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

Hiroki Hamaguchi Naoki Marumo Akiko Takeda

Abstract—Graph drawing is a fundamental task in information visualization, with the Fruchterman–Reingold (FR) force model being one of the most popular choices. We can interpret this visualization task as a continuous optimization problem, which can be solved using the FR algorithm, the original algorithm for this force model, or the L-BFGS algorithm, a quasi-Newton method. However, both algorithms suffer from twist problems and are computationally expensive per iteration, which makes achieving high-quality visualizations for large-scale graphs challenging.
In this research, we propose a new initial placement based on the 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 move a randomly selected vertex along the coordinate Newton direction. We can use the FR or L-BFGS algorithms to obtain the final placement.
We demonstrate the effectiveness of our proposed approach through experiments, highlighting the potential of coordinate descent methods for graph drawing tasks. Additionally, we suggest combining our method with other graph drawing techniques for further improvement. We also discuss the relationship between our proposed method and broader graph-related applications.

Index Terms—Graph Drawing, Optimization, Fruchterman–Reingold Algorithm, L-BFGS algorithm

1 Introduction

Graph is a mathematical structure representing pairwise relationships between objects, and graph drawing is a fundamental task in information visualization. Indeed, numerous kinds of models and algorithms have been proposed, and among these, one of the most popular choices is force-directed graph drawing.

In force-directed graph drawing, the force model is composed of particles with forces acting between them. The equilibrium of these forces is considered suitable for graph visualization, and algorithms aim to find this equilibrium state. Among the force models [6, 19], the Fruchterman–Reingold (FR) force model [8, 21] is the most prominent one, regarded as flexible, intuitive, and simple. It is widely used in various graph drawing libraries such as NetworkX [14], Graphviz [7], and igraph [4].

Contrary to the advantages above, the algorithms using this model face challenges that make producing high-quality visualizations for large-scale graphs challenging. The most critical issue is that *twist* slows down the simulation process. The term twist refers to unnecessary folded and tangled structures in the visualized graph [3, 31]. The results by the FR and L-BFGS algorithms from a random initial placement shown in Fig. 1 highlight these twist issues. Although the optimal graph structure is simple, the twists are likely to occur and weaken or diminish the forces, leading to stagnation and suboptimal visualization outcomes. The FR algorithm is proposed alongside the force model and the most commonly used approach, but the results are excessively twisted. The L-BFGS algorithm, a family of quasi-Newton methods, has been reported as a more practical approach for graph drawing [17] and achieves better results than the FR algorithm. While this algorithm can partially address the twist problem, it sometimes requires many iterations.

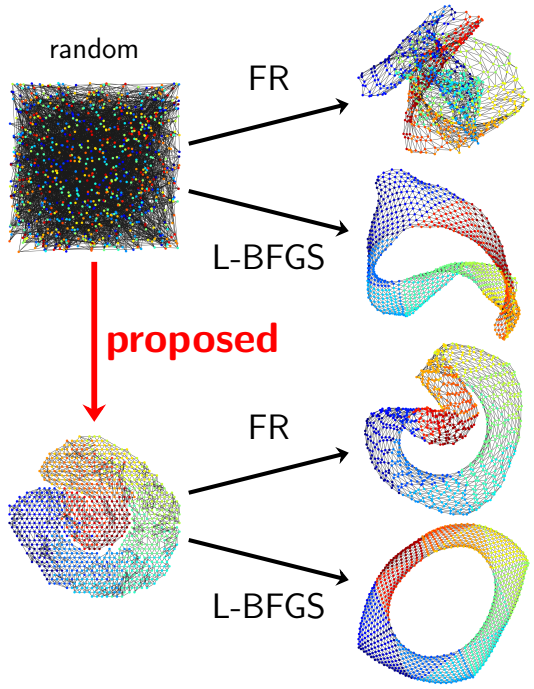


Fig. 1: Comparison of the algorithms for the `jagmesh1`.

As shown in the figure, it may fail to achieve optimal visualization within a limited number of iterations. Further, these algorithms suffer from high computational complexity when directly applied, $\mathcal{O}\left(|V|^2\right)$ per iteration, where $|V|$ is the number of vertices.

Another approach to the twist problem is to provide initial placement in the pre-processing step. Indeed, a pre-

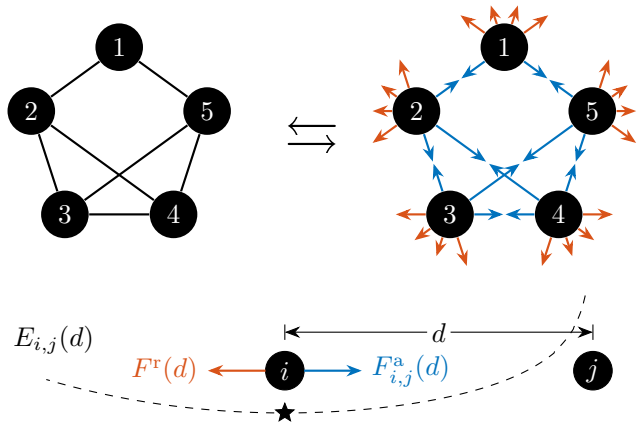


Fig. 2: (Top) The illustration of the force model. Forces act on every pair of vertices. (Bottom) Forces $F^{\mathrm{a}}_{i,j}(d)$ and $F^{\mathrm{r}}(d)$ work between vertices i and j. The equilibrium of them is achieved at $d = k/\sqrt[3]{w_{i,j}}$, which equals k when $w_{i,j} = 1$.

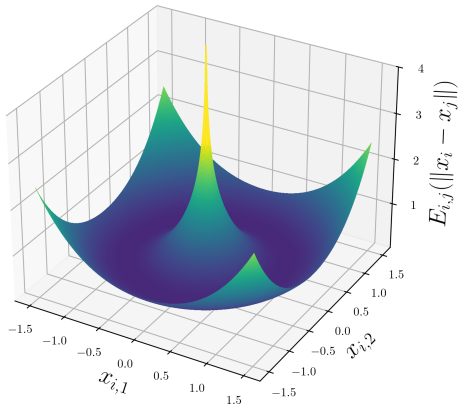


Fig. 3: Energy function $E_{i,j}(\|x_i - x_j\|)$ for $x_i = (x_{i,1}, x_{i,2})$, $x_j = (0, 0)$, $w_{i,j} = 1$, and $k = 1$.

processing step with Simulated Annealing (SA) is also known to be effective [11] since SA can avoid getting stuck in local optima and leads to a better visualization combined with the FR algorithm. Regrettably, this work only uses a circle initial placement for unweighted graphs with a simple method, leaving significant room to improve the effectiveness and extend the applicability. Refer to Sec. 3.2 for more details.

In this paper, we propose a new initial placement for the FR force model as depicted in Fig. 1. We provide an initial placement with fewer twists than random placement within a short time, accelerating the subsequent optimization process. We can use both FR 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 structure 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.

The rest of this paper is organized as follows. Sec. 2 introduces the FR force model. Sec. 3 reviews the related works. Sec. 4 proposes our initial placement algorithm. Sec. 5 shows the experimental results. Sec. 6 provides the rationale of our proposed algorithm. Finally, Sec. 7 discusses and concludes the paper.

2 Fruchterman–Reingold Force Model

Fruchterman and Reingold [8] proposed the FR force model for graph drawing based on the physical analogy of the system of particles. Through the simulation of these forces, the FR algorithm seeks the equilibrium positions. In contrast to this ordinary approach, we minimize the energy, also known as the stress, to seek equilibrium. This section reviews the model and clarifies the problem we solve.

Let $G = (V, E)$ be a connected undirected graph with vertex set $V = \{1, \dots, n\}$ and edge set E. Each edge $\{i, j\} \in E$ has weight $w_{i,j} > 0$. For convenience, we set $w_{i,j} = 0$ for $\{i, j\} \notin E$.

