Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 30, no. 1, pp. 489–516 (2026)
DOI: 10.7155/jgaa.v30i1.3113

Initial Placement for Fruchterman–Reingold Force Model with Coordinate Newton Direction

Hiroki Hamaguchi[1] *Naoki Marumo*[1] *Akiko Takeda*[1,2]

[1]Department of Mathematical Informatics, The University of Tokyo, Tokyo, Japan
[2]RIKEN Center for Advanced Intelligence Project, Tokyo, Japan

Submitted: October 2025 Accepted: October 2026 Published: Nov. 2026

Article type: Regular Paper Communicated by: S. Kobourov

Abstract. The Fruchterman–Reingold (FR) force model is widely used in force-directed graph drawing, and multilevel approaches such as sfdp in Graphviz scale these methods effectively. A crucial step in multilevel schemes is refinement, which improves the graph layout, typically performed with the simulation-based algorithm. For this refinement, we can utilize optimization-based methods such as L-BFGS, or combine them with initial placement methods such as Simulated Annealing to achieve better layouts. However, they have several limitations, such as suffering from high per-iteration costs for large graphs or having difficulty with weighted and structurally complex graphs, leaving room for improvement.

In this research, we propose a new initial placement based on stochastic coordinate descent to accelerate the optimization process. We first reformulate the problem as a discrete optimization problem using a hexagonal lattice and then iteratively update a randomly selected vertex along the coordinate Newton direction with low per-iteration costs. We demonstrate the effectiveness of our method through numerical experiments, showing that our initial placement leads to faster convergence and higher-quality layouts compared to naive optimization approaches. We also discuss applications of our method, such as drawing for Hooke–Coulomb and Eades force models.

1 Introduction

Graph drawing is a fundamental task in information visualization, and force-directed methods are among the most widely used approaches. In these methods, vertices are treated as particles subject to forces, and the resulting equilibrium layouts are regarded as desirable visualizations. Among

This work was supported by JSPS KAKENHI Grant Numbers JP26KJ0936, JP23H03351, JP24K23853 and JST CREST JPMJCR24Q2.

E-mail addresses: hirokihamaguchi0331@gmail.com (Hiroki Hamaguchi) marumo@mist.i.u-tokyo.ac.jp (Naoki Marumo) takeda@mist.i.u-tokyo.ac.jp (Akiko Takeda)

This work is licensed under the terms of the CC-BY license.

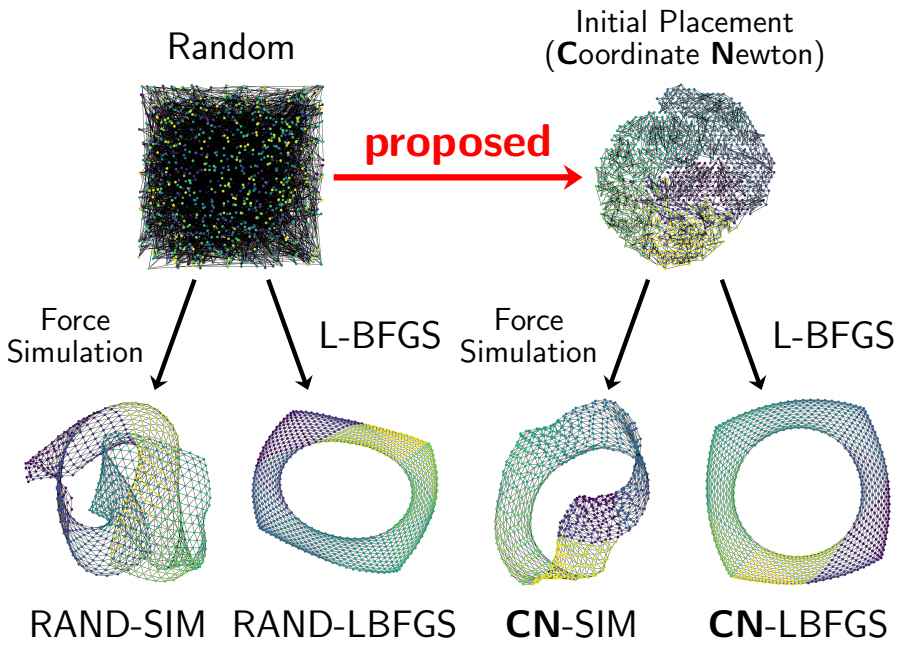


Figure 1: Comparison of the algorithms for the `jagmesh1`.

established force models [10, 26], the Fruchterman–Reingold (FR) force model [15] is particularly prominent for its intuitive formulation, flexibility, and simplicity. It is implemented in libraries such as NetworkX [20], Graphviz [12], and igraph [8].

A central challenge in force-directed graph drawing is efficiently obtaining an equilibrium state of the force system. In force-directed methods, the refinement process iteratively adjusts vertex positions to reduce force imbalance, and its efficiency strongly affects both the computational cost and the quality of the visualization. This refinement remains important even in multilevel approaches such as the Scalable Force-Directed Placement (sfdp) algorithm [12, 25], where layouts must still be refined while propagating solutions from coarse to fine graph levels.

Two major approaches are commonly used for refinement. The first is a simulation-based approach [15], which repeatedly applies the forces to vertices to approximate equilibrium. The second is numerical optimization based on explicit energy minimization. In particular, the L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno) algorithm [27], a general-purpose optimization method, has been reported to produce high-quality layouts for several force models [24], including the FR force model and Kamada–Kawai (KK) model [26]. Although they are effective, both approaches require many computationally expensive iterations. When the number of iterations is insufficient, the resulting layouts may contain twists, including folded or tangled structures that reduce readability [7, 36], as shown in Fig. 1.

One possible approach to mitigating the issue is to improve the initial placement before refinement. Since both the simulation-based algorithm and the L-BFGS algorithm involve relatively expensive iterations, it is often more efficient to first move the vertices toward a reasonably good configuration using inexpensive approximate updates, and then apply full-cost refinement methods only after the layout becomes sufficiently close to equilibrium. This pre-processing substitutes for

part of the refinement while reducing the overall optimization difficulty.

A pre-processing step based on Simulated Annealing (SA) has been shown to improve the final visualization quality when combined with the simulation-based algorithm [18]. However, the existing method assumes circular initial layouts and employs a relatively heuristic objective function, insufficiently grounded in the underlying force model. In addition, the method mainly targets unweighted, small-scale, and structurally simple graphs, limiting its applicability to more general graph drawing settings.

In this paper, we propose a new initial placement to quickly obtain the equilibrium layout of the force models, especially the FR force model. We provide an initial placement with fewer twists than random placement within a short time, accelerating the subsequent optimization process. We can use both the simulation-based and L-BFGS algorithms to obtain the final placement. This work extends the applicability of the initial placement idea to larger-scale, weighted, and complicated structured graphs. To achieve this, we optimize the position of vertices one by one with the coordinate Newton direction, leveraging the inherent structure and the sparsity of graphs. We also demonstrate its effectiveness through various experiments and discuss its potential applications, such as drawing for the HC (Hooke–Coulomb) [11] and Eades [10] force models.

The remainder of the paper is organized as follows.

- In Sec. 2, we provide the technical definitions of the problem.
- In Sec. 3, we review some related work.
- In Sec. 4, we present our initial placement algorithm.
- In Sec. 5, we report experimental comparisons.
- In Sec. 6, we discuss the potential applications and limitations.
- In Sec. 7, we conclude the paper and discuss future work.

2 Preliminaries

In this study, we adopt an energy-based optimization approach for graph drawing. While Fruchterman and Reingold [15] originally proposed the force-directed simulation-based algorithm based on a physical particle system, energy-based methods achieve the same goal by directly minimizing an energy function Φ. These two perspectives are fundamentally equivalent because a local minimum of the energy function $\Phi(X)$ yields an equilibrium position since the negative gradient $-\nabla\Phi(X) = 0$ corresponds to the forces. In this sense, the force and energy perspectives are equivalent as both capture equilibrium, illustrated in Fig. 2.

Let us formally define the energy minimization problem. Let $G = (V, E)$ be an undirected graph with vertex set $V = \{1, \dots, n\}$ and edge set E. Each edge $(i, j) \in E$ $(i < j)$ has positive weight $a_{i,j} = a_{j,i} \in \mathbb{R}$. For convenience, we set $a_{i,j} = 0$ for $(i, j) \notin E$. In this problem, the force model assumes forces between vertices. For vertices i and j with distance $d > 0$, an attractive force $F^{\mathrm{a}}_{i,j}(d)$ and a repulsive force $F^{\mathrm{r}}(d)$ are defined as

$$F^{\mathrm{a}}_{i,j}(d) \coloneqq \frac{a_{i,j} d^2}{k}, \quad F^{\mathrm{r}}(d) \coloneqq -\frac{k^2}{d},$$

where $k > 0$ is a constant parameter, often set to $1/\sqrt{n}$ [15]. The scalar potentials (or energies) of these forces [24] are given by

$$\Phi^{\mathrm{a}}_{i,j}(d) \coloneqq \int_0^d F^{\mathrm{a}}_{i,j}(r)\,\mathrm{d}r = \frac{a_{i,j} d^3}{3k}, \quad \Phi^{\mathrm{r}}(d) \coloneqq \int_1^d F^{\mathrm{r}}(r)\,\mathrm{d}r = -k^2 \log d.$$

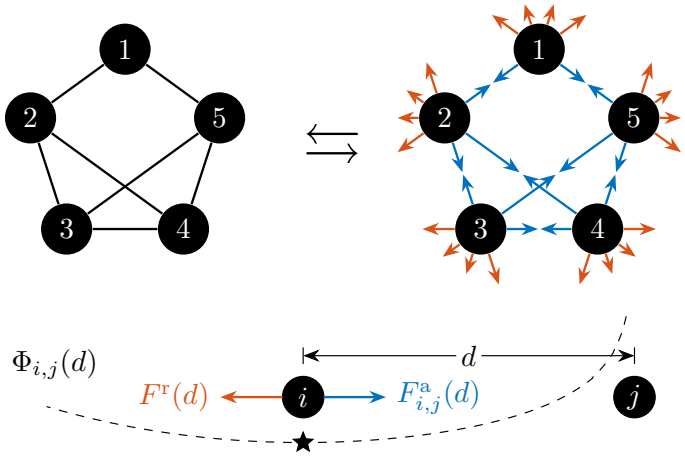


Figure 2: In the model, forces act on every pair of vertices. Forces $F^{\mathrm{a}}_{i,j}(d)$ and $F^{\mathrm{r}}(d)$ work between vertices i and j. The equilibrium of them is achieved at $d = k/\sqrt[3]{a_{i,j}}$, which equals k when $a_{i,j} = 1$.

