An 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. This layout can be interpreted as a solution of a continuous optimization problem, which can be solved using the FR algorithm, a gradient descent-like method, or the L-BFGS algorithm, quasi-Newton method. However, these methods are computationally expensive per iteration, which makes achieving high-quality visualizations for large-scale graphs challenging.
In this paper, to accelerate the optimization process, we propose a new initial placement with coordinate Newton direction. 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 algorithm or L-BFGS algorithm to obtain the final placement.
We demonstrate the effectiveness of our proposed approach through experiments, highlighting the potential of coordinate block based methods for graph drawing tasks. Additionally, we suggest combining our method with other graph drawing techniques for further improvement. We hope that this work will inspire future research of coordinate block based methods not only in graph drawing but also in 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 tasks in information visualization. Indeed, numerous kinds of models and algorithms have been proposed [1], [2], [3], and among these, one of the most popular choices is the force-directed graph drawing.

In force-directed graph drawing, we model a graph as a system of particles with forces acting between them. By simulating the system and seeking the equilibrium of the forces, we can obtain a visualization of the graph. Among the various force models [6], [5], the Fruchterman–Reingold (FR) force model [7], [8] is the central focus of our study.

FR algorithm is the original algorithm for this force model, and it can be regarded as a variant of gradient descent method for the energy function of the model. FR algorithm is implemented in many modern graph drawing libraries such as NetworkX [9], Graphviz [10], and igraph [11].

However, the FR algorithm has several issues that makes it challenging to achieve high-quality visualizations for large-scale graphs. First, it suffers from high computational complexity, $\mathcal{O}(n^2)$ per iteration as it is with n being the number of vertices. Secondly, the occurrence of "twist" [24] is crucial for the force model, as it can significantly impact the simulation efficiency. We refer to "twist" as unnecessary intersections of edges or tangled structures in visual representations, as shown in the upper half of Fig. 1. When "twist" exists, mutual interactions may weaken or diminish the forces, causing the simulation process to stagnate.

Approximating or simplifying the model itself is one of the strategies to address these issues in general. The n-body simulation using multipole expansions [12] or the Barnes–Hut approximation [13], gradually refining the layouts using a multilevel approach [14] and employing stress majorization [15] are examples of such approaches.

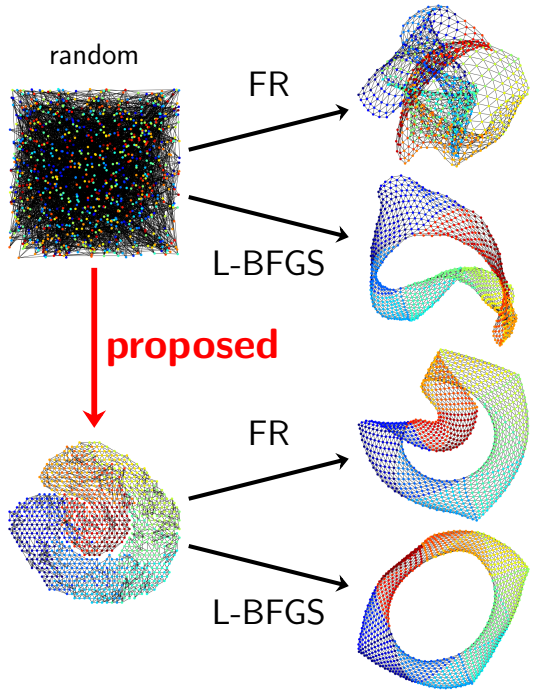


Fig. 1: Comparison of the algorithms for the `jagmesh1`.

Another strategy is to directly accelerate the simulation process with the forces, in other words, the optimization process of the energy function. This aligns with the aim of our work. Recent researches have accelerated the process through various ways, such as adapting to GPU parallel architectures [16], or utilizing numerical optimization meth-

ods. L-BFGS, a family of quasi-Newton methods, is one of the such methods, and is reported to be effective for graph drawing [17]. However, since this method just treats the problem as a general optimization problem, there is room for improvement. Simulated Annealing (SA) is also effective when used as a pre-processing step [4], since SA can deal "twist" issues and leads a better visualization combined with the FR algorithm. However, this work only uses a circle initial placement for unweighted graphs, and the effectiveness of large-scale graphs is unclear since it just repeats to try swapping the positions of two randomly selected vertices.

Based on such advances, in this paper, we propose a new initial placement for the FR force model as depicted in Fig. 1. Our goal is to accelerate the optimization process by leveraging the inherent structure of the problem, which has ignored in the L-BFGS algorithm. To achieve this, we optimize the position of vertices one by one with the Newton direction, which is defined in the coordinate block $\mathbb{R}^2$ in the entire variable space $\mathbb{R}^{2\times n}$. The result provides an initial placement with fewer "twists", accelerating the overall optimization process. Our work extends the applicability of initial placement idea to wider range of graphs. We also demonstrate its effectiveness through various experiments.