The FR force model assumes forces between vertices. For vertices i and j with a distance $d > 0$ between them, an attractive force $F^{\mathrm{a}}_{i,j}(d)$ and a repulsive force $F^{\mathrm{r}}(d)$ work as:

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

where $k > 0$ is a constant parameter, often set to $1/\sqrt{n}$. The scalar potential of these forces [17] is given by

$$\begin{aligned} E^{\mathrm{a}}_{i,j}(d) &\coloneqq \int_0^d F^{\mathrm{a}}_{i,j}(r)\,\mathrm{d}r = \frac{w_{i,j} d^3}{3k}, \\ E^{\mathrm{r}}(d) &\coloneqq \int_\infty^d F^{\mathrm{r}}(r)\,\mathrm{d}r = -k^2 \log d, \\ E_{i,j}(d) &\coloneqq E^{\mathrm{a}}_{i,j}(d) + E^{\mathrm{r}}(d). \end{aligned}$$

For simplicity, we define $E_{i,j}(0) = \infty$. Thus, the problem is to minimize the energy with positions $X \coloneqq (x_1, \dots, x_n) \in \mathbb{R}^{2\times n}$:

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad f(X) \coloneqq \sum_{i<j} E_{i,j}(\|x_i - x_j\|). \tag{1}$$

The local minimum of f yields the equilibrium positions since $\nabla f(X)$ corresponds to the forces. Refer to Fig. 2 for the explanation.

As mentioned, we will only consider undirected connected graphs with non-negative weights. Although some algorithms can handle directed unconnected graphs with negative weights, we do not focus on such cases. For directed graphs, slight modifications of algorithms or converting graphs to undirected ones can be effective. For unconnected graphs, algorithms can be applied to each connected component independently. When negative weights are present, the optimization problem (1) can be unbounded, but with non-negative weights and the connectivity of G, the problem is always bounded and solvable.

While the energy function $E_{i,j}$ is convex and minimized when $d = k/\sqrt[3]{w_{i,j}}$, a function $x_i \mapsto E_{i,j}(\|x_i - x_j\|)$ is non-convex for a fixed x_j. $\|\cdot\|$ denote the Euclidean norm in $\mathbb{R}^2$. Additionally, $E_{i,j}$ is not Lipschitz continuous as it diverges when $d \to 0$. These properties highlight the difficulty of the problem. Refer to Fig. 3 for an illustration of $E_{i,j}$.

3 Related Works

We briefly introduce some critical related works.

3.1 Algorithms for FR Force Model

As mentioned in Sec. 1, we can use the FR and L-BFGS algorithms to visualize graphs with the FR force model. These algorithms solve the problem (1), and both can be applied to the initial placement we provide. For details, refer to Sec. A.

3.2 Pre-Processing by Simulated Annealing

Let $Q^{\mathrm{circle}} := \{(\cos(2\pi i/n), \sin(2\pi i/n)) \mid 1 \le i \le n\}$ 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 2. 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 defines the problem for pre-processing as follows (we change some notations for consistency):

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

The problem (2) is a discrete optimization problem where the placement is limited to Q^{circle} and uses angles, not the function f. Ref. [11] obtain a faster and better visualization by setting the result of Simulated Annealing (SA) for the problem (2) as an initial placement for the FR 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 structure graphs.
- The only neighborhood in the SA is the random swapping of two vertices, making the optimization process inefficient for large-scale graphs.
- $|E_2|$ could be $\Theta(n^2)$, unable to leverage the sparsity of graphs if it exists.

We are dealing with these limitations and can regard our study as an extension of this prior work.

3.3 Graph Drawing by Stochastic Gradient Descent

When we regard graph drawing as an optimization problem, Stochastic Gradient Descent (SGD) for Kamada–Kawai (KK) layout [19] is one of the most notable works [33]. In the KK layout, we regard G as a complete graph and assign the energy function $E^{\mathrm{KK}}_{i,j}$ to all edges. SGD in this context means to repeat randomly selecting an edge $\{i, j\}$ and updating x_i and x_j with the gradient of $E^{\mathrm{KK}}_{i,j}$. This algorithm is known to be effective in solving various optimization problems.

Although applying SGD to the FR force model problem (1) is straightforward, it is ineffective for this problem. This is because the force model we consider assigns the same function $E_{i,j}(d) = -k^2 \log d$ to all $\{i, j\}$ such that $w_{i,j} = 0$. Optimizing $E_{i,j}$ only increases the distance between vertices i and j, no matter how close they are to each other in the optimal solution. Thus, the gradient of $E_{i,j}$ is not informative enough to find the optimal solution, and we need to develop a new optimization method for the problem (1).

Still, the idea of randomly selecting an edge and updating its position is quite suggestive. Based on this idea, we propose to randomly select a vertex and update its position.

3.4 Newton Direction and Coordinate Newton Direction

Let us consider a strictly convex function $f\colon \mathbb{R}^n \to \mathbb{R}$ at x_0. The Newton direction $d = -\nabla^2 f(x_0)^{-1}\nabla f(x_0)$ is an optimal direction for the second order approximation of f:

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

$x = x_0 + d$ is the minimizer of this approximation. Note that the Hessian matrix $\nabla^2 f(x_0)$ is positive definite since f is strictly convex. Although the Newton direction provides a critical step in iterative methods, it requires the computation of the inverse Hessian $\nabla^2 f(x_0)^{-1} \in \mathbb{R}^{n\times n}$, 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 $\nabla^2 f(x_0)^{-1}$ in the entire variable space $\mathbb{R}^n$, we limit the variable x to its coordinate block x_i with fewer dimensions, and compute $\nabla^2 f_i(x_i)^{-1}\nabla f_i(x_i)$ where f_i is a restricted function of f 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 [29] or Randomized Subspace Newton (RSN) [12] in a broader context.

In particular, this coordinate Newton direction has an apparent natural affinity to the problem (1) in Sec. 2. 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 problem (1) is challenging, as we will discuss in Sec. 6, we leverage this coordinate Newton direction to propose our algorithm.

4 Proposed Algorithm

This section proposes a new initial placement algorithm for the problem (1). We define the discrete optimization problem for initial placement and propose a new algorithm based on stochastic coordinate descent.

4.1 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 problem (1) into a more manageable and well-behaved discrete optimization problem:

$$\begin{array}{lll} \underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} & \displaystyle\sum_{\{i,j\}\in E} \frac{w_{i,j}\|x_i - x_j\|^3}{3k}, & \\ \text{subject to} & x_i \in Q^{\mathrm{hex}} & \text{for } 1 \le i \le n, \\ & x_i \neq x_j & \text{for } 1 \le i < j \le n. \end{array} \tag{3}$$

where

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

This section explains how this simplification is derived.

First, the problem (1) is equivalent to the following:

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

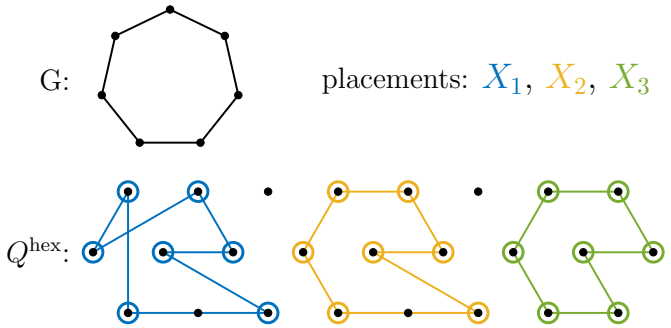


Fig. 4: Concept of Q. The assignment from V to a discrete point placement Q, especially a hexagonal lattice Q^{hex}. Apparently, among X_1, X_2, X_3, the right one X_3 is the best placement for the problem (3).

This formalism separates the $\mathcal{O}(|E|)$ terms from $E^{\mathrm{a}}_{i,j}$ and the $\mathcal{O}\left(|V|^2\right)$ terms from E^{r}.

Here, 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 $|Q| \geq |V|$. It means that we consider the following:

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

The goal is to simplify the objective function and choose Q appropriately to derive a well-simplified problem.