For simplicity, we define $\Phi^{\mathrm{r}}(0) = \infty$.[1] Let $\|\cdot\|$ denote the Euclidean norm in $\mathbb{R}^2$. Then, the problem is to minimize the energy with $X \coloneqq (x_1, \dots, x_n) \in \mathbb{R}^{2\times n}$:

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad \Phi(X) \coloneqq \sum_{(i,j)\in E} \Phi^{\mathrm{a}}_{i,j}(\|x_i - x_j\|) + \sum_{i<j} \Phi^{\mathrm{r}}(\|x_i - x_j\|). \tag{1}$$

We refer to Problem 1 as the energy minimization problem for the FR force model.

We also define the vertex pairwise energy function $\Phi_{i,j}(d) \coloneqq \Phi^{\mathrm{a}}_{i,j}(d) + \Phi^{\mathrm{r}}(d)$ for convenience. Here, $\Phi_{i,j}(\cdot) = \Phi_{j,i}(\cdot)$ holds for all $i, j \in V$. While the energy function $\Phi_{i,j}(d)$ is convex and minimized when $d = k/\sqrt[3]{a_{i,j}}$ for a positive $a_{i,j}$, a function $x_i \mapsto \Phi_{i,j}(\|x_i - x_j\|)$ is non-convex for a fixed x_j. Additionally, $\Phi_{i,j}$ is not Lipschitz continuous near $d = 0$, as illustrated in Fig. 3. These properties highlight the difficulty of the optimization problem.

3 Related Work

We next summarize the existing literature relevant to our work.

3.1 Optimization Approach

Optimization-based methods for graph drawing have been actively investigated. Representative approaches include GPU acceleration [17], machine learning techniques [33], multi-objective formulations [1], and Stochastic Gradient Descent (SGD) [40]. Among these, our attention is on the underlying optimization algorithms themselves.

The energy minimization problem can be interpreted as a continuous optimization problem. Let Φ_i denote the contribution of vertex x_i to the objective function Φ, defined by

$$\Phi_i(x_i) \coloneqq \sum_{j\in V\setminus\{i\}} \Phi_{i,j}(\|x_i - x_j\|).$$

[1] We can also use $-k^2 \log(d + \epsilon^{\mathrm{r}})$ to prevent divergence, where ϵ^{r} is constant.

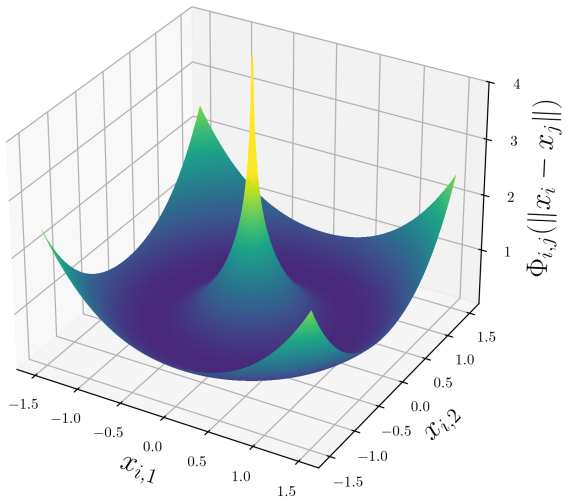


Figure 3: The energy $\Phi_{i,j}(\|x_i - x_j\|)$ for $x_i = (x_{i,1}, x_{i,2})$, $x_j = (0,0), a_{i,j} = 1$, and $k = 1$.

The i-th column of the gradient $\nabla\Phi(X) \in \mathbb{R}^{2\times n}$ is given by

$$(\nabla\Phi(X))_i \coloneqq \nabla\Phi_i(x_i) = \sum_{j\in V\setminus\{i\}} \left(\frac{a_{i,j}\|x_i - x_j\|}{k} - \frac{k^2}{\|x_i - x_j\|^2} \right) (x_i - x_j).$$

Since both the objective value and gradient are explicitly available, a variety of gradient-based optimization methods can be applied to solve the energy minimization problem. The classical simulation-based algorithm (also known as the FR algorithm, presented in Algorithm 2) may be interpreted as a steepest-descent-type method in which the step size is controlled by a cooling schedule. Indeed, $-(\nabla\Phi(X))_i$ corresponds to the force acting on vertex i. The L-BFGS algorithm, a renowned and widely used optimization method, is another first-order gradient-based method that has been shown to consistently outperform the original simulation-based algorithm in speed and layout quality [24]. A common choice for the initial solution is a random placement of nodes within the bounding box [37].

However, when the input is highly twisted or otherwise ill-conditioned, both the simulation-based algorithm and the L-BFGS algorithm can suffer from high computational costs. Each iteration operates on the full set of coordinates, which limits its ability to untangle local geometric defects, and the total number of iterations required to reach a good layout can be large. Thus, for pre-processing purposes, it is desirable to use inexpensive updates that move the layout toward a reasonably good configuration before applying the simulation-based algorithm or L-BFGS. This study therefore proposes stochastic coordinate descent with coordinate Newton directions as a lightweight initial-placement method.

3.2 Initial Placement Approach

A preprocessing step with Simulated Annealing (SA) is known to be effective [18] since SA can avoid getting stuck in local optima and leads to better visualization, combined with the simulation-based

algorithm. Let

$$Q^{\mathrm{circle}} \coloneqq \left\{ \left(\cos\left(\frac{2\pi k}{n}\right), \sin\left(\frac{2\pi k}{n}\right) \right) \;\middle|\; 1 \le k \le n \right\}$$

be the points on a unit circle in $\mathbb{R}^2$. For an unweighted graph G, let E_2 be a set of vertex pairs with a shortest path distance equal to two. Let $\angle(a, b)$ denote the angle between the lines from the origin to the points a and b, measured in the interval $(-\pi, \pi]$. This study [18] defines the problem for the preprocessing of graph drawing as follows:

$$\begin{array}{lll} \underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} & \displaystyle\sum_{(i,j)\in E\cup E_2} |\angle(x_i, x_j)|, & \\ \text{subject to} & x_i \in Q^{\mathrm{circle}} & \text{for } 1 \le i \le n, \\ & x_i \ne x_j & \text{for } 1 \le i < j \le n. \end{array} \tag{2}$$

Problem (2) is a discrete optimization problem where the placement is limited to Q^{circle} and uses angles, not the function Φ in the energy minimization problem. This study obtains faster and better visualization by setting the result of SA for Problem (2) as an initial placement for the simulation-based algorithm.

Still, the following limitations remain:

- The target graphs are restricted to unweighted ones.
- The layout is confined to a simple circle, which could be ineffective for complex structured graphs.
- The neighborhood in the SA is the random swapping of two vertices, making the optimization process inefficient for large-scale graphs.
- $|E_2|$ could be $\Theta(|V|^2)$, unable to leverage the sparsity of graphs if it exists.

We address these limitations by proposing a new initial placement algorithm.

4 Proposed Algorithm

With the prior studies in Sec. 3.1, we propose a new initial placement algorithm for the energy minimization problem. The algorithm solves a discrete optimization problem with stochastic coordinate descent to find an initial placement with fewer twists. This algorithm leverages the properties of the objective function to determine the initial layout quickly. By combining it with the simulation-based algorithm and the L-BFGS algorithm, it enhances the overall convergence speed of graph drawing through optimization.

4.1 Newton Direction and Coordinate Newton Direction

Let us first explain the Newton direction and the coordinate Newton direction, the key concept in our algorithm. Consider a twice continuously differentiable function $g\colon \mathbb{R}^n \to \mathbb{R}$ whose Hessian $\nabla^2 g(x_0)$ is positive definite. The Newton direction $d = -\nabla^2 g(x_0)^{-1} \nabla g(x_0)$ is an optimal direction for the second-order approximation of g:

$$g(x_0) + \nabla g(x_0)^\top (x - x_0) + \frac{1}{2}(x - x_0)^\top \nabla^2 g(x_0)(x - x_0),$$

where $x = x_0 + d$ is the minimizer of this approximation. The positive definiteness of $\nabla^2 g(x_0)$ ensures that this quadratic approximation has a unique minimizer. Although the Newton direction is essential in various iterative methods, it requires the computation of the inverse Hessian $\nabla^2 g(x_0)^{-1} \in \mathbb{R}^{n \times n}$, posing a high computational cost for large-scale problems.

Still, we can leverage the concept of the Newton direction in a different manner: the coordinate Newton direction. Instead of computing the inverse Hessian $\nabla^2 g(x_0)^{-1}$ in the entire variable space $\mathbb{R}^n$, we restrict the variable x to its coordinate block x_i with fewer dimensions, and compute $-\nabla^2 g_i(x_i)^{-1} \nabla g_i(x_i)$ where g_i is a restricted function of g to x_i. Since the coordinate Newton direction computation is much cheaper than that of the Newton direction, we can repeat this procedure many times. In general, this idea is known as stochastic coordinate descent [39] or Randomized Subspace Newton [19] in a broader context.

In particular, this coordinate Newton direction has an apparent natural affinity to the energy minimization problem. We can compute the coordinate Newton direction by taking the position x_i of the vertex i as the coordinate block. Although directly applying this idea to the energy minimization problem is challenging, as we will discuss in Sec. 6.2, we leverage this coordinate Newton direction to propose our algorithm.

4.2 Discrete Optimization Problem for Initial Placement

Even at the expense of accuracy, obtaining an approximate solution quickly is crucial for the initial placement. To obtain it, we simplify the energy minimization problem into a more manageable and well-behaved discrete optimization problem:

$$\begin{aligned} \underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad & \Phi^{\mathrm{a}}(X) \coloneqq \sum_{(i,j)\in E} \frac{a_{i,j}\|x_i - x_j\|^3}{3k}, \\ \text{subject to} \quad & x_i \in Q^{\mathrm{hex}} \quad \text{for } 1 \le i \le n, \\ & x_i \neq x_j \quad \text{for } 1 \le i < j \le n, \end{aligned} \tag{3}$$

where

$$Q^{\mathrm{hex}} \coloneqq \left\{ \left(q + \frac{1}{2}r, \frac{\sqrt{3}}{2} r \right) \,\middle|\, q \in \mathbb{Z},\ r \in \mathbb{Z} \right\}. \tag{4}$$

Let us explain the rationale for this formulation. First, recall that the energy minimization problem is equivalent to the following:

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad \sum_{(i,j)\in E} \frac{a_{i,j}\|x_i - x_j\|^3}{3k} - \sum_{i<j} k^2 \log \|x_i - x_j\|.$$

This formulation separates the $\mathcal{O}(|E|)$ terms from $\Phi^{\mathrm{a}}_{i,j}$ and the $\mathcal{O}(|V|^2)$ terms from Φ^{r}. Following previous research mentioned in Sec. 3.2, we fix the possible positions x_i of each vertex i to a discrete set of points Q, where $|Q| \ge |V|$. We also impose a condition on Q such that $\|q_i - q_j\| \ge 1$ for all $q_i, q_j \in Q$ with $q_i \neq q_j$. This makes the second $\mathcal{O}(|V|^2)$ terms from Φ^{r} bounded above by zero. Since $\Phi^{\mathrm{r}}(d) = -k^2 \log d$ is a convex function such that it decreases monotonically with respect to d, for sufficiently large d, the value of $-k^2 \log d$ does not vary excessively. We can also prevent the repulsive energy from diverging to positive infinity as vertices approach each other by imposing

