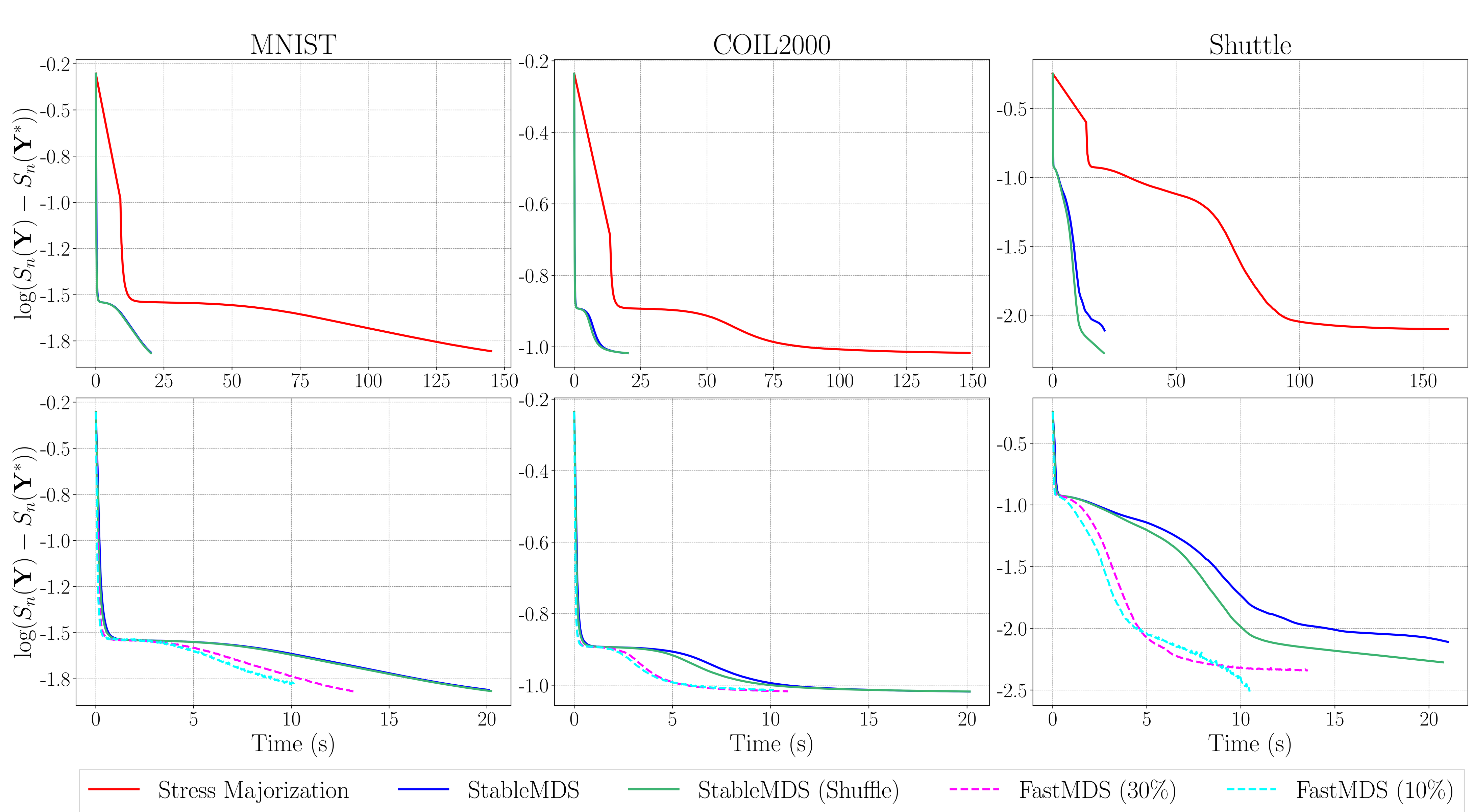

Figure 1. Optimization loss curves on the `MNIST`, `COIL2000`, and `Shuttle` datasets (3,000 samples each)