Due to the sparsity of many practical graphs, $|E| \ll |V|^2$ holds. We want to leverage this sparsity for simplification. To do that, we have to drop the second term that arose from the repulsive energy E^r:

$$-\sum_{i<j} k^2 \log \|x_i - x_j\|.$$

We impose a condition to Q such that $\|q_i - q_j\| \geq \epsilon$ for all $q_i, q_j \in Q (q_i \neq q_j)$. In this case, the term above is negligible. Since $E^{\mathrm{r}}(d) = -k^2 \log d$ is a convex function such that it decreases monotonically concerning d, for sufficiently large d, the value of $-k^2 \log d$ does not vary excessively. For too small d, we can prevent the divergence of the energy function by setting ϵ. Thus, under this condition, we drop the second term and derive the problem as:

$$\begin{aligned} \underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad & f^{\mathrm{a}}(X) \coloneqq \sum_{\{i,j\}\in E} \frac{w_{i,j}\|x_i - x_j\|^3}{3k}, \\ \text{subject to} \quad & x_i \in Q \quad \text{for } 1 \leq i \leq n, \\ & x_i \neq x_j \quad \text{for } 1 \leq i < j \leq n \end{aligned} \tag{4}$$

This means that we skip to consider the $\mathcal{O}\left(|V|^2\right)$ 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.

We can consider various placements for the Q above, including Q^{circle} [11]. In this study, we adopt a hexagonal lattice Q^{hex} [23, 27]. When minimizing the attraction energy

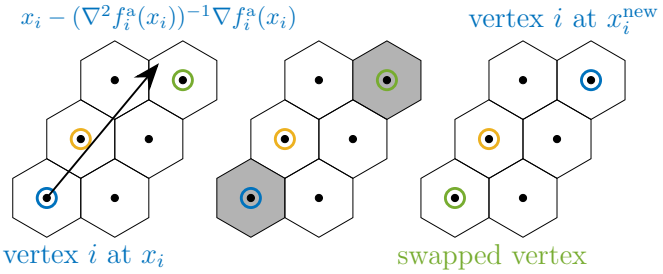


Fig. 5: One iteration of the proposed algorithm. Step1. Compute the coordinate Newton direction for a randomly selected vertex (blue). Step2. Decide x_i^{new} by rounding the direction and adding a random vector. Step3. Move the vertex and swap the vertices if there is a collision. In this case, swap blue and green vertices.

f^{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 with the least distance ϵ between points and offers computational simplicity. Thus, Q^{hex} is a suitable choice, and we have derived the problem (3).

4.2 Newton Direction for Discrete Optimization

Next, we solve the discrete optimization problem (3). Although this problem is challenging to solve just using simple methods, the coordinate Newton direction of a randomly selected vertex i provides significant insights, as mentioned in Sec. 3.4. Therefore, we can provide an efficient solution to the problem. Let the limited objective function $f_i^{\mathrm{a}}(x_i)$ be

$$f_i^{\mathrm{a}}(x_i) \coloneqq \sum_{j\neq i} \frac{w_{i,j}\|x_i - x_j\|^3}{3k}.$$

Its gradient and Hessian matrix are

$$\begin{aligned} \nabla f_i^{\mathrm{a}}(x_i) &= \sum_{j\neq i} \frac{w_{i,j}\|x_i - x_j\|}{k}(x_i - x_j), \\ \nabla^2 f_i^{\mathrm{a}}(x_i) &= \sum_{j\neq i} \frac{w_{i,j}\|x_i - x_j\|}{k}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ &\quad + \sum_{j\neq i} \frac{w_{i,j}}{k\|x_i - x_j\|}(x_i - x_j)(x_i - x_j)^\top. \end{aligned}$$

This means f_i^{a} is strictly convex, assuring the Hessian matrix $\nabla^2 f_i^{\mathrm{a}}(x_i)$ is positive definite. This is a large difference from the functions $f_i(x_i)$ in Eq. (6) and $f^{\mathrm{a}}(X)$ in the problem (4), which are non-convex.

The ordinary updated rule with the coordinate Newton direction is

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

x_i^{new} may not be in the hexagonal lattice Q^{hex} in the problem (3). Thus, we must round this value to the nearest point in Q^{hex}. We also empirically found that adding a random noise vector to the Newton 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 updated rule for the vertex i is

$$x_i^{\mathrm{new}} \leftarrow \mathrm{round}\big(x_i - \nabla^2 f_i^{\mathrm{a}}(x_i)^{-1} \nabla f_i^{\mathrm{a}}(x_i) + t \cdot \mathrm{rand}\big),$$

where $\mathrm{round}(\hat{x})$ denotes the operation assigning $\hat{x}$ to the nearest point in the hexagonal lattice Q^{hex}, rand is a random vector with a unit norm, and t is a parameter controlling the randomness converges to 0.

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 solves the problem (3).

4.3 Optimal Scaling

Finally, the obtained solution could be too small or too large since we did not care about the scale ϵ. Thus, as the final step, we rescale the placement to obtain the initial placement. This subsection explains how to find the optimal scaling factor c^* that minimizes f for a given placement.

Let us formulate the optimization problem for the scaling factor $c > 0$. For an initial placement $X = (x_1, \ldots, x_n)$, we scale it as $x_i \leftarrow cx_i$ for all i. This problem is to minimize the energy function $\phi(c)$ defined by

$$\phi(c) := \left(\sum_{\{i,j\} \in E} \frac{w_{i,j}(c\|x_i - x_j\|)^3}{3k} \right) - k^2 \sum_{i<j} \log(c\|x_i - x_j\|)$$

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

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

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

This value can be computed in $\mathcal{O}(|E|)$ complexity, enabling us to obtain a better initial placement for the problem (1).

Notably, the optimal solution to the problem (3) is invariant under scaling. Thus, we can select any ϵ to define the hexagonal lattice Q^{hex} as far as we scale by c^*.

4.4 Pseudo Code

We presented the overall framework of the proposed method in Algorithm 1. The proposed algorithm is not a complete solution to the problem (1); this only provides an initial placement for the FR or L-BFGS algorithms.

4.5 Alternative Approach

To end this section, we explain an alternative approach to this algorithm. We can also optimize by updating all vertices simultaneously, not one by one. It means moving all the points $\{x_i\}_{i \in V}$ to arbitrary points and then assigning all these $|V|$ points to the nearest points. If appropriately defined, we can solve the assignment problem by minimum-cost flow or Hungarian algorithm with $\mathcal{O}\left(|V|^3\right)$ complexity. Additionally, a heuristic solution can be obtained in $\mathcal{O}(|V| \log |V|)$ by appropriately sorting.

They offer advantages such as simplified implementation or avoiding random access to arrays.

Algorithm 1: Proposed algorithm as initial placement

Input: Graph $G = (V, E)$, Weight $(w_{i,j})_{\{i,j\} \in E}$, Parameters $N_{\mathrm{iter}}^{\mathsf{CN}} \in \mathbb{N}$, $t_0 > 0$
Output: Initial placement $X = (x_1, \ldots, x_n)$

```
t ← t_0;
Sample x_i ∈ Q for all i ∈ V without replacement;
for m ← 0 to N_iter^CN do
    Select vertex i ∈ V randomly;
    x_i^new ← round(x_i − ∇²f_i(x_i)^{-1} ∇f_i(x_i) + t·rand);
    if ∃j ∈ V s.t. x_j = x_i^new then
        Swap x_i and x_j;
    else
        x_i ← x_i^new;
    t ← t − t_0/N_iter^CN;
x_i ← c* x_i for all i ∈ V with c* by Eq. (5);
return X
```

5 Numerical Experiment

In this section, we evaluate the proposed algorithm using various numerical experiments. We also confirm that the proposed algorithm extends the applicability of the preprocessing step in Ref. [11], as mentioned in Sec. 1.

5.1 Experimental Setup

We conducted all numerical experiments in this section using C++17 compiled by GCC 10.5.0 on a laptop computer powered by Intel(R) Core(TM) i7-10510U CPU with 16 GB RAM.