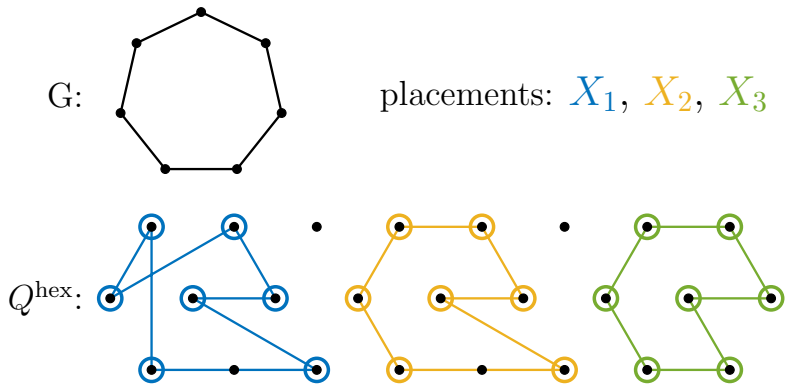


Figure 4: Concept of Q. The assignment from V to a discrete point placement Q, especially a hexagonal lattice Q^{hex}. Apparently, among the three placements, the right one is the best placement for Problem (3).

the condition $\|q_i - q_j\| \geq 1$. Under these observations, we drop the second term and derive the problem as:

$$\begin{aligned} \underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad & \sum_{(i,j)\in E} \frac{a_{i,j}\|x_i - x_j\|^3}{3k}, \\ \text{subject to} \quad & x_i \in Q \quad \text{for } 1 \leq i \leq n, \\ & x_i \neq x_j \quad \text{for } 1 \leq i < j \leq n. \end{aligned}$$

This means that we avoid considering the $\mathcal{O}(|V|^2)$ pairs by fixing the possible point placement in advance, reducing the computational complexity to $\mathcal{O}(|E|)$ and thus offering significant speedup. See Fig. 4 for a visual explanation.

In this study, we adopt a hexagonal lattice Q^{hex} [31] defined by Eq. (4) (see Fig. 4) as Q. When minimizing the objective function that arises from the attractive energy Φ^{a}, it is advantageous for the points to cluster as closely as possible. In this context, the hexagonal lattice is known for its densest packing structure in space [6] and offers computational simplicity.

We note that the scaling of the lattice structure does not affect the optimization process. As we will see later, our algorithm is scale invariant, meaning that the distance $\|q_i - q_j\|$ does not affect the update.

4.3 Newton Direction for Discrete Optimization

Next, we formulate an optimization procedure for Problem (3). Although it is challenging to solve, the coordinate Newton direction of a randomly selected vertex i provides significant insights as mentioned in Sec. 4.1. Let the restricted objective function $\Phi_i^{\mathrm{a}}(x_i)$ be

$$\Phi_i^{\mathrm{a}}(x_i) \coloneqq \sum_{j\in V\setminus\{i\}} \frac{a_{i,j}\|x_i - x_j\|^3}{3k}.$$

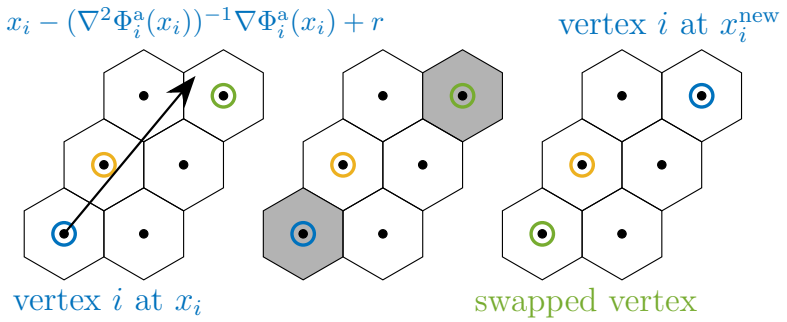


Figure 5: One iteration of the proposed algorithm. Step 1. Compute the coordinate Newton direction (blue). Step 2. Decide x_i^{new} by adding a random vector and rounding. Step 3. Move the vertex and swap the vertices if necessary (swap blue and green vertices).

Its gradient and Hessian matrix are

$$\nabla \Phi_i^{\mathrm{a}}(x_i) = \sum_{j \in V \setminus \{i\}} \frac{a_{i,j} \|x_i - x_j\|}{k} (x_i - x_j),$$

$$\nabla^2 \Phi_i^{\mathrm{a}}(x_i) = \sum_{j \in V \setminus \{i\}} \frac{a_{i,j} \|x_i - x_j\|}{k} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \sum_{j \in V \setminus \{i\}} \frac{a_{i,j}}{k \|x_i - x_j\|} (x_i - x_j)(x_i - x_j)^\top.$$

For distinct vertex positions, the above expression shows that $\nabla^2 \Phi_i^{\mathrm{a}}(x_i)$ is positive definite whenever vertex i has at least one incident edge with positive weight.

The standard update rule with the coordinate Newton direction is

$$x_i^{\mathrm{new}} \leftarrow x_i - \nabla^2 \Phi_i^{\mathrm{a}}(x_i)^{-1} \nabla \Phi_i^{\mathrm{a}}(x_i).$$

Note that when we scale the lattice structure by a factor of $s > 0$, the gradient is scaled by s^2 and the Hessian matrix is scaled by s, resulting in the coordinate Newton direction being scaled by s. Thus, the algorithm for the FR force model is scale invariant. Since x_i^{new} may not be in the hexagonal lattice Q^{hex} in Problem (3), we project this x_i^{new} onto the nearest point in Q^{hex}. We also empirically found that adding a random noise vector to the direction is effective for the optimization process, a strategy similar to the SA in Sec. 3.2. This randomness can help to escape from local minima and to explore the solution space more effectively. In conclusion, the update rule for the vertex i is

$$x_i^{\mathrm{new}} \leftarrow \mathrm{round}\big(x_i - \nabla^2 \Phi_i^{\mathrm{a}}(x_i)^{-1} \nabla \Phi_i^{\mathrm{a}}(x_i) + r\big),$$

where $\mathrm{round}(\cdot)$ denotes the operation assigning a point in $\mathbb{R}^2$ to the nearest point in the hexagonal lattice Q^{hex}, and r is a random vector with a decreasing magnitude (we will detail in Sec. 5.1).

If there is a vertex j such that $x_j = x_i^{\mathrm{new}}$, we swap the positions x_i and x_j to satisfy the condition $x_i \neq x_j$. Otherwise, we just update x_i to x_i^{new}. Refer to Fig. 5 for a visual explanation. Repeating this procedure yields an approximate solution to Problem (3).

4.4 Optimal Scaling

As the final step, we rescale the placement. Let us formulate the optimization problem for the scaling factor $s > 0$. For an initial placement $X = (x_1, \ldots, x_n)$, we scale it as $x_i \leftarrow s x_i$ for all i.

Algorithm 1: Proposed algorithm for the initial placement

Input: Graph $G = (V, E)$, Weight $(a_{i,j})_{(i,j)\in E}$, Parameter $N_{\mathrm{iter}}^{\mathsf{CN}} \in \mathbb{N}$.
Output: Initial placement $X = (x_1, \dots, x_n)$

```
Assign all vertices to distinct points in Q^hex randomly;
for m ← 0 to N_iter^CN − 1 do
    Select vertex i ∈ V randomly;
    x_i^new ← round(x_i − ∇²Φ_i^a(x_i)^{-1} ∇Φ_i^a(x_i) + r) with r being a random vector;
    if ∃j ∈ V such that x_j = x_i^new then
        x_j ← x_i;
    x_i ← x_i^new;
x_i ← s* x_i for all i ∈ V with s* given by Eq. (5);
return X
```

The problem of minimizing $\Phi(X)$ through scaling is equivalent to minimizing $\phi(s)$ defined by

$$\phi(s) \coloneqq \sum_{(i,j)\in E} \frac{a_{i,j}(s\|x_i - x_j\|)^3}{3k} - k^2 \sum_{i<j} \log(s\|x_i - x_j\|),$$

$$\phi'(s) = \sum_{(i,j)\in E} \frac{a_{i,j}\|x_i - x_j\|^3}{k} s^2 - \frac{k^2 n(n-1)}{2s}.$$

The function $\phi(s)$ is convex, and the optimal scaling factor s^* satisfies $\phi'(s^*) = 0$:

$$s^* = \left(\frac{k^3 n(n-1)}{2\sum_{(i,j)\in E} a_{i,j}\|x_i - x_j\|^3} \right)^{1/3}. \tag{5}$$

This value can be computed in $\mathcal{O}(|E|)$ complexity, completing the initial placement algorithm.

We summarize the proposed algorithm in Algorithm 1.

4.5 Application to Other Force Models

To conclude this section, we discuss the application of our proposed method to other force models. Although we have focused only on the FR force model so far, the concept of our approach can be applied to other models, as long as the objective function has a similar structure to $\Phi_{\mathrm{FR}}(X)$. Recall that the energy minimization problem (1) is

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad \Phi_{\mathrm{FR}}(X) = \sum_{(i,j)\in E} \frac{a_{i,j}\|x_i - x_j\|^3}{3k} - \sum_{\substack{(i,j)\in V^2 \\ i<j}} k^2 \log\left(\|x_i - x_j\| + \epsilon^{\mathrm{r}}\right),$$

where ϵ^{r} is a small constant to prevent divergence. Similar objective functions arise from other force models, such as the HC (Hooke–Coulomb) [11] and Eades [10] force models, which are as follows:

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad \Phi_{\mathrm{HC}}(X) \coloneqq \sum_{(i,j)\in E} \frac{k_1(\|x_i - x_j\| - a_{ij})^2}{2} + \sum_{\substack{(i,j)\in V^2 \\ i<j}} \frac{k_2}{\|x_i - x_j\| + \epsilon^{\mathrm{r}}},$$

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad \Phi_{\text{Eades}}(X) \coloneqq \sum_{(i,j)\in E} k_1 \|x_i - x_j\| \left(\log \frac{\|x_i - x_j\|}{a_{ij}} - 1\right) + \sum_{\substack{(i,j)\in V^2 \\ i<j}} \frac{k_2}{\|x_i - x_j\| + \epsilon^{\text{r}}}.$$

See also [24]. In general, these optimization problems are more broadly treated as "objective functions arising from graphs." [39] Clearly, these objective functions have a similar structure to $\Phi_{\text{FR}}(X)$, which consists of an attractive term that depends on the edges and a repulsive term that depends on all pairs of vertices.