The rest of this paper is organized as follows. In Sec. 2, we define the optimization problem we consider and introduce the conventional approaches. In Sec. 3, we propose a new initial placement algorithm. In Sec. 4, we show the experimental results. In Sec. 5, we discuss the potential of the coordinate based methods, the key concept of our research. Finally, we discuss future work and conclude the paper in Sec. 6.

2 Preliminary

In this section, we formulate the simulation process of the force model as a continuous optimization problem and introduce the conventional approaches to this problem, namely the FR algorithm and the L-BFGS algorithm. We also briefly review the pre-processing step by Ref. [4].

2.1 Formulation of the Force Model

Let $\mathbb{R}_{>0} \coloneqq \{x \in \mathbb{R} \mid x > 0\}$, $\mathbb{R}_{\geq 0} \coloneqq \{x \in \mathbb{R} \mid x \geq 0\}$, and let $W = (w_{i,j}) \in \mathbb{R}_{\geq 0}^{n\times n}$ be an adjacency matrix of a graph $G_W = (V, E)$, where $V = \{v_i \mid 1 \leq i \leq n\}$ is a set of vertices and $E = \{\{v_i, v_j\} \mid w_{i,j} > 0\}$ is a set of edges. For simplicity, we sometimes identify each vertex v_i with its index i, and use notation such as $(i, j) \in E$. We call $w_{i,j}$ as a weight of the edge $\{v_i, v_j\}$.

We will only consider undirected connected graphs with non-negative weights. Although the FR algorithm in NetworkX, for example, can handle directed unconnected graphs with negative weights, this paper does not focus on such cases. For directed graphs, slight modifications of algorithms or converting graphs to undirected ones may be effective. For unconnected graphs, algorithms can be applied to each connected component independently. When negative weights are present, the optimization Prob. (3) defined below can be unbounded, but with non-negative weights and the connectivity of G, the problem is always bounded and solvable. In summary, the conditions for W is formulated as follows:

$$W \in \mathbb{R}_{\geq 0}^{n\times n}, \quad W = W^\top \quad \text{and } G_W \text{ is connected.} \tag{1}$$

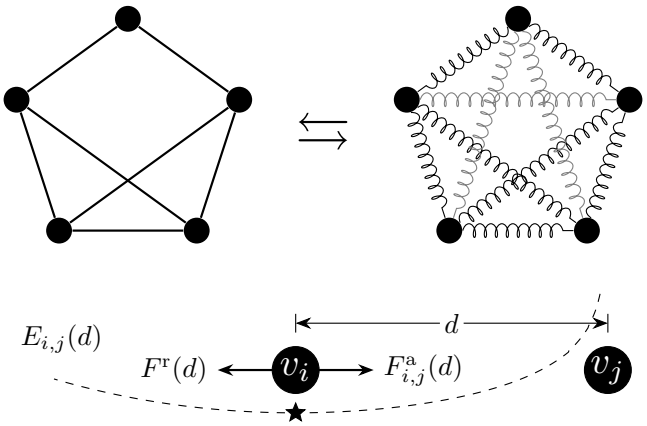


Fig. 2: (Top) The illustration of the force model. Forces acts on every pairs of vertices. (Bottom) Forces $F^{\mathrm{a}}_{i,j}(d)$ and $F^{\mathrm{r}}(d)$ work between v_i and v_j. The equilibrium of them is achieved at $d = k/\sqrt[3]{w_{i,j}}$, which equals k when $w_{i,j} = 1$.

Fruchterman and Reingold proposed a force-directed model for graph drawing, which is based on the physical analogy of the system of particles [7]. Let $x_i \in \mathbb{R}^2$ be the position of the vertex $v_i \in V$, and $X = (x_1, \ldots, x_n) \in \mathbb{R}^{2\times n}$ be the placement of the graph. Let $\|\cdot\|$ denote the Euclidean distance in $\mathbb{R}^2$. For a parameter k and a distance d between two vertices v_i and v_j, the attraction force $F^{\mathrm{a}}_{i,j} \colon \mathbb{R}_{>0} \to \mathbb{R}$ and the repulsion force $F^{\mathrm{r}} \colon \mathbb{R}_{>0} \to \mathbb{R}$ is defined as

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

The FR algorithm explained in Sec. 2.2 seeks the equilibrium of the forces between all pairs of vertices, as shown in Fig. 2.