For the FR and L-BFGS algorithms, we referenced NetworkX version 3.3 [14], SciPy 1.14.1 [32], and C++ L-BFGS [26, 28]. In particular, we used almost the same parameters as NetworkX's spring_layout for the FR algorithm. We also referenced the open-source code of the hexagonal grid from [27] and Graphviz version 2.43.0 [7].

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

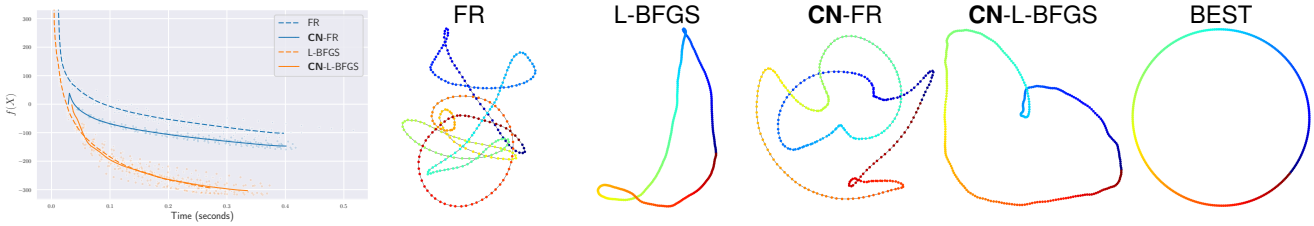
- Random initialization (no prefix),
- The circle initialization obtained by Simulated Annealing (SA-) [11],
- The proposed initialization obtained with coordinate Newton direction (CN-).

As an algorithm to solve the problem (1), we used

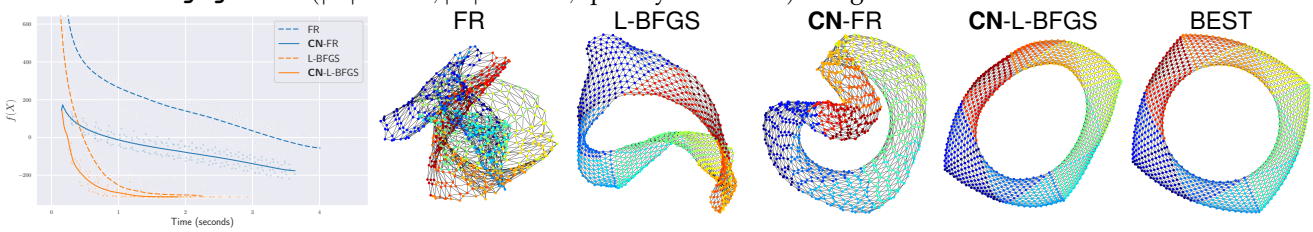
- FR algorithm (FR),
- L-BFGS algorithm (L-BFGS).

As parameters, we used $N_{\mathrm{iter}}^{\mathsf{CN}} = 2|V|^3/|E|$, $t_0 = 1.5$ for Algorithm 1, and we also set the total number of iterations of Simulated Annealing (SA) in Sec. 3.2 to the same value. Since the amortized time complexity per iteration of Algorithm 1 is $\mathcal{O}(|E|/|V|)$, we can roughly expect that the computational time of the proposed algorithm is $\mathcal{O}\left(2|V|^2\right)$, equivalent to a few iterations of the FR or L-BFGS algorithm. All the codes are available at GitHub [16].

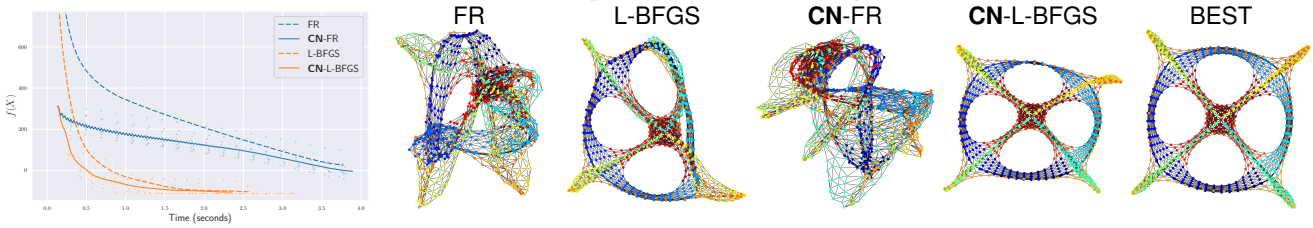
cycle300 ($|V| = 300, |E| = 300, \text{sparsity} = 0.669\%$) Figures are at 150 iterations.



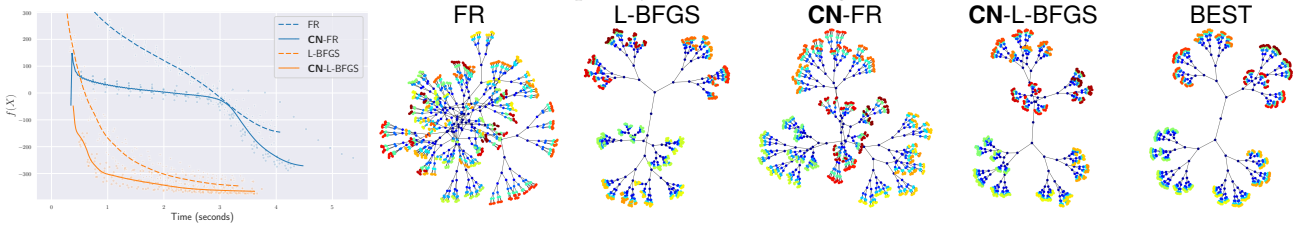
jagmesh1 ($|V| = 936, |E| = 2664, \text{sparsity} = 0.609\%$) Figures are at 50 iterations.



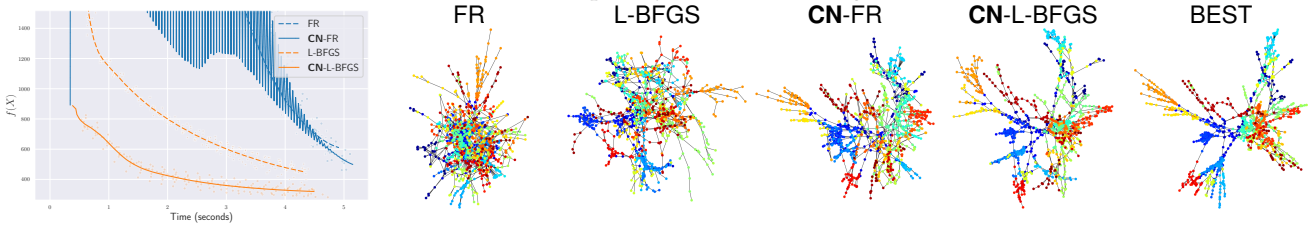
dwt_1005 ($|V| = 1005, |E| = 3808, \text{sparsity} = 0.755\%$) Figures are at 100 iterations.



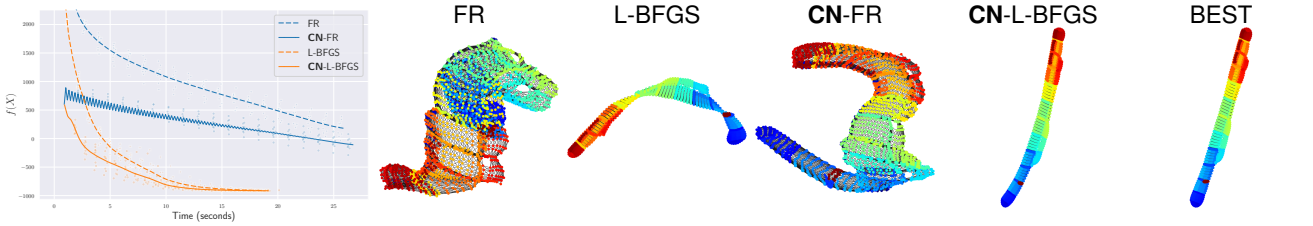
btree9 ($|V| = 1023, |E| = 1022, \text{sparsity} = 0.196\%$) Figures are at 150 iterations.