Thus, our proposed approach can be applied to these force models as well. In these models, however, the Hessian matrix for the coordinate Newton direction may be indefinite, so the resulting Newton direction is not necessarily a descent direction. This issue can be addressed by standard remedies in the optimization literature, such as adding a damping term [28], or using regularization to ensure positive definiteness [35]. Clipping the maximum step size can additionally prevent excessively large updates. Hence, the coordinate Newton direction can still be used in almost the same way as in the FR model. Another difference is that this direction depends on the scaling of Q^{hex}; we therefore numerically optimized the scaling factor s and applied this scaling before computing the initial placement.

We emphasize that the proposed method is not suitable for all graph-derived objective functions. A representative example is the KK model [26], whose objective function is

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad \Phi_{\text{KK}}(X) \coloneqq \sum_{\substack{(i,j)\in V^2 \\ i<j}} \frac{(\|x_i - x_j\| - l_{ij})^2}{2l_{ij}^2}.$$

This objective differs from those considered above because it consists only of all-pairs terms and has no principal edge-based term. Since our approach computes the Hessian matrix for edge-defined terms, it is not directly applicable to the KK model. Although one could compute the coordinate Newton direction for this objective directly, its behavior is not necessarily favorable; see also Sec. 6.2.

For the KK model, SGD-based methods have been proposed [1, 40] and are likely more appropriate. These methods repeatedly select a pair of vertices and optimize with respect to that pair, making them complementary to our method, which repeatedly selects a single vertex and optimizes with respect to that vertex.

5 Numerical Experiment

In this section, we evaluate the proposed algorithm with various numerical experiments.

5.1 Experimental Setup

We first summarize the experimental setup. We conducted all numerical experiments in this section using C++17 compiled with g++ 13.3.0. The experiments were run on a Dynabook CZ/MWS laptop equipped with a 13th Gen Intel Core i7-1360P processor and 16 GB of RAM. The software environment was Ubuntu 24.04 LTS under Windows Subsystem for Linux 2 on Windows 11 Home 64-bit.

For a fair comparison, we implemented the simulation-based algorithm in C++ based on NetworkX version 3.3 [20], SciPy 1.14.1 [37], and utilized the C++ L-BFGS [30, 32] library for the

L-BFGS algorithm. We also referred to the open-source code of the hexagonal grid [31] and Graphviz version 2.43.0 [12].

We used the 3×2 algorithms. As initial placements, we used

- random initialization (RAND-),
- the SA initialization obtained by Simulated Annealing (SA-) [18], and
- the proposed initialization obtained with coordinate Newton direction (CN-).

As algorithms to solve the energy minimization problem, we used

- the simulation-based algorithm (SIM), and
- L-BFGS algorithm (LBFGS).

The key remark is that we slightly modified the original simulation-based algorithm (presented in Algorithm 2) so that it automatically adjusts the stepsize and isolates the effect of the initialization from that of the stepsize. Specifically, we applied a simple backtracking line search at the first iteration to suppress overshooting. The maximum numbers of iterations of the simulation-based and L-BFGS algorithms are $N_{\mathrm{iter}}^{\mathsf{SIM}} = N_{\mathrm{iter}}^{\mathsf{LBFGS}} = 150$ for the FR force model, and $N_{\mathrm{iter}}^{\mathsf{LBFGS}} = 300$ for the HC and Eades force models.

As parameters, we set the number of iterations $N_{\mathrm{iter}}^{\mathsf{CN}} = 2|V|^3/|E|$ for Algorithm 1, and we also set the total number of iterations of Simulated Annealing (SA) in Sec. 3.2 to the same value. For a random vector r in Algorithm 1, the magnitude at iteration m is drawn uniformly from $[0, \max\{(1 - 0.5m/N_{\mathrm{iter}}^{\mathsf{CN}})\|d_i\|, h\})$, and its direction is uniformly distributed in $[0, 2\pi)$. Here, d_i is the coordinate Newton direction for the selected vertex i, and h is the lattice spacing, which is one for the FR model. Since Algorithm 1 requires a computational time proportional to the degree of the selected vertex in each iteration, the expected computational time per iteration is $\mathcal{O}(|E|/|V|)$. Consequently, we can roughly estimate that the total computational time of the proposed algorithm is $\mathcal{O}(|V|^2)$, equivalent to a few iterations of the FR or L-BFGS algorithms. All the code is available on GitHub [21].

5.2 Plots and Visualizations

We first show the plots and visualizations of the four algorithms in Fig. 6, RAND-SIM, CN-SIM, RAND-LBFGS, and CN-LBFGS. We tested with 7 graphs: `cycle300`, `jagmesh1`, `dwt_1005`, `btree9`, `1138_bus`, `dwt_2680`, and `3elt`. Here `cycle300` is a cycle graph with 300 vertices, and `btree9` is a perfect binary tree with $2^{9+1} - 1 = 1023$ vertices. Other graphs are from the Sparse Matrix Collection [9], and these choices are based on Ref. [40]. We provide more visualization results for various graphs in Sec. A.

In Fig. 6, the plots on the left illustrate the objective function values $\Phi(X)$, the average of the five trials for each algorithm. The graphs on the right are at the 100th iteration of the simulation-based and L-BFGS algorithms, and each graph in the BEST column is at the 1000th iteration of CN-LBFGS as a reference for the best solution. The reference layouts in the BEST columns of Figs. 6 and 13 use random seed 1. The vertices in the graphs are colored according to vertex indices.

The observations and implications of Fig. 6 are as follows. First, the plots of the objective function values show that the proposed CN methods, represented by solid lines, start from substantially lower objective values. This indicates that they produce high-quality approximate solutions within at most a few seconds, which is the main contribution of the proposed initialization. Overall, the CN methods outperform their non-CN counterparts over the iterations. In particular, CN-SIM

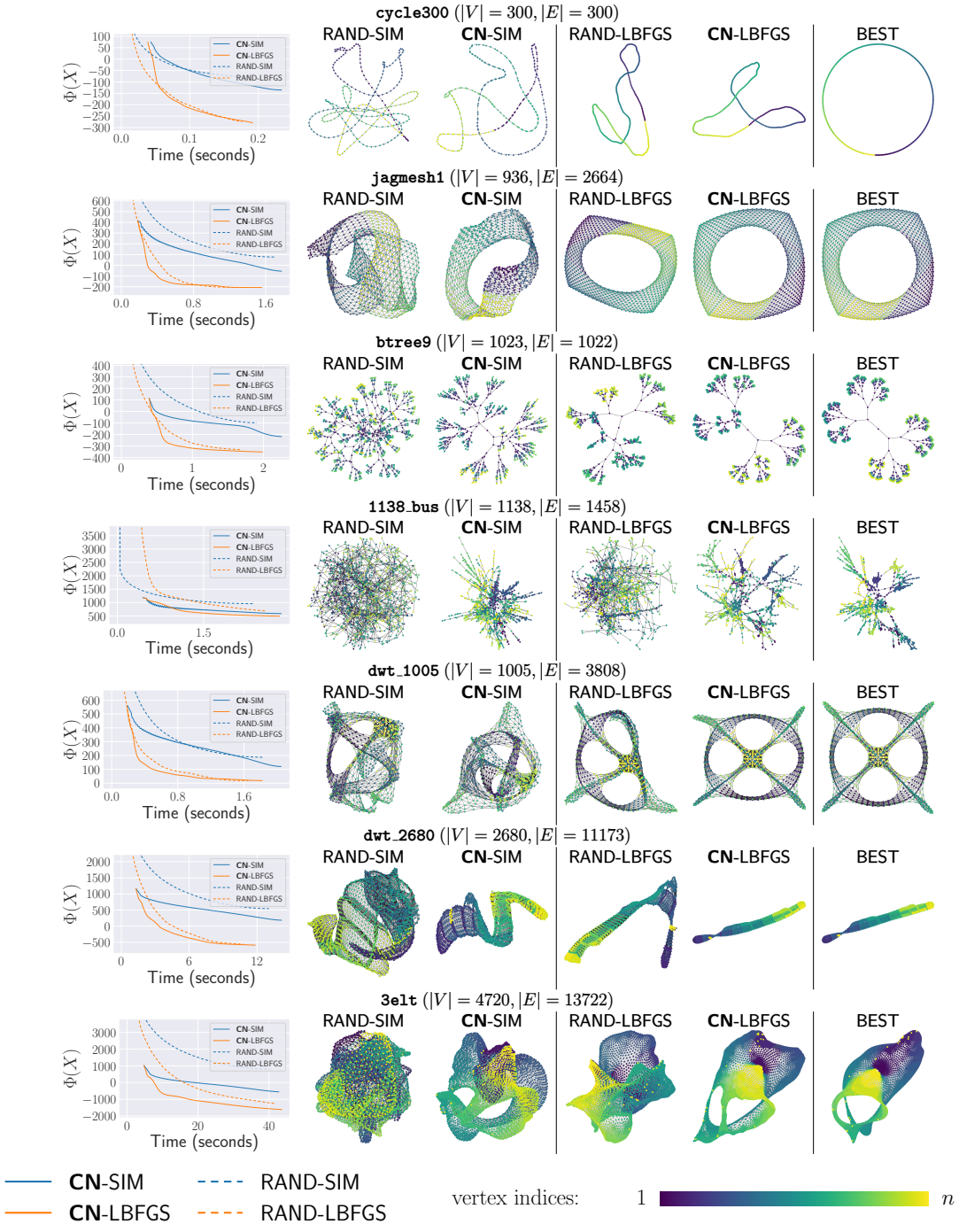


Figure 6: Experimental results for the FR force model. Our proposed method CN yields improved results for both the simulation-based algorithm and L-BFGS. The BEST column shows the reference layout obtained by running CN-LBFGS for 1000 iterations.

consistently improves upon RAND-SIM across almost all graphs and iteration counts. Similarly, CN-LBFGS often achieves lower objective values than RAND-LBFGS, although both sometimes converge to the same value. This suggests that, given sufficiently many iterations, the effect of the initial layout on the final solution quality may diminish. Under limited iteration budgets, the advantage of the proposed initialization remains substantial.

Second, the visualization results further support the effectiveness of the proposed initial placement. In most cases, CN produces layouts that better approximate the nearly optimal arrangement, with more structured and less cluttered placements than random initialization. Note that subtle differences in the objective values can still correspond to visible differences in the resulting layouts. Additionally, the plot shows averages of the trials, whereas each visualization comes from a single trial.

As a side note, regardless of CN or non-CN, the algorithms with LBFGS consistently outperform those with SIM. This finding is consistent with prior research [24].

5.3 Other Force Models

As we have mentioned in Sec. 4.5, the proposed algorithm can be applied to other force models, such as the HC and Eades models. The main differences from the experimental setup for the FR model are as follows. (i) The graph weights are all set to one, since the models are quite sensitive to extreme edge weights. (ii) The total number of iterations is 300 as mentioned in Sec. 5.1. (iii) We only run the L-BFGS algorithm since the simulation-based algorithm was originally designed for the FR model and thus is not expected to perform well for these models.