We can also interrupt forces by its scalar potential [17], in other words, energy $E_{i,j} : \mathbb{R}_{>0} \to \mathbb{R}$, which is defined by

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

Although the energy function $E_{i,j}$ is convex and minimized when $d = k/\sqrt[3]{w_{i,j}}$, $E_{i,j}$ is not Lipschitz continuous since it diverges as $d \to 0$. Plus, a function $x_i \mapsto E_{i,j}(\|x_i - x_j\|)$ is not convex for a fixed x_j. Refer to Fig. 3.

Now, the optimization problem for the FR layout can be formulated as the minimization of the energy function $f : \mathbb{R}^{2\times n} \to \mathbb{R}$, as known as the stress of the graph G:

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad f(X) \coloneqq \sum_{i<j} E_{i,j}(\|x_i - x_j\|). \tag{3}$$

Seeking an equilibrium of the forces is equivalent to seeking a local minimum of the energy function $E_{i,j}$ for all pairs of vertices. In the following, we will discuss how to optimize this minimization Prob. (3).

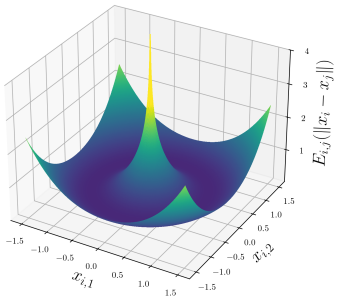


Fig. 3: Energy function $E_{i,j}(\|x_i - x_j\|)$ for $x_i = (x_{i,1}, x_{i,2})$, $x_j = (0,0)$, $w_{i,j} = 1$ and $k = 1$. Although $E_{i,j}$ is convex as a function of $d = \|x_i - x_j\|$, not convex as a function of x_i. As x_i approaches x_j, the energy function diverges.

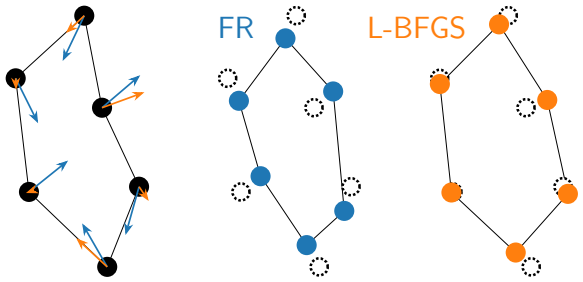


Fig. 4: Comparison of the FR algorithm and the L-BFGS algorithm. The FR algorithm moves vertices in a descent direction with a fixed step size (blue arrows), whereas the L-BFGS algorithm adjusts them differently since it utilizes approximated inverse Hessian (orange arrows).

2.2 Fruchterman–Reingold Algorithm

The Fruchterman–Reingold algorithm [7] is the original force-directed algorithm and most standard approach for this force model.

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

$$f_i(x_i) := \sum_{j \neq i} E_{i,j}(\|x_i - x_j\|), \tag{4}$$

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

$$\nabla f_i(x_i) = \sum_{j \neq i} \left(\frac{w_{i,j}\|x_i - x_j\|}{k} - \frac{k^2}{\|x_i - x_j\|^2} \right)(x_i - x_j), \tag{5}$$

The pseudo-code of the FR algorithm is shown in Algorithm 1, which can be regarded as a variant of gradient (steepest) descent method for the energy function f [19]. Alg. 1 is based on the original pseudo-code [7] and implementation in NetworkX [9] with some omitted details. The initial placement of the n points are sampled from a uniform distribution on a unit square in general. The parameter t denotes the temperature, which governs the step size along the steepest descent. As the temperature gradually decreases, the algorithm converges to a particular placement, though this placement is not necessarily the optimal solution.

Algorithm 1: Fruchterman–Reingold algorithm

Input: Graph $G_W = (V, E)$
Output: Point placement $X = (x_1, \ldots, x_n)$
Define parameters $N_{\text{iter}}^{\text{FR}}, k, t, \Delta_t := t/N_{\text{iter}}^{\text{FR}}$;
Define initial placement for all $v_i \in V$ randomly;
for $n_{\text{iter}} \leftarrow 1$ **to** $N_{\text{iter}}^{\text{FR}}$ **do**
  compute gradient $\nabla f_i(x_i)$ for all $v_i \in V$;
  $x_i \leftarrow x_i - t\frac{\nabla f_i(x_i)}{\|\nabla f_i(x_i)\|}$ for all $v_i \in V$;
  $t \leftarrow t - \Delta_t$;
  if *convergence condition is satisfied* **then**
    break;
return X;

2.3 L-BFGS Algorithm

Another approach to solve the optimization Prob. (3) is to use the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [17]. Using only a few recent gradient vectors, L-BFGS algorithm approximates the inverse Hessian of the objective function f, which is necessary to determine a descent direction [20]. We fix the maximum number of iterations to $N_{\text{iter}}^{\text{L-BFGS}}$. 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. 4 for a comparison to the FR algorithm.