1138_bus ($|V| = 1138, |E| = 1458, \text{sparsity} = 0.225\%$) Figures are at 150 iterations.



dwt_2680 ($|V| = 2680, |E| = 11173, \text{sparsity} = 0.311\%$) Figures are at 150 iterations.



3elt ($|V| = 4720, |E| = 13722, \text{sparsity} = 0.123\%$) Figures are at 150 iterations.

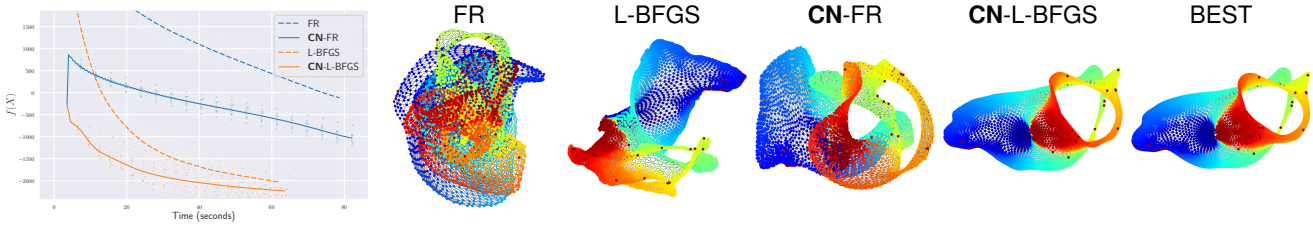


Fig. 6: Numerical experiment results for various graphs. Please refer to Sec. 5.2 for details.

5.2 detailed Experiment

We conducted a detailed experiment to investigate the behavior of the proposed algorithm in detail. Fig. 6 shows the results.

The experiment details are as follows. We fix the maximum number of iterations of the FR and L-BFGS algorithms to $N_{\text{iter}}^{\mathsf{FR}}$ and $N_{\text{iter}}^{\mathsf{L-BFGS}}$, respectively. We set $N_{\text{iter}}^{\mathsf{FR}} = N_{\text{iter}}^{\mathsf{L-BFGS}} = 200$ in this experiment. We tested with 7 graphs: `cycle300`, `jagmesh1`, `dwt_1005`, `btree9`, `1138_bus`, `dwt_2680`, and `3elt`. `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 Sparse Matrix Collection [5], and these choices are based on the experiments conducted in Ref. [33]. Thus, although all the graphs are quite sparse so that $|E|/(|V|(V-1)/2)$ is less than 1%, this is not an arbitrary choice. The important graphs often have such a sparsity.

We first explain what Fig. 6 represents. The plots on the left illustrate the objective function values $f(X)$ on the vertical axis versus execution time on the horizontal axis, using ten trials for each algorithm. Faint crosses represent the results with random initialization, while faint circles represent the results of CN, plotted every ten iterations for each trial. The solid and dashed lines represent the average of these values across the ten trials for each algorithm. If one of the trials terminated before reaching $N_{\text{iter}}^{\mathsf{FR}}$-th or $N_{\text{iter}}^{\mathsf{L-BFGS}}$-th iterations, we plotted up to the minimum trial count achieved across all trials. Each trial was conducted with a different seed. The graphs on the right are at the iteration in which the most significant difference appeared among {50,100,150}-th iterations (or at the last iteration if it ended earlier), using seed 1. We obtained the "BEST" column by running at most 500 iterations of CN-L-BFGS. The colors of the vertices in the graphs presented in this section are assigned according to vertex indices using a color map that provides a gradient from blue to red.

The observations and implications of Fig. 6 are as follows. First, the solid plots for CN generally demonstrate superior performance compared to non-CN, validating the efficacy of the proposed method. Some exceptions of the superiority arise with FR, which exhibits oscillations in the plot, likely due to excessive stepsizes leading to overshooting. Although we refrained from altering FR for fairness, adjusting the stepsize (or using adaptive stepsize) could enable the proposed method to achieve its intended performance. Still, the initial $f(X)$ of CN-FR is small enough compared to the objective function values produced by FR alone, suggesting that the proposed method is yielding a good initial placement. Additionally, the visualization results support the effectiveness of the proposed initial placement. In most cases, placements obtained with CN better represent the intended geometric arrangement shown in the "BEST".

As a side note, regardless of CN or non-CN, the algorithms using L-BFGS consistently outperform those using FR. This finding is consistent with prior research [17], though regrettably, this technique remains relatively unknown in graph drawing. One of the aims of our paper is to emphasize further and popularize the use of the L-BFGS algorithm in graph drawing, and these results provide substantial proof for this argument.

5.3 Comparison with Other Initializations

Next, we conducted exhaustive experiments to evaluate the performance of the proposed algorithm (CN) compared to random initialization (no prefix) and circle initialization (SA) with various graphs.

Let $W = (w_{i,j}) \in \mathbb{R}_{\geq 0}^{n \times n}$ be an adjacency matrix of a graph G. Since we only consider undirected connected graphs with non-negative weights, W should satisfy the following conditions:

$$W \in \mathbb{R}_{\geq 0}^{n \times n}, \quad W = W^{\top}, \quad \text{and} \quad G \text{ is connected.}$$

We used matrices from Sparse Matrix Collection [5] satisfying the above condition with $|V| \leq 1000$ as the adjacency matrix W, in total 124 graphs. For fairness, when we compare with SA, we converted the graph to an unweighted one. We set weights $w_{i,j}$ to 1 if $w_{i,j} > 0$; otherwise, we set it to 0.

We set $N_{\text{iter}}^{\mathsf{FR}}$ and $N_{\text{iter}}^{\mathsf{L-BFGS}}$ as 50, the default parameter of NetworkX [14]. For the CN algorithms compared with random initialization, We set them as 45 since it contains the pre-processing step. We can roughly expect that the computational time is equivalent to a few iterations of FR or L-BFGS algorithms, as mentioned in Sec. 5.1.

The results are shown in Fig. 7 and Fig. 8. In almost all cases, the proposed algorithm performed better than random or circle initialization. A few cases where the proposed algorithm performed worse than other methods. As for random initialization, one of such cases is `Spectro_10NN` shown in Fig. 9. The proposed algorithm might fail to resolve the twist in the initial placement, shown in the figure, leading to a worse result. Still, the average performance for this case was almost the same as that of the random initialization. As for circle initialization, we showed examples in Fig. 11. The proposed algorithm outperforms in `collins_15NN` and underperforms in `dwt_992`.

The interpretation of the results is as follows. First, this result strongly supports the effectiveness of the proposed algorithm. It successfully untangles the twists, leading to better outcomes with faster convergence. Even when the proposed algorithm performed worse, the difference was insignificant in almost all cases. Secondly, the choice of initial placement can lead to advantages or disadvantages depending on the optimal placement. In particular, since the optimal shape of `collins_15NN` is a linear shape, which differs from a circle, `SA`'s performance can be worse than the proposed algorithm. Such difference likely contributed to the advantage of the proposed algorithm. Although the proposed method uses a hexagonal lattice Q^{hex}, it can be adapted to other fixed placement Q. If the shape of the optimal solution is roughly known in advance, selecting a different Q, such as a Q^{circle}, could further enhance the performance of the proposed algorithm.

Additionally, although this is a case not considered in Ref. [11], we also conducted experiments with CN and SA for a case where the edge weight $w_{i,j}$ is not necessarily in $\{0, 1\}$. We made a weighted graph with 100 vertices, 1000 edges, and three groups of vertices. We generated every edge 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. It exhibits both strong and weak connections. Fig. 10

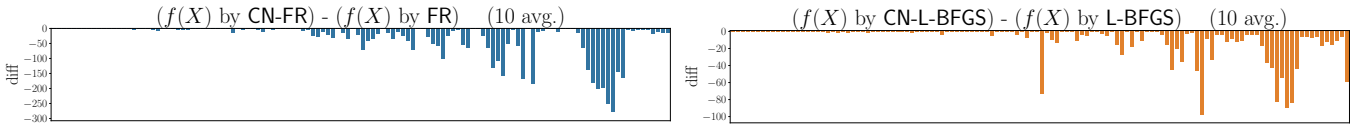


Fig. 7: Comparison of the proposed initialization (CN) with random initialization (no prefix). In almost all cases, the difference is negative, meaning the proposed algorithm performed faster and better than random initialization.