We showed the results for these models in Fig. 7. The proposed algorithm also performed better than random initialization for these models, and the differences in the graphs are quite expressive in some cases. This is because these force models are much more difficult to optimize than the FR model, and thus the quality of the initial placement has a more significant effect on the final solution.

5.4 Comparison with Other Initializations

Next, we conducted experiments to evaluate the performance of the proposed algorithm (CN) compared to random initialization (RAND) and SA initialization (SA) with various graphs. We used all undirected, connected, non-negative weighted graphs with 1,000 or fewer vertices, which can be generated using the matrices from the Sparse Matrix Collection [9] as adjacency matrices. In total, we tested with 124 graphs. For fairness, when we compare with SA, we set all the weights to one since the SA initialization was originally designed for unweighted graphs.

For the algorithms, we set $N_{\mathrm{iter}}^{\mathsf{SIM}} = N_{\mathrm{iter}}^{\mathsf{LBFGS}} = 50$, the same as the default parameter in NetworkX [20]. Only when we compare CN algorithms with RAND algorithms, we set $N_{\mathrm{iter}}^{\mathsf{SIM}} = N_{\mathrm{iter}}^{\mathsf{LBFGS}} = 45$ for the CN algorithms, since the preprocessing step is expected to take as long as a few iterations of FR or L-BFGS, as mentioned in Sec. 5.1.

The results are shown in Fig. 8. The proposed algorithm performed better than random or SA initialization, except for a few cases. These results strongly support the effectiveness of the proposed algorithm. Even when the proposed algorithm performed worse, the difference was insignificant in almost all cases.

We also investigated cases in which the proposed algorithm performed worse than SA initialization in Fig. 9. The proposed algorithm CN outperformed SA on `collins_15NN`, whereas it underperformed on `dwt_992`.

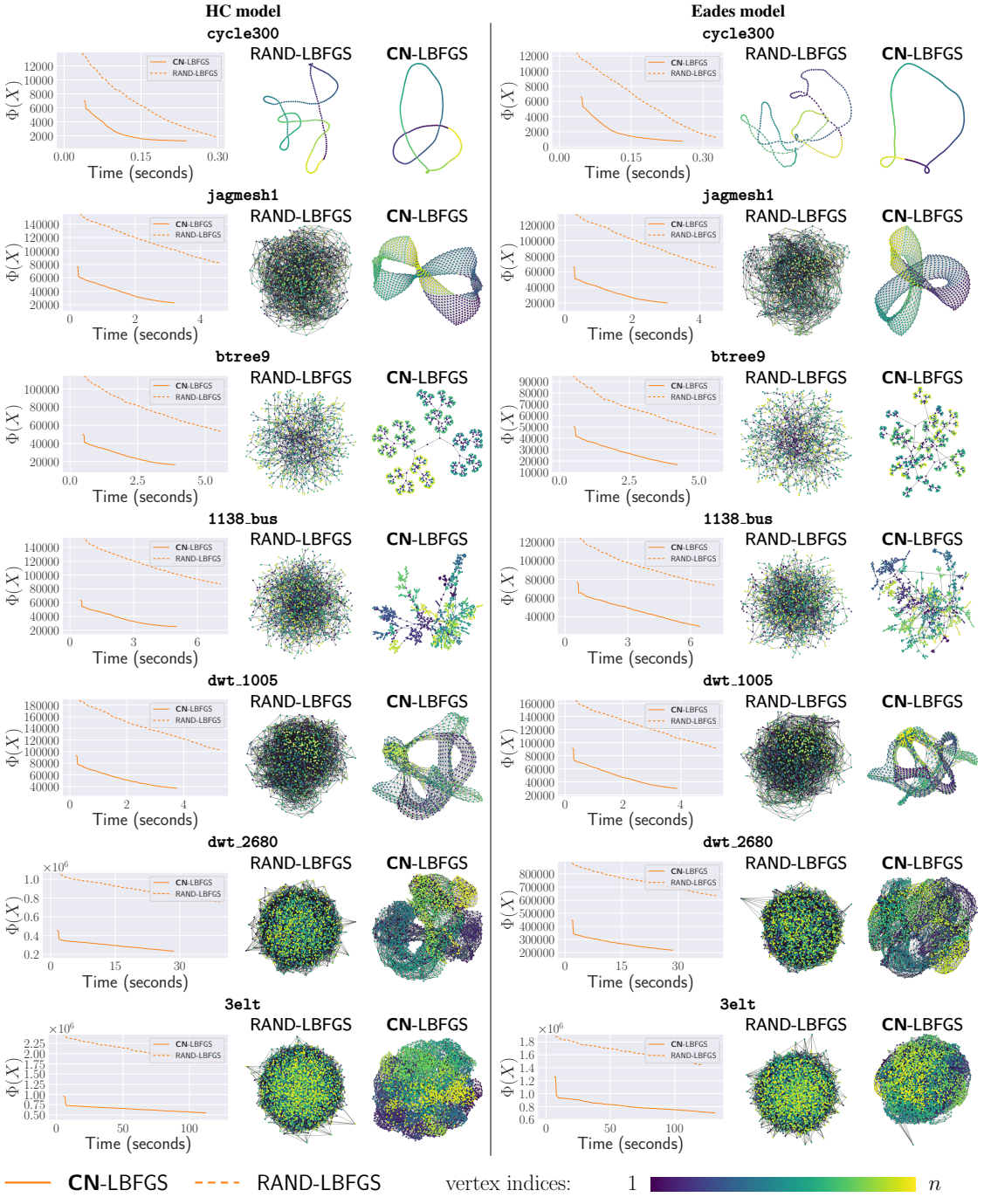


Figure 7: Experimental results for the HC and Eades force models. Our proposed method CN yields improved results.

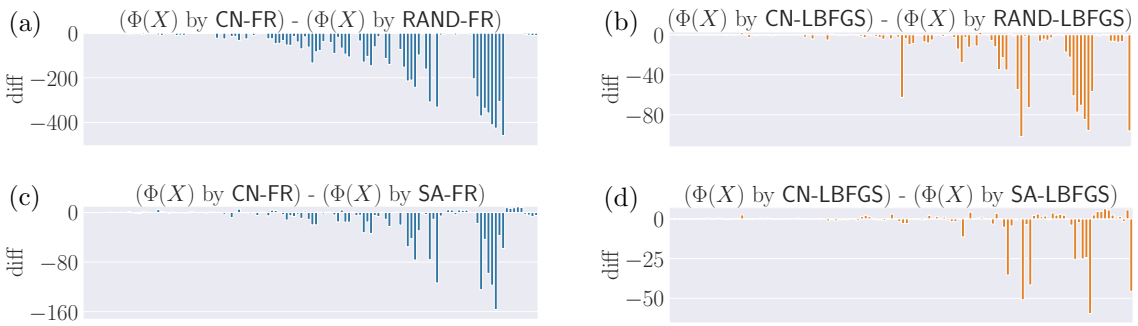


Figure 8: Comparison of the initializations. The bins arranged along the x-axis correspond to the 124 graphs. The y-axis shows the difference in $\Phi(X)$ between CN and the baseline initialization method, averaged over ten trials. Panels (a) and (b) compare CN with RAND, whereas panels (c) and (d) compare CN with SA. Blue ones show results obtained using SIM, whereas orange ones show results obtained using LBFGS. In almost all cases for RAND and in most cases for SA, the differences are negative, meaning that the proposed initialization produced better final layouts than the baseline initialization methods.

In many cases, the main factor that slows the convergence of subsequent simulation-based algorithms or L-BFGS algorithms is the presence of a twisted initial configuration, since untwisting such a configuration often requires many iterations. Indeed, for `dwt_992`, CN failed to resolve this twist, and the final layout remained twisted. Conversely, for `collins_15NN`, the initial layout produced by SA was twisted, causing the subsequent algorithm to spend additional time resolving the twist.

In general, it is difficult to predict how an initial layout will affect the final drawing. This is because multiple factors interact in a complex manner, including the structure of the graph, the fixed positional constraints imposed by the initial layout, the characteristics of the optimization algorithm, and the random seed.

Although the proposed method uses a hexagonal lattice Q^{hex}, alternative lattices or point sets, such as Q^{circle}, may improve the performance of the proposed method in some cases.

Furthermore, although this is a case not considered in Ref. [18], we also conducted experiments with CN and SA for a case where the graph is weighted, i.e., the edge weight $a_{i,j}$ is not necessarily in $\{0, 1\}$. We generated a weighted graph with 100 vertices in three groups and 1000 edges randomly, and if the two vertices were in the same group, we set the edge weight to 1.0; otherwise, we set it to 0.1. The difference between the two initializations is shown in Fig. 10. When we ignore the edge weights and just solve Problem (2), the graph is just an Erdős–Rényi graph, and thus SA cannot find any meaningful structure in the initial placement. In contrast, the proposed algorithm can identify the graph's structure, and we can observe that the graph in Fig. 10(a) is separated into groups, as indicated by the node colors. This result suggests that our proposed algorithm is effective even for weighted graphs, extending the applicability of the preprocessing step.

6 Discussion

In this section, we discuss the implications of the proposed method and its potential applications and extensions.

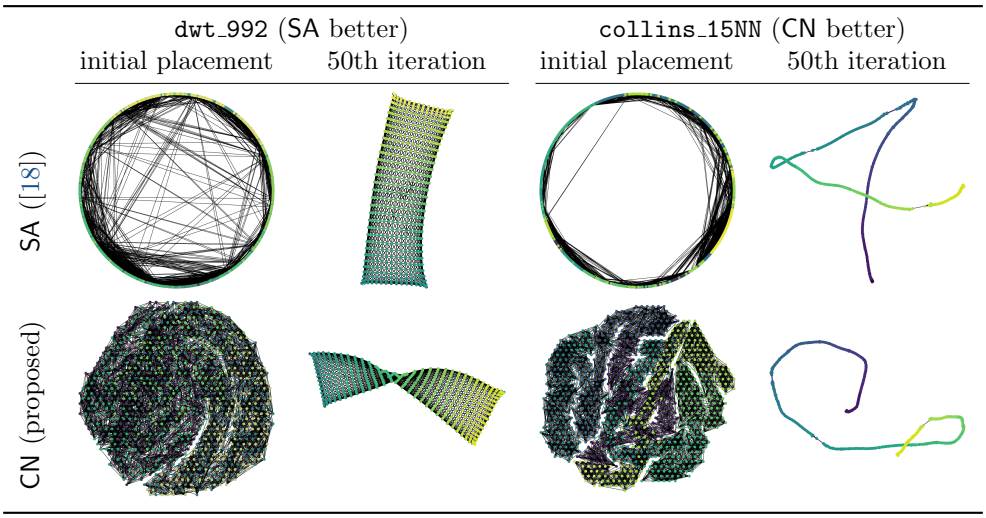


Figure 9: Visualization results showing initial and 50th iteration placements for `dwt_992` and `collins_15NN`.