For the optimization Prob. (3), we can apply the L-BFGS algorithm via flattening the matrix $X \in \mathbb{R}^{2\times n}$ to a vector $\overline{X} \in \mathbb{R}^{2n}$. However, it is worth noting that this method ignores the structure of X and treats it just as a general optimization problem. Thus, we can expect room for improvement by leveraging what we have ignored in this L-BFGS algorithm.

2.4 Pre-Processing by Simulated Annealing

As a related work, we briefly introduce and discuss the pre-processing step proposed by Ref. [4]. Let $Q^{\text{circle}} := \{(\cos(2\pi i/n), \sin(2\pi i/n)) \mid 1 \leq i \leq n\}$ be the points on a unit circle in $\mathbb{R}^2$. For an unweighted graph G_W, let $E_2 := \{\{v_i, v_j\} \mid \exists v_k \in V \text{ s.t. } \{v_i, v_k\} \in E \wedge \{v_k, v_j\} \in E\} \setminus E$ be a set of vertex pairs with a theoretical distance equal to 2. This study defines the problem for pre-processing as

$$\begin{array}{ll} \underset{\pi:\, V \to Q^{\text{circle}}}{\text{minimize}} & \displaystyle\sum_{(i,j)\in E\cup E_2} \|\pi(v_i) - \pi(v_j)\|^2, \\ \text{subject to} & \pi(v_i) \neq \pi(v_j), \quad \forall (i,j)\ (i<j). \end{array} \tag{6}$$

Prob. (6) is a discrete optimization problem to find the best assignment π from vertices V to the points on the circle Q^{circle} just using Euclidean distances, not the energy functions. By setting the result of Simulated Annealing (SA) for Prob. (6) as the initial placement for the FR algorithm, we can obtain a faster and better visualization.

However, the following limitations remain:

1) Restricted to unweighted graphs.
2) Confined to a simple circle layout, which could be ineffective for complex structure graphs.
3) The only neighborhood in the SA is the random swapping of two vertices, making the optimization process inefficient for large-scale graphs.
4) $|E_2|$ could be $\Theta(n^2)$, unable to leverage the sparsity of graphs if it exists.

Although our algorithm and motivation is different from this study's, we are dealing with these limitations and can regard our study as an extension of this prior work.

3 Proposed Algorithm

In this section, we propose a new initial placement algorithm for the FR layout. We first introduce the graph drawing by Stochastic Gradient Descent (SGD) [18] and the concept of the coordinate Newton direction, the key concepts of our research. Then, we define the discrete optimization problem for initial placement and propose a new algorithm to optimize it.

3.1 Related Work

Firstly, we introduce two important previous works. Although these works are not completely same as our work, explaining them is quite helpful to understand the motivation of our work.

3.1.1 Graph Drawing by Stochastic Gradient Descent

When we regard graph drawing as an optimization problem, Stochastic Gradient Descent (SGD) for Kamada–Kawai (KK) layout [5] is one of the most notable works. In KK layout, we regard G as a complete graph and assign the energy function $E_{i,j}$ to all edges. SGD in this context means to repeat randomly selecting a edge $\{v_i, v_j\}$ and updating x_i and x_j with the gradient of $E_{i,j}$. This algorithm is known to be effective various optimization problems in general, and its effectiveness for graph drawing is shown in Ref. [18].

Although applying SGD to our problem Prob. (3) is straightforward, it is not effective for this problem. This is because the force model we consider assigns exactly 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 v_i and v_j, no matter how close they are 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 Prob. (3).

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

3.1.2 Newton Direction and Subspace Newton Direction

We also introduce the Newton direction. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. The second order approximation at x_0 is

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

Since f is convex, the Hessian matrix $\nabla^2 f(x_0)$ is positive semi-definite. The argmin x^* of this approximation satisfies

$$\begin{aligned} &\nabla f(x_0) + \nabla^2 f(x_0)(x^* - x_0) = 0 \\ \iff &x^* = x_0 - \nabla^2 f(x_0)^{-1} \nabla f(x_0). \end{aligned}$$

We call the direction $d = -\nabla^2 f(x_0)^{-1} \nabla f(x_0)$ as the Newton direction. Although Newton direction is the optimal direction for the approximated f, 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. Actually, the reason why L-BFGS algorithm approximates the inverse Hessian is to avoid this computational cost.