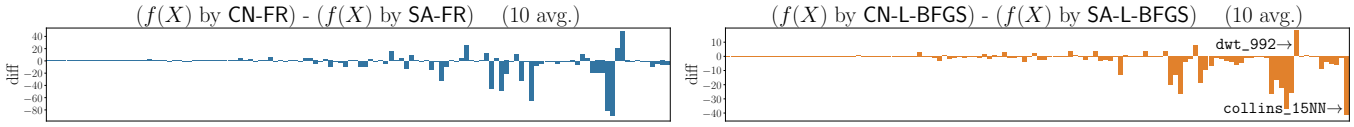


Fig. 8: Comparison of the proposed initialization (CN) with circle initialization (SA) in Ref. [11]. In most cases, the proposed algorithm performed better than the circle initialization.

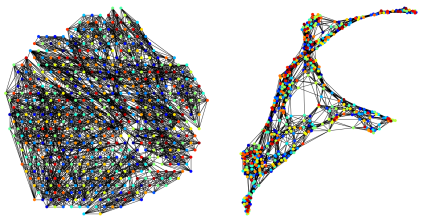


Fig. 9: An example of a case where the proposed algorithm performed worse than random initialization. Left: Initial placement by CN. Middle and Right: 50th and 200th iteration of CN-L-BFGS.

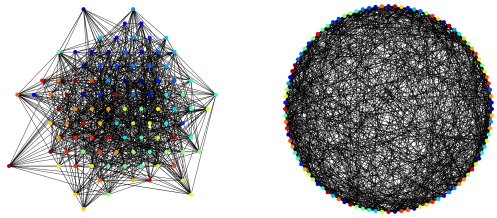


Fig. 10: An example of a case where $w_{i,j} \notin \{0, 1\}$. This result shows the proposed algorithm is extending the applicability of the pre-processing step. Left: the result of CN. Separated by colors. Right: the result of SA. No separation.

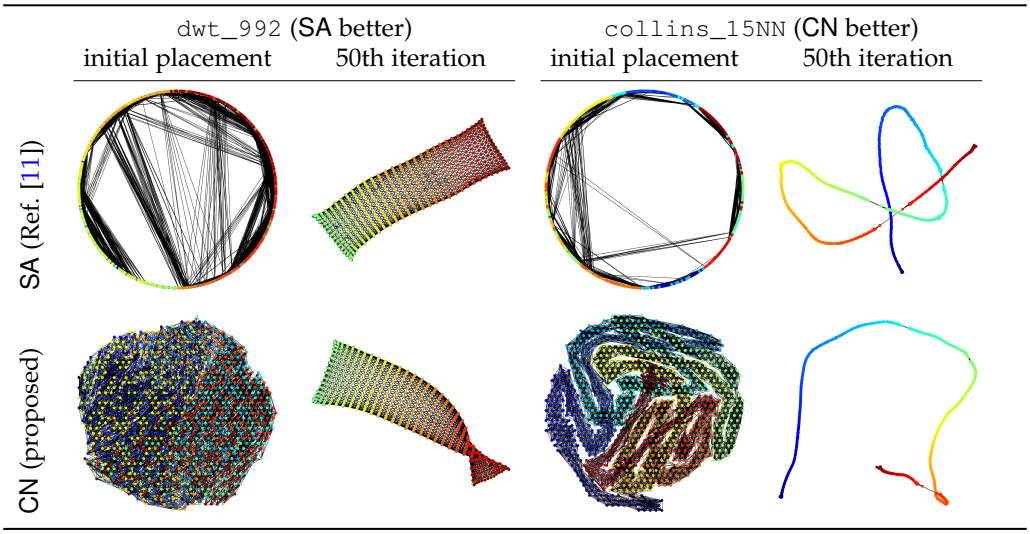


Fig. 11: Visualization results showing initial and 50th iteration placements for `dwt_992` and `collins_15NN`.

shows the difference between the two initialization. When we ignore the edge weights and just solve the problem (2), the graph is just an Erdos–Renyi graph, and thus SA cannot find any meaningful structure in the initial placement. On the other hand, the proposed algorithm can find the graph's structure, and we can observe that the left graph in Fig. 10 is separated by the groups, i.e., the node color. This result suggests that our proposed algorithm is effective even for weighted graphs, extending the applicability of the pre-processing step.

6 Challenges of Coordinate Newton Direction

This section explains why we took a roundabout approach to solve the problem (1). As we have explained, we first transformed it into the discrete optimization problem, the problem (3), then optimized it using the coordinate Newton direction. Instead, we can naturally consider directly applying the coordinate Newton direction to optimize the problem (1). Is it still effective? We think the answer is no, and in this section, we explain the reasons behind this conclusion and the rationale for our approach.

Fig. 12: The inaccurate quadratic approximation. For the red vertex on the right graph, its coordinate Newton direction is the red arrow, which is a bad direction.

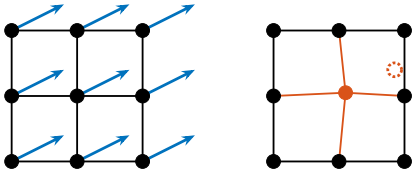


Fig. 13: The ignorance of other vertex movements. Although the blue arrows show the forces in this situation, the red vertex barely moves by the coordinate Newton direction.

6.1 Possible Approach with Coordinate Newton Direction

We first explain a possible approach using the coordinate Newton direction to optimize the problem (1). As mentioned, our approach is based on the stochastic coordinate descent and is also resembles the randomized subspace Newton, one of the subspace methods [2, 9, 12, 15, 25].

The direct application of such method to the problem (1) is as follows: we randomly select a vertex i, apply Newton's method or its regularized variant to f_i using the gradient in Eq. (7) and its Hessian:

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

Then, we update the position of vertex i and repeat this process until convergence. We do not go through any discrete optimization problem with this approach. However, this approach fails to work effectively in practice. In the following, we explain the reasons behind this failure.

6.2 Inaccuracy of Quadratic Approximation

One of the reasons why it fails is the inaccuracy of quadratic approximation; particularly, a specific issue arises when restricting the optimization to a coordinate block.

We show an example of this issue in Fig. 12. Let a graph G be the one on the left, where k and all positive edge weights $w_{i,j}$ are 1. In the situation depicted on the right, the position of points are

$$X = \begin{pmatrix} 0 & -1 & -0.85 & -0.85 & 1 \\ 0 & 0 & +0.155 & -0.155 & 0 \end{pmatrix}.$$

The key point of this example is that the Hessian

$$\nabla^2 f_2(x_2) \approx \begin{pmatrix} 1.841 & 0 \\ 0 & 1.159 \end{pmatrix}$$

is both positive definite and not ill-conditioned, which means that the Newton direction is good in general. Despite such a suitable property of the Hessian, the Newton direction for x_2 (red arrow) is a bad direction, leading to a significant deviation from the global optimal solution as it is. This badness comes from the inaccurate approximation of f in the restricted block coordinates x_2. We cannot completely resolve this illness by modifying Newton's method, which we use for f_i. This is an inherent and inevitable issue of the coordinate Newton direction.

6.3 Ignorance of Other Vertex Movements

Another reason is the ignorance of other vertex movements when optimizing each vertex individually. When optimizing for a vertex i, the coordinate Newton direction treats all other vertices $j (j \neq i)$ as fixed.