Figure 10: An example with a weighted graph. The color indicates the group of vertices. (a): the result of CN, which is better as the vertices are separated by colors. (b): the result of SA, which is worse as there is no separation.

6.1 Application to Multilevel Approaches

We briefly discuss how the proposed method can be combined with multilevel approaches. Multilevel approaches, such as sfdp [25] in Graphviz, constitute important prior work for computing equilibrium layouts under force-directed models, that is, for solving the associated energy minimization problem. Typically, a graph is recursively coarsened, laid out at the coarsest level, and refined through successive uncoarsening steps using simulation-based methods.

Our work is complementary to multilevel layout methods. The proposed method could be used at the coarsest or selected intermediate levels as an alternative to the random initialization commonly used in multilevel approaches [22, 25]. It could also be used during refinement: after a coarse vertex is expanded into finer vertices [22, 25], the proposed initialization could be applied locally to the induced subgraph.

The proposed variants may also be combined with other refinement methods. It is known that simulation-based refinement could be replaced by other methods [38], and our proposed placement or its variants can be used as an initial placement for such methods. Although our implementation uses a hexagonal lattice, the essential requirement is the availability of sufficiently separated can-

didate positions. Thus, similar initialization or coordinate-wise minimization strategies could be applied globally over the whole graph after coarsening. Determining the most effective implementation, the appropriate level at which to apply the method, and the graph classes for which this integration is most beneficial remains an important direction for future work. For further details on multilevel approaches, the reader is referred to [3].

The proposed approach may also be useful for dynamic graph layouts, including animation and interactive visualization. Such scenarios are among the most prominent applications of the FR force model [13, 14]. Since inexpensive updates are important in real-time large-graph settings, our algorithm's lower per-iteration cost than the standard FR algorithm may help initialize new vertices, improve local configurations, or accelerate refinement in scalable simulation-based dynamic layouts. Since temporal coherence is central in dynamic graph drawing, integration with dynamic layout algorithms requires careful investigation, which we leave for future work.

6.2 Rationale for the Discretization Approach

This section explains why we did not apply the coordinate Newton direction technique directly to the original Problem (1), but instead first transformed it into the discrete optimization Problem (3). At first sight, one might expect that applying the coordinate Newton method directly to (1) would be more natural and possibly more efficient, since it avoids the detour through discretization. However, as we shall see, this approach suffers from intrinsic difficulties that undermine its effectiveness. In contrast, our discretization-based formulation allows us to circumvent these difficulties while still exploiting the advantages of the coordinate Newton direction. This not only justifies our methodological choice but also provides a foundation for exploring further optimization strategies built upon this framework. In what follows, we clarify the limitations of the direct approach and explain how our method addresses them.

6.2.1 Possible Alternatives

One natural idea for solving the energy minimization problem is to apply stochastic coordinate descent, which resembles Randomized Subspace Newton methods [5, 16, 19, 23, 29]. For each vertex i, define the local energy

$$\Phi_i(x_i) \coloneqq \sum_{j \in V \setminus \{i\}} \Phi_{i,j}(\|x_i - x_j\|), \tag{6}$$

whose gradient (the negative of the net force on i) and Hessian are

$$\nabla \Phi_i(x_i) = \sum_{j \in V \setminus \{i\}} \left(\frac{a_{i,j}\|x_i - x_j\|}{k} - \frac{k^2}{\|x_i - x_j\|^2} \right)(x_i - x_j),$$

$$\nabla^2 \Phi_i(x_i) = \sum_{j \in V \setminus \{i\}} \left(\frac{a_{i,j}\|x_i - x_j\|}{k} - \frac{k^2}{\|x_i - x_j\|^2} \right) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \sum_{j \in V \setminus \{i\}} \left(\frac{a_{i,j}}{k\|x_i - x_j\|} + \frac{2k^2}{\|x_i - x_j\|^4} \right)(x_i - x_j)(x_i - x_j)^\top.$$

A direct stochastic coordinate descent scheme randomly selects a vertex i, applies Newton's method (or a regularized variant) to Φ_i using the above gradient and Hessian, and updates x_i iteratively until convergence.

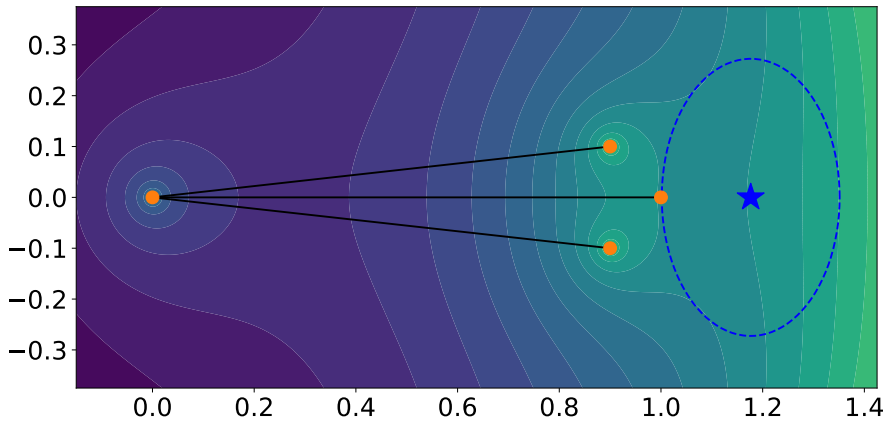


Figure 11: The inaccurate quadratic approximation. The blue star indicates the optimal solution for the quadratic approximation of $\Phi_1(x_1)$, but this differs significantly from the situation shown by the contour lines.

Although reasonable in spirit, this approach is also ineffective in practice. In the following, we illustrate the reasons for this failure with nontrivial examples.

6.2.2 Inaccuracy of Quadratic Approximation

One of the reasons it fails is the inaccuracy of the quadratic approximation; specifically, a particular issue arises when we restrict the optimization to a coordinate block.

Let G be a graph with $V = \{1, 2, 3, 4\}$ and $E = \{(1, 2), (2, 3), (2, 4)\}$. Set $k = 1/2$ and assign all positive edge weights $a_{i,j} = 1$ for every edge in E. The position of the vertices is

$$X = \begin{pmatrix} 1 & 0 & 0.9 & 0.9 \\ 0 & 0 & +0.1 & -0.1 \end{pmatrix}.$$

We show the contour of $\Phi_1(x_1)$ in Fig. 11. The key point of this example is that the Hessian

$$\nabla^2 \Phi_1(x_1) = \begin{pmatrix} 4.25 & 0 \\ 0 & 1.75 \end{pmatrix}$$

is positive definite and well-conditioned. Despite this favorable property of the Hessian, the coordinate Newton direction for x_1 results in a deviation from the global optimum. This issue arises from the inaccurate approximation of Φ_1 in the restricted block coordinates x_1. The attractive force from vertex 2 and the repulsive forces from vertices 3 and 4 cancel each other out, leading to a highly inaccurate quadratic approximation. This deficiency cannot be entirely resolved by modifying Newton's method, as it is an intrinsic and unavoidable limitation of stochastic coordinate descent.

6.2.3 Ignorance of Other Vertices' Movements

Another reason is the ignorance of other vertices' movements when optimizing each vertex individually. When optimizing for a vertex i, the coordinate Newton direction treats all other vertices

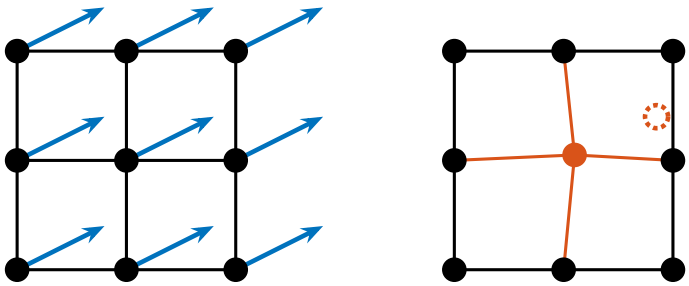


Figure 12: The blue arrows represent the acting forces in this situation. The center vertex exhibits only a minor shift in the coordinate Newton direction, while the red dashed vertex marks its ideal configuration.

$V \setminus \{i\}$ as fixed.

Fig. 12 illustrates this issue. Consider a subset of vertices forming a mesh-like structure in G, where all vertices receive forces in the directions indicated by the blue arrows. In this situation, the FR and L-BFGS algorithms move all vertices simultaneously, allowing the simulation or optimization to proceed without issues. In contrast, stochastic coordinate descent, as shown on the right side of the figure, brings stagnation. Here, all other vertices are considered fixed. Thus, optimizing the red vertex results in minimal movement, as its directly connected neighbors impede it. As a result, even after numerous iterations, little optimization is achieved. Thus, ignoring the movements of other vertices can be a significant limitation of the coordinate Newton method.

6.2.4 Rationale for Proposed Method

Our proposed method resolves issues in Secs. 6.2.2 and 6.2.3 by transforming Problem (1) into Problem (3) in Sec. 4.2, and optimizing Φ_i^{a} on Q^{hex}.

Regarding the issue in Sec. 6.2.2, this transformation brings the convexity of the objective function, as it treats the repulsive forces via a hexagonal lattice, making the quadratic approximation more accurate than the original problem. Regarding the issue in Sec. 6.2.3, the discrete point set Q ensures that the stepsize of any nonzero move is at least one, as each point is always separated by at least one. This helps avoid very small coordinate updates and can improve exploration of the feasible set. Furthermore, this transformation also brings the benefit of reducing computational complexity from $\mathcal{O}(|V|^2)$ to $\mathcal{O}(|E|)$.

Thus, converting the problem is crucial to provide a high-quality initial placement with the coordinate Newton direction. There may also be better transformation methods, and further exploration is warranted.

7 Conclusion

In this study, we proposed a new initial placement with the coordinate Newton direction for the FR force model, Problem (1). The obtained initial placements have fewer twists than the random initialization, leading to faster convergence and better visualization. Numerical experiments revealed that the proposed method is effective across various graphs, extending the applicability of the preprocessing step. The proposed method may advance graph drawing, and we also hope it

highlights the potential of stochastic coordinate descent and its variants for addressing a broader range of graph-related optimization problems.

Acknowledgements

We want to express our sincere gratitude to PL Poirion and Andi Han for their insightful discussions, which have greatly inspired and influenced this research. We also thank the developers of NetworkX and Graphviz. Their excellent work has been a great help in conducting this research.

References

[1] R. Ahmed, F. De Luca, S. Devkota, S. Kobourov, and M. Li. Multicriteria Scalable Graph Drawing via Stochastic Gradient Descent, $(SGD)^2$. *IEEE Transactions on Visualization and Computer Graphics*, 28(6):2388–2399, 2022. doi:10.1109/TVCG.2022.3155564.