Still, we can leverage the concept of the Newton direction in a different way, 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 computation of the coordinate Newton direction is much cheaper than that of the Newton direction, we can repeat this procedure many times. In general, this idea is also known as Randomized Subspace Newton (RSN) [25] in a broader context.

In particular, this coordinate Newton direction has obvious natural affinity to the Prob. (3) in Sec. 2.4. By taking x_i, the position of the vertex v_i, as the coordinate block, we can compute the Newton direction of f_i in Eq. (4). Although directly applying this idea to Prob. (3) is challenging as we will discuss in Sec. 5, we leverage this coordinate Newton direction to propose our algorithm.

3.2 Reduction To The Discrete Optimization Problem

Now, to define a problem for our initial placement, we transform the optimization problem (3) into a constrained discrete optimization problem. As written in Sec. 3.1.1, the energy function $E_{i,j}$ is $-k^2 \log d$ for all $(i,j) \notin E$. Considering the sparsity of many practical graphs ($|E| \ll |V|^2$), simplifying as follow is reasonable:

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad \sum_{(i,j)\in E} \frac{w_{i,j}\|x_i - x_j\|^3}{3k} - \sum_{i<j} k^2 \log \|x_i - x_j\|. \tag{7}$$

Further, by converting the second term into a constraint, the problem (7) can be approximated so that the objective function can be computed with a complexity dependent on $|E|$ rather than $|V|^2$:

$$\begin{aligned} \underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad & f^{\mathrm{a}}(X) \coloneqq \sum_{(i,j)\in E} \frac{w_{i,j}\|x_i - x_j\|^3}{3k} \\ \text{subject to} \quad & \|x_i - x_j\| \geq \epsilon, \quad \forall (i,j)\ (i<j) \end{aligned} \tag{8}$$

where ϵ is a suitably chosen positive constant. This conversion does not lose the essence of the problem too much because $E^{\mathrm{r}}(d) = -k^2 \log d$ is a convex function such that it decreases monotonically concerning d. Thus, for sufficiently large d, the value of $-k^2 \log d$ does not grow excessively, and for too small d, we can prevent the divergence of the energy function by setting ϵ.

However, problem (8) still involves $\mathcal{O}\left(|V|^2\right)$ constraints, which negates the advantage of computing the objective function with $\mathcal{O}(|E|)$ complexity. To further simplify, we incorporate the concept of initial placements as mentioned in Sec. 2.4. By simplifying problem (8) with a fixed initial placement Q whose points are separated by at least ϵ, we obtain the following discrete optimization problem:

$$\begin{aligned} \underset{\pi : V \to Q}{\text{minimize}} \quad & \sum_{(i,j)\in E} \frac{w_{i,j}\|\pi(v_i) - \pi(v_j)\|^3}{3k}, \\ \text{subject to} \quad & \pi(v_i) \neq \pi(v_j), \quad \forall (i,j)\ (i<j). \end{aligned} \tag{9}$$

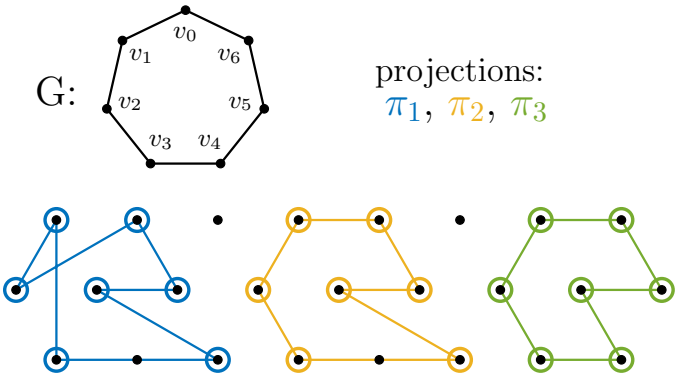


Fig. 5: Concept of Q and π. The injection π maps vertices V to a discrete point placement Q, especially a hexagonal lattice Q^{hex}. Apparently, among π_1, π_2, π_3, the right one π_3 is the best mapping for the Prob. (9).

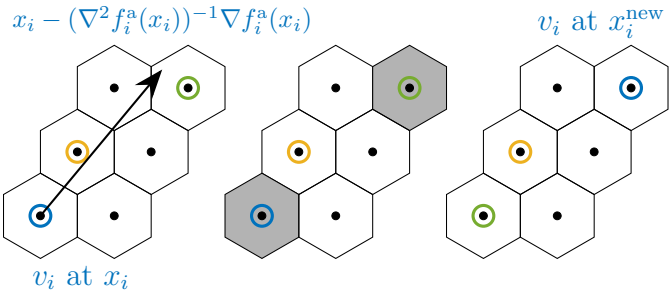


Fig. 6: Visual explanation of the one iteration of the proposed algorithm. Step1. Compute the coordinate Newton direction for a randomly selected vertex (blue). Step2. Decide x_i^{new} by rounding the direction and adding a random vector. Step3. Swap the vertices if there is a collision. In this case, swap blue and green vertices.