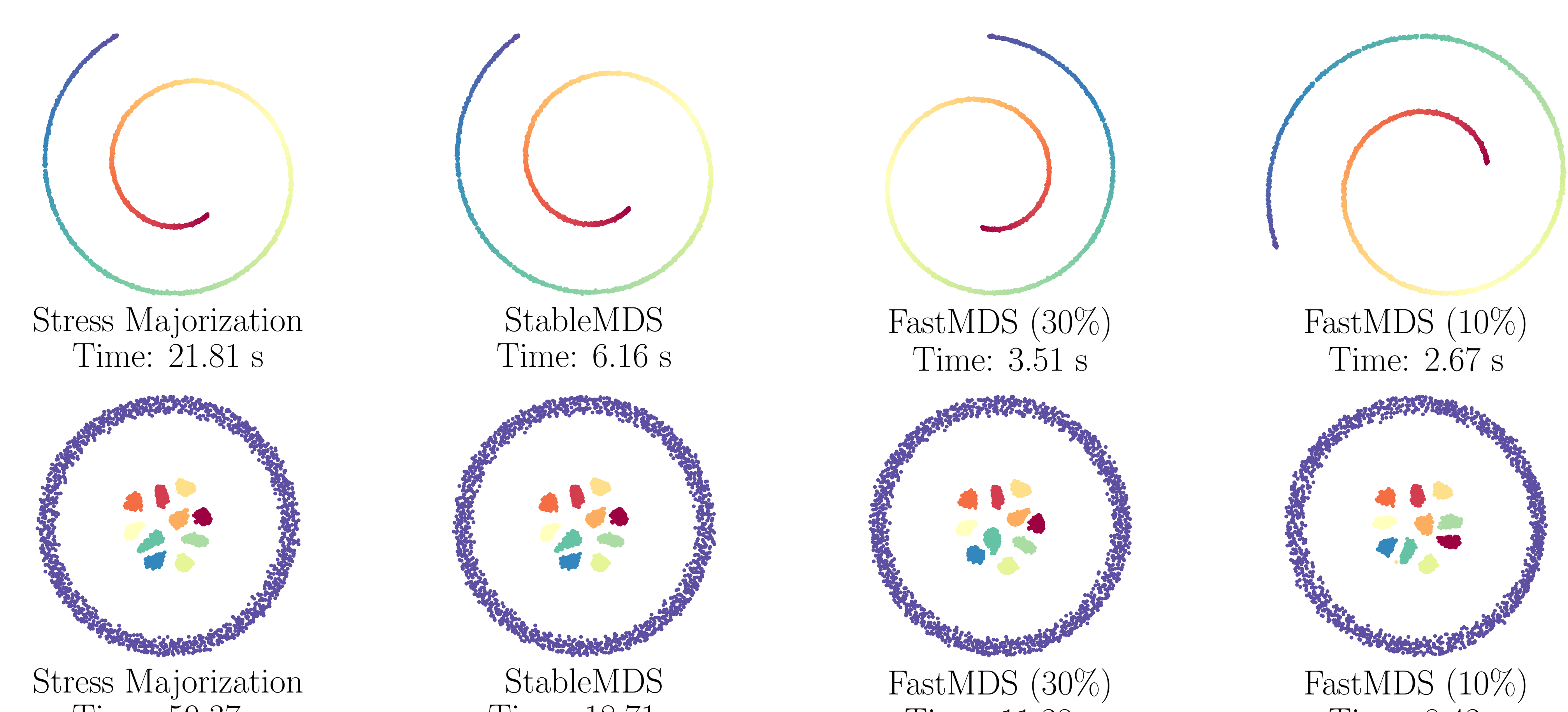

Figure 2. Dimensionality reduction results on the `Swiss Roll` and `Spheres` datasets (3,000 samples each)

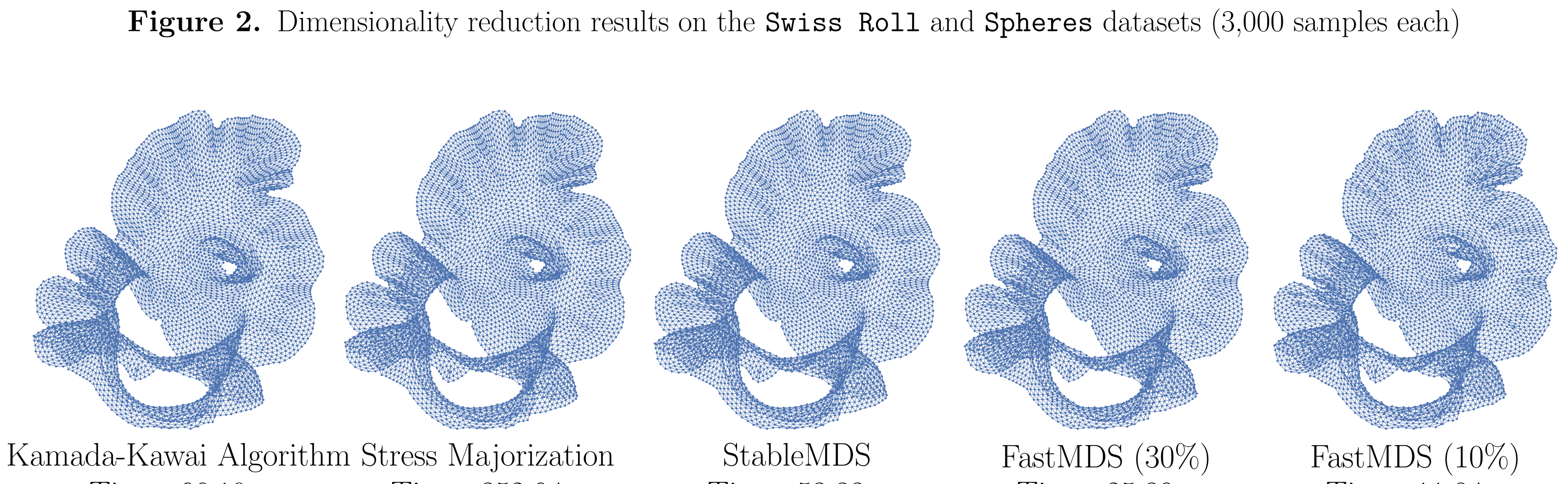

Figure 3. Graph layouts generated by different algorithms on the `3elt` dataset (4,720 nodes, 13,722 edges)