[2] A. Arleo, W. Didimo, G. Liotta, and F. Montecchiani. A Distributed Multilevel Force-Directed Algorithm. *IEEE Transactions on Parallel and Distributed Systems*, 30(4):754–765, 2019. doi:10.1109/TPDS.2018.2869805.

[3] A. Arleo, S. Miksch, and D. Archambault. A Multilevel Approach for Event-Based Dynamic Graph Drawing. In M. Agus, C. Garth, and A. Kerren, editors, *EuroVis 2021 - Short Papers*, pages 103–107. The Eurographics Association, 2021. doi:10.2312/evs.20211063.

[4] G. Bartel, C. Gutwenger, K. Klein, and P. Mutzel. An Experimental Evaluation of Multilevel Layout Methods. In U. Brandes and S. Cornelsen, editors, *Graph Drawing*, pages 80–91. Springer, 2011. doi:10.1007/978-3-642-18469-7_8.

[5] C. Cartis, J. Fowkes, and Z. Shao. Randomised subspace methods for non-convex optimization, with applications to nonlinear least-squares, 2022. doi:10.48550/arXiv.2211.09873.

[6] H.-C. Chang and L.-C. Wang. A Simple Proof of Thue's Theorem on Circle Packing, 2010. doi:10.48550/arXiv.1009.4322.

[7] S.-H. Cheong and Y.-W. Si. Snapshot Visualization of Complex Graphs with Force-Directed Algorithms. In *2018 IEEE International Conference on Big Knowledge (ICBK)*, pages 139–145, 2018. doi:10.1109/ICBK.2018.00026.

[8] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL: https://igraph.org.

[9] T. A. Davis and Y. Hu. The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.*, 38(1):1:1–1:25, 2011. doi:10.1145/2049662.2049663.

[10] P. Eades. A heuristic for graph drawing. *Congressus Numerantium*, 42:149–160, 1984.

[11] P. Eades, C. Gutwenger, S.-H. Hong, and P. Mutzel. Graph drawing algorithms. In *Algorithms and Theory of Computation Handbook: Special Topics and Techniques*, pages 6–1–6–27. Chapman & Hall/CRC, 2 edition, 2010. doi:10.1201/9781584888215-c6.

[12] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull. Graphviz— Open Source Graph Drawing Tools. In P. Mutzel, M. Jünger, and S. Leipert, editors, *Graph Drawing*, pages 483–484. Springer, 2002. doi:10.1007/3-540-45848-4_57.

[13] C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. Yee. GraphAEL: Graph Animations with Evolving Layouts. In G. Liotta, editor, *Graph Drawing*, pages 98–110. Springer, 2004. doi:10.1007/978-3-540-24595-7_9.

[14] Y. Frishman and A. Tal. Online Dynamic Graph Drawing. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):727–740, 2008. doi:10.1109/TVCG.2008.11.

[15] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. doi:10.1002/spe.4380211102.

[16] T. Fuji, P.-L. Poirion, and A. Takeda. Theoretical analysis of the randomized subspace regularized Newton method for non-convex optimization. *Open Journal of Mathematical Optimization*, 6:1–35, 2025. doi:10.5802/ojmo.46.

[17] P. Gajdoš, T. Ježowicz, V. Uher, and P. Dohnálek. A parallel Fruchterman–Reingold algorithm optimized for fast visualization of large graphs and swarms of data. *Swarm and Evolutionary Computation*, 26:56–63, 2016. doi:10.1016/j.swevo.2015.07.006.

[18] F. Ghassemi Toosi, N. S. Nikolov, and M. Eaton. Simulated Annealing as a Pre-Processing Step for Force-Directed Graph Drawing. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, GECCO '16 Companion, pages 997–1000. Association for Computing Machinery, 2016. doi:10.1145/2908961.2931660.

[19] R. M. Gower, D. Kovalev, F. Lieder, and P. Richtárik. RSN: Randomized Subspace Newton. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 614–623. Curran Associates Inc., 2019. URL: https://dl.acm.org/doi/10.5555/3454287.3454343.

[20] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11–15, 2008. doi:10.25080/TCWV9851.

[21] H. Hamaguchi. HirokiHamaguchi/Initial Placement for Fruchterman–Reingold Force Model with Coordinate Newton Direction, 2026. URL: https://github.com/HirokiHamaguchi/Initial_Placement_for_Fruchterman-Reingold_Force_Model_with_Coordinate_Newton_Direction.

[22] D. Harel and Y. Koren. A Fast Multi-Scale Method for Drawing Large Graphs. *Journal of Graph Algorithms and Applications*, 6(3):179–202, 2002. doi:10.7155/jgaa.00051.

[23] R. Higuchi, P.-L. Poirion, and A. Takeda. Improving Convergence Guarantees of Random Subspace Second-order Algorithm for Nonconvex Optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL: https://openreview.net/forum?id=tuu4de7HL1.

[24] H. Hosobe. Numerical optimization-based graph drawing revisited. In *2012 IEEE Pacific Visualization Symposium*, pages 81–88, 2012. doi:10.1109/PacificVis.2012.6183577.

[25] Y. Hu. Efficient, high-quality force-directed graph drawing. *The Mathematica Journal*, 10(1):37–71, 2005. URL: https://yifanhu.net/GALLERY/GRAPHS/citation.html.

[26] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31(1):7–15, 1989. doi:10.1016/0020-0190(89)90102-6.

[27] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989. doi:10.1007/BF01589116.

[28] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006. doi:10.1007/978-0-387-40065-5.

[29] R. Nozawa, P.-L. Poirion, and A. Takeda. Randomized subspace gradient method for constrained optimization, 2023. doi:10.48550/arXiv.2307.03335.

[30] N. Okazaki. Chokkan/liblbfgs, 2024. URL: https://github.com/chokkan/liblbfgs.

[31] A. J. Patel. Hexagonal Grids. Technical report, Red Blob Games, 2013. URL: https://www.redblobgames.com/grids/hexagons/.

[32] Y. Qiu. Yixuan/LBFGSpp, 2024. URL: https://github.com/yixuan/LBFGSpp.

[33] M. Tiezzi, G. Ciravegna, and M. Gori. Graph Neural Networks for Graph Drawing. *IEEE Transactions on Neural Networks and Learning Systems*, 35(4):4668–4681, 2024. doi:10.1109/TNNLS.2022.3184967.

[34] D. Tunkelang. *A Numerical Optimization Approach to General Graph Drawing*. PhD thesis, Carnegie Mellon University, 1999. URL: https://csd.cmu.edu/sites/default/files/phd-thesis/CMU-CS-98-189.pdf.

[35] K. Ueda and N. Yamashita. A regularized Newton method without line search for unconstrained optimization. *Computational Optimization and Applications*, 59(1):321–351, 2014. doi:10.1007/s10589-014-9656-x.

[36] T. L. Veldhuizen. Dynamic Multilevel Graph Visualization, 2007. doi:10.48550/arXiv.0712.1549.

[37] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020. doi:10.1038/s41592-019-0686-2.

[38] C. Walshaw. A Multilevel Algorithm for Force-Directed Graph-Drawing. *Journal of Graph Algorithms and Applications*, 7(3):253–285, 2003. doi:10.7155/jgaa.00070.

[39] S. J. Wright and B. Recht. *Optimization for Data Analysis*. Cambridge University Press, 2022. doi:10.1017/9781009004282.

[40] J. X. Zheng, S. Pawar, and D. F. M. Goodman. Graph Drawing by Stochastic Gradient Descent. *IEEE Transactions on Visualization and Computer Graphics*, 25(9):2738–2748, 2019. doi:10.1109/TVCG.2018.2859997.

Algorithm 2: Simulation-Based (Fruchterman–Reingold) algorithm [15]

Input: Graph $G = (V, E)$, Weights $(a_{i,j})_{(i,j)\in E}$, Parameters $N_{\mathrm{iter}}^{\mathsf{SIM}} \in \mathbb{N}$, $t_0 > 0$.
Output: Final placement X

Initialize $X \in \mathbb{R}^{2\times n}$;
$t \leftarrow t_0$;
for $m \leftarrow 1$ **to** $N_{\mathrm{iter}}^{\mathsf{SIM}}$ **do**
  compute gradient $\nabla\Phi_i(x_i)$ for all $i \in V$;
  $x_i^{\mathrm{new}} \leftarrow x_i - t\frac{\nabla\Phi_i(x_i)}{\|\nabla\Phi_i(x_i)\|}$ for all $i \in V$;
  $x_i \leftarrow x_i^{\mathrm{new}}$;
  $t \leftarrow t - t_0/N_{\mathrm{iter}}^{\mathsf{SIM}}$;
  if *convergence condition is satisfied* **then**
    break;
return X;

A Additional Experimental Results

In addition to the evaluation in Sec. 5.2, we conducted experiments on several other graphs, with the results presented in Fig. 13. For the choice of graphs, we referred to [2, 4] and selected seven graphs: `cylinder_30_30`, `gr_30_30`, `sierpinski_06`, `jagmesh8`, `USpowerGrid`, `wiki-Vote`, and `crack`. The graph data were obtained from the SuiteSparse Matrix Collection [9] or generated by ourselves.

We can observe a similar trend to that in Fig. 6. For the simulation-based algorithm, the effect of the proposed method is significant, and its visualization results well reflect the structure of the graph. For L-BFGS algorithms, the positive effect is relatively smaller, but the proposed method generally achieves lower objective function values in most cases. It is also worth noting that the visualization results shown on the right were produced using a single random seed. They may appear worse than the expected performance shown in the plots. Developing methods that are more robust to random seed selection remains a direction for future work.

As a side note, for larger graphs, the computational cost of convergence for both simulation-based algorithms and L-BFGS becomes substantially higher, and even a single iteration can require a considerable amount of time. For further large-scale graphs with over ten thousand vertices, it will likely be necessary to combine the proposed method with a multilevel approach, as discussed in Sec. 6.1.

B NetworkX Implementation

As supplementary information, this section explains how to solve the energy minimization Problem (1) and obtain the final placement. We note that our implemented L-BFGS algorithm (without the proposed initialization) is available in NetworkX [20], and this section also serves as a reference for the details of the implementation in NetworkX.