It means that with a discrete set of points Q such that the points are separated by at least ϵ, we seek the best injection $\pi : V \to Q$ that minimizes the objective function f^{a}. By fixing the possible point placement in advance, we can skip the check of the $\mathcal{O}\left(|V|^2\right)$ constraints, reducing the computational complexity to $\mathcal{O}(|E|)$ and thus offering significant speedup. See Fig. 5 as a visual explanation.

As an instance of such a discrete point placement Q, we can consider various types of placements, including Q^{circle} [4]. However, in this study, we adopt a hexagonal lattice Q^{hex} [27], [26]. When minimizing the attraction energy 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 for our purpose.

3.3 Newton Direction for Discrete Optimization

Next, using the coordinate Newton direction of a randomly selected vertex, as mentioned in Sec. 3.1, we optimize the discrete optimization problem.

Let the objective function $f_i^{\mathrm{a}}(x_i)$ corresponding to a vertex v_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

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

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

Since f_i^{a} is convex, the Hessian matrix $\nabla^2 f_i^{\mathrm{a}}(x_i)$ is positive semi-definite. This is a large difference from the functions $f_i(x_i)$ in Eq. (4) and $f^{\mathrm{a}}(X)$ in Prob. (8), which are not convex.

The ordinary updated rule with the coordinate Newton direction is

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

However, x_i^{new} may not be in the hexagonal lattice Q^{hex}, which is a constraint of Prob. (9). Thus, we need to round to the nearest point in Q^{hex}. Plus, we empirically found that adding a random vector to the Newton direction is effective for the optimization process, which is a similar strategy to the SA in Sec. 2.4. 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 v_i is

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

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.

If there is a vertex v_j such that $\pi(v_j) = x_i^{\mathrm{new}}$, we swap the vertices v_i and v_j in π to keep the injective property of π. Otherwise, we just update $\pi(v_i)$ to x_i^{new}. Refer to Fig. 6 for a visual explanation.

3.4 Optimal Scaling

When we optimize a placement for FR-layout with an initial placement obtained, scaling the initial placement at first can often yield better results than directly using the unmodified initial placement. In this subsection, we explain how to find the optimal scaling factor that minimizes the energy function for a given placement.

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

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

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

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

The function $\phi(c)$ is convex, and we can find the optimal scaling factor c^* by solving $\phi'(c^*) = 0$, which yields

$$c^* = \left(\frac{k^2 n(n-1)}{3\sum_{(i,j)\in E} \frac{w_{i,j}\|x_i - x_j\|^3}{k}} \right)^{1/3}. \tag{10}$$

This value can be computed in $\mathcal{O}(|E|)$ complexity. Using this scaling factor, we can obtain a further better initial placement for the FR layout.

It is notable that the optimal solution to Prob. (9) is invariant under scaling, thus we can select any ϵ to define the hexagonal lattice Q^{hex} as far as we scale the placement by c^* as post-processing.

3.5 Pseudo Code

We presented the overall framework of the proposed method in Algorithm 2. Be aware that the proposed algorithm is not a complete solution to Prob. (3), this only provides an initial placement for the algorithms such as the FR algorithm or the L-BFGS algorithm.

Algorithm 2: Proposed algorithm as initial placement for the FR layout

Input: Graph $G_W = (V, E)$, subspace dimension s
Output: Point placement $X = (x_1, \ldots, x_n)$
Define parameters $N_{\text{iter}}^{\mathsf{CN}}, k, t, \Delta t$ and hexagonal lattice Q^{hex};
Set π as a injection from V to Q^{hex} randomly;
for $j \leftarrow 0$ **to** $N_{\text{iter}}^{\mathsf{CN}}$ **do**
  $v_i \leftarrow$ randomly selected vertex from V;
  $x_i \leftarrow \pi(v_i)$;
  $x_i^{\text{new}} \leftarrow \text{round}(x_i - \nabla^2 f_i(x_i)^{-1}\nabla f_i(x_i) + t \cdot \text{rand})$;
  if $\exists v_j$ *s.t.* $\pi(v_j) = x_i^{\text{new}}$ **then**
    Swap v_i and v_j in π;
  else
    $\pi(v_i) \leftarrow x_i^{\text{new}}$;
$x_i \leftarrow \pi(v_i)$ for all $v_i \in V$;
$c^* \leftarrow$ optimal scaling factor by Eq. (10);
$x_i \leftarrow c^* x_i$ for all $v_i \in V$;
return X