Figure 13 illustrates this issue. Consider a subset of vertices that form a mesh-like structure in G, and all vertices receive forces to the blue arrow directions. In this setting, the FR and L-BFGS algorithms progress the optimization without issue. On the other hand, if we attempt to optimize only for the red vertex 1 in the right graph, due to its fixed directly connected neighbors, 1 barely moves. As a result, the overall optimization barely advances, in contrast to the alignment of the forces (blue arrows).

In this way, ignoring the direction of forces on other vertices can be a significant shortcoming of the coordinate Newton direction.

6.4 Rationale for Proposed Method

As explained, we suspect that while the coordinate Newton direction effectively reduces the twist in a rough sense, it is unsuitable for optimizing the overall placement.

However, by converting the problem as stated in Section 4.1, we can mitigate these issues. The advantage of the problem (3) is that not only reduction of the computational complexity from $\mathcal{O}\left(|V|^2\right)$ to $\mathcal{O}(|E|)$. It also brings the convexity of the problem, making the quadratic approximation more accurate than the original problem. Moreover, the discrete point set Q mitigates the adverse effects caused by ignoring the movement of other vertices. This is because each point is always separated by at least ϵ, making minor movements negligible and non-critical. The stepsize in the optimization is also assured to be more than ϵ, evading the stagnate of the optimization. These advantages are the key to preventing the issues of Sec. 6.3 and Sec. 6.2. Thus, making the problem discrete is essential to provide a high-quality initial placement.

The crucial point is that, when combined with suitable strategies, the stochastic coordinate descent or randomized subspace Newton can effectively solve the original continuous problem, the problem (1). Focusing on each vertex separately is a natural and valid approach. With further refinements and by addressing the issues discussed in Sec. 6.2 and Sec. 6.3, these methods hold the potential to efficiently provide good solutions for the problem (1).

7 Discussion

In this section, we discuss the future directions of this research. Firstly, we discuss combining our algorithm with

conventional techniques, such as the multilevel approach. Secondly, we explore the applications beyond the scope of graph drawing. Finally, we conclude this paper.

7.1 Combination with Other Techniques

This paper has demonstrated the effectiveness of the proposed method on various graphs, but we can also apply it to larger-scale problems. In general, approximating or simplifying the model itself is one of the strategies to deal with large graphs. The n-body simulation using multipole expansions [13], approximating by the Barnes–Hut approximation [1], gradually refining the layouts using a multilevel approach [18], and employing stress majorization [10] are examples of such approaches. For instance, the Scalable Force-Directed Placement (sfdp) of Graphviz [7], based on [18], employs a multilevel approach to accelerate processing for larger graphs by progressively coarsening vertices.

In particular, the coarsening operation in sfdp does not critically conflict with the proposed method, making it feasible to combine both approaches. Specifically, by iteratively applying the proposed method to the entire coarsened graph or groups of vertices consolidated through coarsening, it is possible to extend its applicability to larger-scale problems. We can expect this approach to yield faster and higher-quality solutions. Addressing this integration is one of the challenges for future research.

In addition, the FR force model is sometimes used not only in $\mathbb{R}^2$ but also in $\mathbb{R}^3$ [22]. Although we have to modify some parts of the proposed algorithm for $\mathbb{R}^3$, such as the hexagonal lattice, its application would be easy.

7.2 Application to Other Problems

In this subsection, we briefly discuss and explore the potential applicability of stochastic coordinate descent to a broader range of problems. Although we utilized the coordinate Newton direction only for the optimization problem (1), we can see that its application is not necessarily limited to the FR force model alone.

In general, the optimization problem (1) is more broadly treated as "objective functions arising from graphs" [29]:

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad f(X) = \sum_{\{i,j\}\in E} f_{i,j}(x_i, x_j) + \lambda \sum_{i=1}^{n} \Omega_i(x_i),$$

where Ω_i is a regularization term for vertex i and λ is a regularization parameter. The optimization problem (1) is a special case of this problem class. The authors of [29] claim that coordinate descent, based only on coordinate gradients, effectively solves such problems. A variant of the proposed method utilizing the coordinate Newton direction can also be effective for such problems.

For instance, we suspect that the graph isomorphism problem is one of the candidates for the application. The graph isomorphism problem is a well-known combinatorial optimization problem to determine whether two graphs G_1, G_2 are isomorphic, i.e., $G_1 \cong G_2$. The graph isomorphism problem is closely related to the graph drawing. Drawing a graph in a way that reveals its symmetry is at least as difficult as the graph isomorphism problem [6]. Indeed, if we can draw two graphs G_1 and G_2 in the same way, it becomes evident that $G_1 \cong G_2$. See Fig. 14 for reference. When we relax it to a continuous optimization problem on Riemannian manifolds [20, 20], we might be able to apply the stochastic coordinate descent or coordinate Newton direction to this problem as well. Investigating the variant of our proposed algorithm for these problems constitutes one of the future research directions.

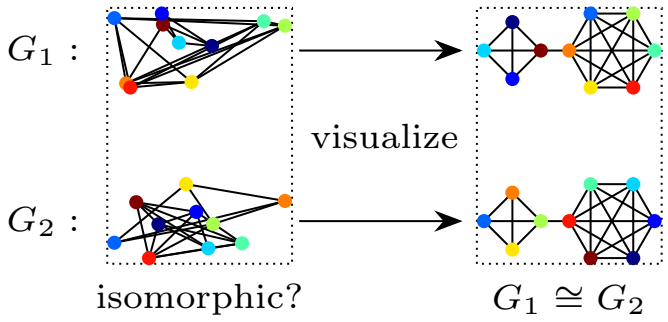


Fig. 14: Illustration of the relationship between graph drawing and graph isomorphism. If we can draw graphs G_1 and G_2 symmetrically, then it is clear that $G_1 \cong G_2$.

7.3 Conclusion

In this study, we proposed a new initial placement with the coordinate Newton direction for the FR force model. The obtained initial placements have fewer twists than the random initialization, leading to faster convergence and better outcomes. To demonstrate its effectiveness, we conducted numerical experiments, which revealed that the proposed method is effective across various graphs, extending the applicability of the pre-processing step.

We expect that the proposed method may advance the graph drawing of the FR force model and highlight the potential of the stochastic coordinate descent and its variants for addressing a broader range of graph-related optimization problems.

8 Acknowledgment

The author would like to express our sincere gratitude to PL Poirion and Andi Han for their insightful discussions, which have greatly inspired and influenced this research.

The author also thanks the developers of NetworkX and Graphviz. Their excellent work has been a great help in conducting this research.

This work was partially supported by JSPS KAKENHI (23H03351,24K23853) and JST ERATO (JPMJER1903).

References

[1] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," *Nature*, vol. 324, no. 6096, pp. 446–449, 1986. [Online]. Available: https://www.nature.com/articles/324446a0

[2] C. Cartis, J. Fowkes, and Z. Shao, "Randomised subspace methods for non-convex optimization, with applications to nonlinear least-squares," 2022. [Online]. Available: http://arxiv.org/abs/2211.09873

[3] 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)*, 2018, pp. 139–145. [Online]. Available: https://ieeexplore.ieee.org/document/8588785/?arnumber=8588785

[4] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006. [Online]. Available: https://igraph.org

[5] T. A. Davis and Y. Hu, "The University of Florida sparse matrix collection," *ACM Transactions on Mathematical Software (TOMS)*, vol. 38, no. 1, pp. 1–25, 2011.

[6] P. Eades, "A heuristic for graph drawing," *Congressus numerantium*, vol. 42, no. 11, pp. 149–160, 1984.

[7] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz— Open Source Graph Drawing Tools," in *Graph Drawing*, P. Mutzel, M. Jünger, and S. Leipert, Eds. Springer, 2002, pp. 483–484.