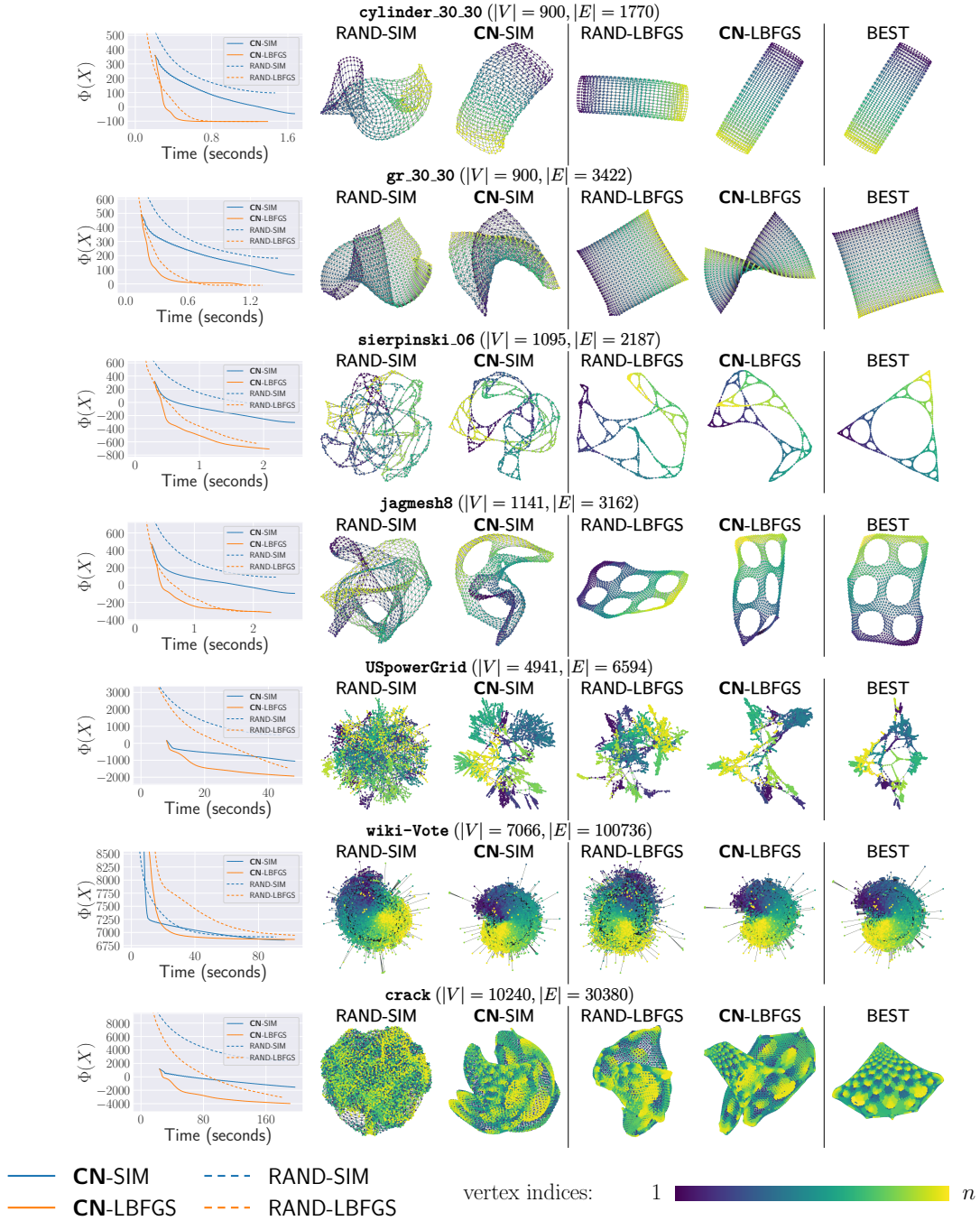


Figure 13: Additional experimental results for the FR force model. The objective function values at early states are generally improved for both simulation-based and L-BFGS algorithms, and the visual improvements by our proposed method CN are apparent for the simulation-based algorithms. The BEST column shows the reference layout obtained by running CN-LBFGS for 1000 iterations.

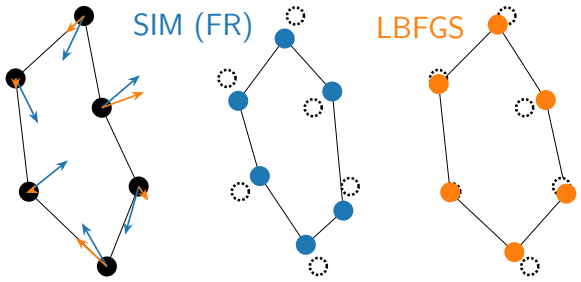


Figure 14: Comparison of the simulation-based and L-BFGS algorithms. Blue arrows represent fixed stepsizes in FR, whereas orange arrows represent adaptive stepsizes determined by the approximate inverse Hessian.

B.1 Simulation-Based Algorithm

The simulation-based algorithm, known as the Fruchterman–Reingold algorithm [15], is the original and most standard approach to obtain the equilibrium state of the force model. The pseudo-code is shown in Algorithm 2, which is a variant of the gradient (steepest) descent method with the function Φ_i in Eq. (6) [34].

Algorithm 2 is based on the original pseudo-code [15] and implementation in NetworkX [20] with some omitted details. We typically draw the initial placement X from a uniform distribution on $[0,1]^{2\times n}$. The parameter t denotes the temperature, which governs the stepsize along the steepest descent. As the temperature gradually decreases, the algorithm converges to a particular placement, which is not necessarily a local optimum of Problem (1).

B.2 L-BFGS Algorithm

Another approach is to leverage the L-BFGS algorithm [24]. Using only a few recent gradient vectors, the L-BFGS algorithm approximates the inverse Hessian of the objective function Φ [27]. L-BFGS is known to be very efficient for large-scale optimization problems. We can apply the L-BFGS algorithm by flattening the matrix $X \in \mathbb{R}^{2\times n}$ to a vector $\overline{X} \in \mathbb{R}^{2n}$. Refer to Fig. 14 for a comparison to the simulation-based algorithm.

B.3 NetworkX Implementation

In this subsection, we explain the details of our NetworkX implementation. Refer to Fig. 15 for a comparison to the FR algorithm. The problem setting considered in NetworkX is similar to that described in Sec. 2, but there are some key differences. First, the number of dimensions is not always two. Since the extension of the method is straightforward, we restrict our discussion to two-dimensional layouts. Other than that, the following differences exist:

- Negative edge weights are allowed.
- Directed graphs are supported, not just undirected graphs.
- The graph is not always connected.

Under these settings, the energy minimization problem (1) can be unbounded. It means that the optimal solution does not exist in a finite region. When edge weights are negative, or the graph

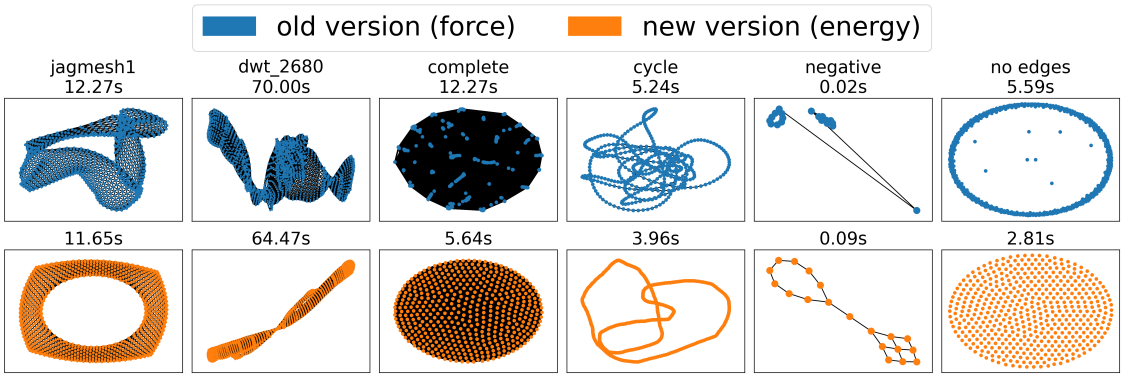


Figure 15: The simulation-based (FR) algorithm and L-BFGS algorithm in NetworkX.

is disconnected, i.e., when it consists of $n_{\mathrm{comp}} > 1$ connected components ($V = \bigcup_{j=1}^{n_{\mathrm{comp}}} V_j$), the solution may be unbounded, since the farther apart the vertices are, the lower the energy.

To overcome this issue, we do the following suitable modifications. First, when the graph has negative edge weights, one may transform them into non-negative values, for example, by taking their absolute values. Second, when the graph is a directed one, only the sum $a_{i,j} + a_{j,i}$ matters in the formulation, so symmetrization is sufficient. Thus, the directed and possibly negatively weighted adjacency matrix $\tilde{A} = (\tilde{a}_{i,j}) \in \mathbb{R}^{n \times n}$ can be transformed into a symmetric matrix $A = (a_{i,j})$, where $a_{i,j} = (|\tilde{a}_{i,j}| + |\tilde{a}_{j,i}|)/2 \geq 0$. Third, we add additional forces to the energy function to prevent the divergence of each connected component. In general, adding terms that attract each group to the center is a common approach to avoid unboundedness. We adopt this approach with the center $x_{\mathrm{center}} = (0.5, 0.5)^\top$, as it is the center of the expected layout bounding box.

Based on Sec. 2, let us define the sum of terms associated with the vertex x_i as Φ_i, which is explicitly given by

$$\Phi_i(x_i) \coloneqq \sum_{j \in V \setminus \{i\}} \left(\Phi_{i,j}(\|x_i - x_j\|) + \Phi_{j,i}(\|x_j - x_i\|)\right).$$

The i-th column of the gradient $\nabla\Phi(X) \in \mathbb{R}^{2 \times n}$ is

$$\begin{aligned}(\nabla\Phi(X))_i = \nabla\Phi_i(x_i) &= \sum_{j \in V \setminus \{i\}} \left(\frac{(a_{i,j} + a_{j,i})\|x_i - x_j\|}{k} - \frac{2k^2}{\|x_i - x_j\|^2}\right)(x_i - x_j) \\ &= 2 \sum_{j \in V \setminus \{i\}} \left(\frac{a_{i,j}\|x_i - x_j\|}{k} - \frac{k^2}{\|x_i - x_j\|^2}\right)(x_i - x_j).\end{aligned}$$

To obtain the final layout while preventing the divergence, we need to minimize $\Phi(X) + \Phi_{\mathrm{g}}(X)$, where $\Phi_{\mathrm{g}}(X)$ is the additional term. Let the center of gravity of the j-th connected component V_j ($1 \leq j \leq n_{\mathrm{comp}}$) be $g_j \coloneqq \frac{1}{|V_j|} \sum_{i \in V_j} x_i$. Then, using some constant $c_{\mathrm{g}} > 0$, we define the additional energy term as

$$\Phi_{\mathrm{g}}(X) \coloneqq c_{\mathrm{g}} \sum_{j=1}^{n_{\mathrm{comp}}} \frac{|V_j|}{2} \|g_j - x_{\mathrm{center}}\|_2^2,$$

with the gradient

$$(\nabla \Phi_{\mathrm{g}}(X))_i = c_{\mathrm{g}}(g_j - x_{\mathrm{center}}) \quad \text{where} \quad i \in V_j.$$

We can now apply the L-BFGS algorithm to these functions to derive the algorithm. Further implementation details and experimental results are available in our merged pull request to NetworkX.