3.6 Alternative Approach

As the end of this section, we explain an alternative approach of this algorithm. We can also consider to update all vertices simultaneously, not one by one. It means that moving all the points $\{x_i\}_{1\le i\le |V|} \subseteq Q^{\text{hex}}$ on the hexagonal grid to arbitrary points $\{\hat{x}_i := x_i - \nabla^2 f_i(x_i)^{-1}\nabla f_i(x_i)\}_{1\le i\le |V|} \subseteq \mathbb{R}^2$, and then assign all these $|V|$ points to the nearest points in Q^{hex}. If we define the assignment problem as the minimization of the sum of the squared distances between $\{x_i\}_{1\le i\le |V|}$ and $\{\hat{x}_i\}_{1\le i\le |V|}$, we can solve this problem by minimum-cost flow or Hungarian algorithm with $\mathcal{O}\left(|V|^3\right)$ complexity. Additionally, we can obtain a heuristic solution in $\mathcal{O}(|V|\log|V|)$ by appropriately sorting $\{\hat{x}_i^{\text{new}}\}_{1\le i\le |V|}$.

While we are not expecting as good performances as the proposed algorithm, they offer certain advantages, such as simplified implementation or avoiding random access to arrays.

4 Numerical Experiment

In this section, we evaluate the proposed algorithm by various numerical experiments based on the setup described in Sec. 4.1. We conducted two types of experiments, based on Ref. [18]: exhaustive experiments to evaluate the performance of the proposed algorithm in various situations in Sec. 4.2, and detailed experiments to investigate the behavior of the proposed algorithm in detail in Sec. 4.3.

4.1 Experimental Setup

All numerical experiments in this section were conducted 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.

To implement FR algorithm and L-BFGS algorithm, we referenced NetworkX version 3.3 [9], SciPy 1.14.1 [21], and C++ L-BFGS [22], [23]. 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]. As a side note, we also used Graphviz version 2.43.0 [10] to draw Fig. **??**.

We used the 4 algorithms: FR algorithm (FR), the proposed initialization plus FR algorithm (CN-FR), L-BFGS algorithm (L-BFGS), and the proposed initialization plus L-BFGS algorithm (CN-L-BFGS). CN represents the Subspace Newton, and we refer CN-FR and CN-L-BFGS as CN, and FR and L-BFGS as non-CN.

As a parameter, we used $N_{\text{iter}}^{\mathsf{CN}} = 3\frac{|V|^3}{|E|}$ for the FR algorithm. Since the amortized time complexity per iteration of Algorithm 2 is $\mathcal{O}\left(\frac{|E|}{|V|}\right)$, we can roughly expect that the computational time of the proposed algorithm is equivalent to 3 iterations of the FR algorithm.

All the codes are available at our GitHub [28].

4.2 Exhaustive Experiment

To begin with, we conducted an exhaustive experiment to evaluate the performance of the proposed algorithm with various graphs.

As a dataset, we used matrices from Sparse Matrix Collection [29] satisfying the condition (1) with $|V| \le 1000$, in total 124 graphs.

The result is shown in Fig. 12. Almost all of the cases, the proposed algorithm performed better than random initialization.

There are a few cases where the proposed algorithm performed worse than random initialization, and we visualized such cases in Fig. **??**, Fig. **??**, Fig. **??**, and Fig. **??**. We can observe that the reasons why it was worse though these figures. todo

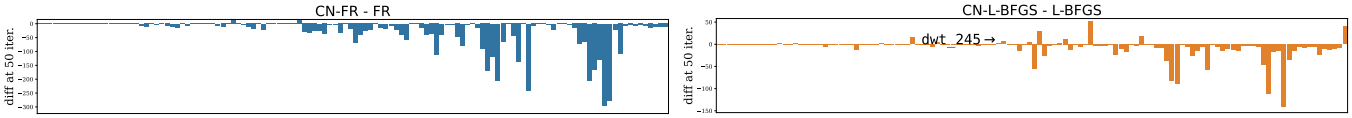


Fig. 7: Comparison of the proposed initialization with random initialization. For non-CN algorithms, $N_{\text{iter}}^{\text{FR}}$ and $N_{\text{iter}}^{\text{L-BFGS}}$ as 50, the default parameter of NetworkX [9]. For CN algorithms, we set them as 45, since it contains pre-processing step. Almost all of the cases, the difference are negative, meaning that the proposed algorithm performed faster and better than random initialization.