[8] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software: Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380211102

[9] T. Fuji, P.-L. Poirion, and A. Takeda, "Randomized subspace regularized Newton method for unconstrained non-convex optimization," 2022. [Online]. Available: http://arxiv.org/abs/2209.04170

[10] E. R. Gansner, Y. Koren, and S. North, "Graph Drawing by Stress Majorization," in *Graph Drawing*, D. Hutchison *et al.*, Eds. Springer Berlin Heidelberg, 2005, vol. 3383, pp. 239–250. [Online]. Available: http://link.springer.com/10.1007/978-3-540-31843-9_25

[11] 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*, ser. GECCO '16 Companion. Association for Computing Machinery, 2016, pp. 997–1000. [Online]. Available: https://dl.acm.org/doi/10.1145/2908961.2931660

[12] R. Gower, D. Kovalev, F. Lieder, and P. Richtarik, "RSN: Randomized subspace newton," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/bc6dc48b743dc5d013b1abaebd2faed2-Paper.pdf

[13] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, vol. 73, no. 2, pp. 325–348, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0021999187901409

[14] A. Hagberg, P. J. Swart, and D. A. Schult, "Exploring network structure, dynamics, and function using NetworkX," Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[15] R. Higuchi, P.-L. Poirion, and A. Takeda, "Fast Convergence to Second-Order Stationary Point through Random Subspace Optimization," 2024. [Online]. Available: http://arxiv.org/abs/2406.14337

[16] H. Hiroki, "Hari64boli64/FruchtermanReingoldByRandomSubspa [Online]. Available: https://github.com/hari64boli64/FruchtermanReingoldByRandomSubspace

[17] H. Hosobe, "Numerical optimization-based graph drawing revisited," in *2012 IEEE Pacific Visualization Symposium*, 2012, pp. 81–88.

[18] Y. Hu, "Efficient, high-quality force-directed graph drawing," *The Mathematica journal*, vol. 10, pp. 37–71, 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:14599587

[19] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, no. 1, pp. 7–15, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0020019089901026

[20] S. Klus and P. Gelß, "Continuous optimization methods for the graph isomorphism problem," 2023. [Online]. Available: http://arxiv.org/abs/2311.16912

[21] S. G. Kobourov, "Spring Embedders and Force Directed Graph Drawing Algorithms," 2012. [Online]. Available: http://arxiv.org/abs/1201.3011

[22] G. Kortemeyer, "Virtual-Reality graph visualization based on Fruchterman-Reingold using Unity and SteamVR," *Information Visualization*, vol. 21, no. 2, pp. 143–152, 2022. [Online]. Available: https://doi.org/10.1177/14738716211060306

[23] J. Li, Y. Tao, K. Yuan, R. Tang, Z. Hu, W. Yan, and S. Liu, "Fruchterman–reingold hexagon empowered node deployment in wireless sensor network application," *Sensors*, vol. 22, no. 5179, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/14/5179

[24] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1, pp. 503–528, 1989. [Online]. Available: https://doi.org/10.1007/BF01589116

[25] R. Nozawa, P.-L. Poirion, and A. Takeda, "Randomized subspace gradient method for constrained optimization," 2023. [Online]. Available: http://arxiv.org/abs/2307.03335

[26] N. Okazaki, "Chokkan/liblbfgs," 2024. [Online]. Available: https://github.com/chokkan/liblbfgs

[27] A. J. Patel, "Hexagonal Grids," Red Blob Games, Tech. Rep., 2013. [Online]. Available: https://www.redblobgames.com/grids/hexagons/

[28] Y. Qiu, "Yixuan/LBFGSpp," 2024. [Online]. Available: https://github.com/yixuan/LBFGSpp

[29] B. Recht and S. J. Wright, "Optimization for modern data analysis," 2019. [Online]. Available: https://people.eecs.berkeley.edu/~brecht/opt4ml_book/

[30] D. Tunkelang, "A numerical optimization approach to general graph drawing," Ph.D. dissertation, Carnegie Mellon University, 1999.

[31] T. L. Veldhuizen, "Dynamic Multilevel Graph Visualization," 2007. [Online]. Available: http://arxiv.org/abs/0712.1549

[32] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[33] J. X. Zheng, S. Pawar, and D. F. M. Goodman, "Graph

Drawing by Stochastic Gradient Descent," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 9, pp. 2738–2748, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8419285

APPENDIX A
ALGORITHM TO SOLVE THE PROBLEM 1

This section explains how to solve the problem (1) and obtain the final placement for the graph drawing.

A.1 Fruchterman–Reingold Algorithm

The Fruchterman–Reingold algorithm [8] is the original force-directed algorithm and the most standard approach for the FR force model.

Let $f_i \colon \mathbb{R}^2 \to \mathbb{R}$ denote the energy function for the vertex i at x_i. This is defined by

$$f_i(x_i) \coloneqq \sum_{j \neq i} E_{i,j}(\|x_i - x_j\|), \tag{6}$$

and its gradient, the sum of forces acting on the vertex i, is

$$\nabla f_i(x_i) = \sum_{j \neq i} \left(\frac{w_{i,j}\|x_i - x_j\|}{k} - \frac{k^2}{\|x_i - x_j\|^2} \right)(x_i - x_j). \tag{7}$$

The pseudo-code of the FR algorithm is shown in Algorithm 2, which can be regarded as a variant of gradient (steepest) descent method for the function f [30].

Algorithm 2: Fruchterman–Reingold algorithm

Input: Graph $G = (V, E)$, Weights $(w_{i,j})_{\{i,j\} \in E}$, Parameters $N_{\text{iter}}^{\text{FR}} \in \mathbb{N}$, $t_0 > 0$, Initial placement X
Output: Final placement $X = (x_1, \ldots, x_n)$

$t \leftarrow t_0$;
for $m \leftarrow 1$ **to** $N_{\text{iter}}^{\text{FR}}$ **do**
  compute gradient $\nabla f_i(x_i)$ for all $i \in V$;
  $x_i^{\text{new}} \leftarrow x_i - t \frac{\nabla f_i(x_i)}{\|\nabla f_i(x_i)\|}$ for all $i \in V$;
  $x_i \leftarrow x_i^{\text{new}}$;
  $t \leftarrow t - t_0 / N_{\text{iter}}^{\text{FR}}$;
  if *convergence condition is satisfied* **then**
    break;
return X;

The Algorithm 2 is based on the original code [8] and implementation in NetworkX [14] with some omitted details. The initial placement X is often drawn from a uniform distribution on a unit square. 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, though this placement is not necessarily the local optimum to the problem (1).

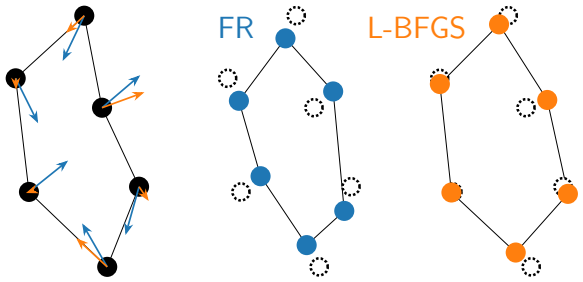


Fig. 15: Comparison of the FR algorithm and the L-BFGS algorithm. While the FR algorithm moves vertices in a descent direction with a fixed stepsize (blue arrows), the L-BFGS algorithm adjusts them differently since it utilizes approximated inverse Hessian (orange arrows).

A.2 L-BFGS Algorithm

Another approach to solving the optimization problem (1) is to use the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [17]. Using only a few recent gradient vectors, the L-BFGS algorithm approximates the inverse Hessian of the objective function f [24]. L-BFGS is known to be very efficient for large-scale optimization problems, and the superior performance of the L-BFGS algorithm to the FR algorithm reported in Ref. [17] also indicates this fact. Refer to Fig. 15 for a comparison to the FR algorithm.

For the optimization problem (1), we can apply the L-BFGS algorithm via flattening the matrix $X \in \mathbb{R}^{2 \times n}$ to a vector $\overline{X} \in \mathbb{R}^{2n}$. It is worth noting that this method ignores the structure of X and treats it just as a general optimization problem.

Hiroki Hamaguchi Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan.

Naoki Marumo Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan.

Akiko Takeda Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan. Center for Advanced Intelligence Project, RIKEN, Tokyo, Japan.