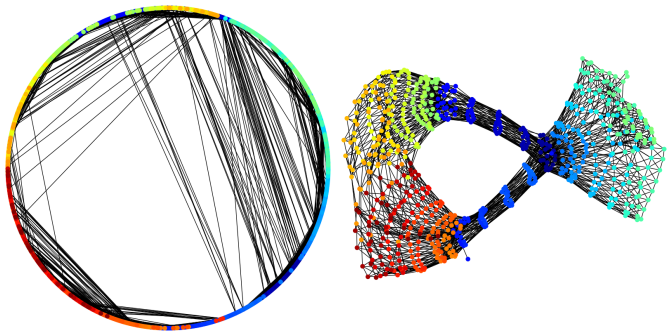


Fig. 8: `can_715` with CI-L-BFGS

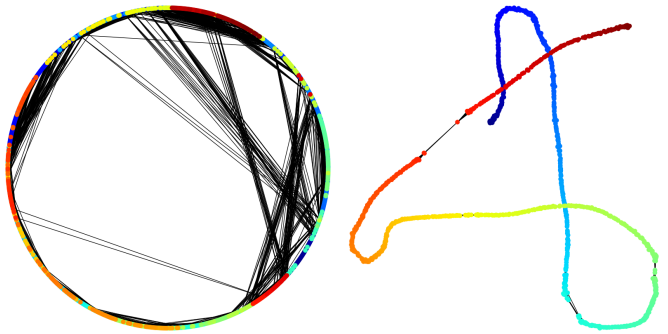


Fig. 10: `collins_15NN` with CI-L-BFGS

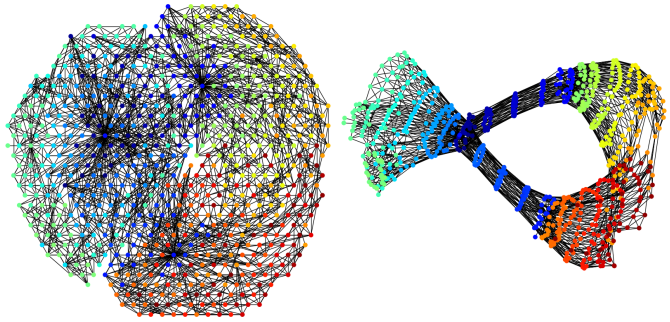


Fig. 9: `can_715` with CN-L-BFGS

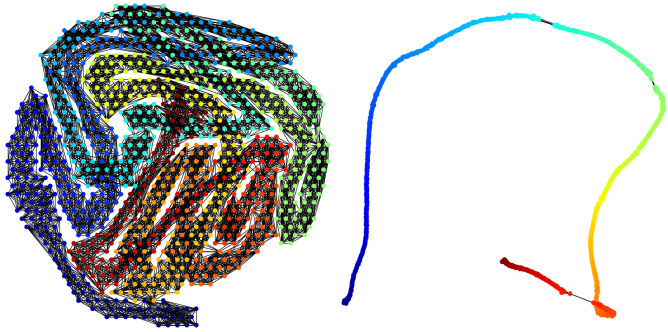


Fig. 11: `collins_15NN` with CN-L-BFGS

4.3 detailed Experiment

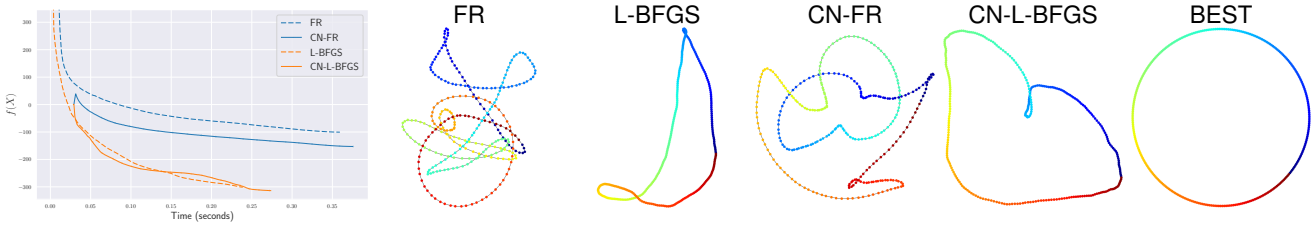
We also conducted a detailed experiment to investigate the behavior of the proposed algorithm in detail. The result is shown in Fig. 12.

The experiment details are as follows. As parameters, we used $N_{\text{iter}}^{\text{FR}} = N_{\text{iter}}^{\text{L-BFGS}} = 200$ as the maximum number of iterations. 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 [29], and these choices are based on the experiments conducted in Ref. [18]. 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.

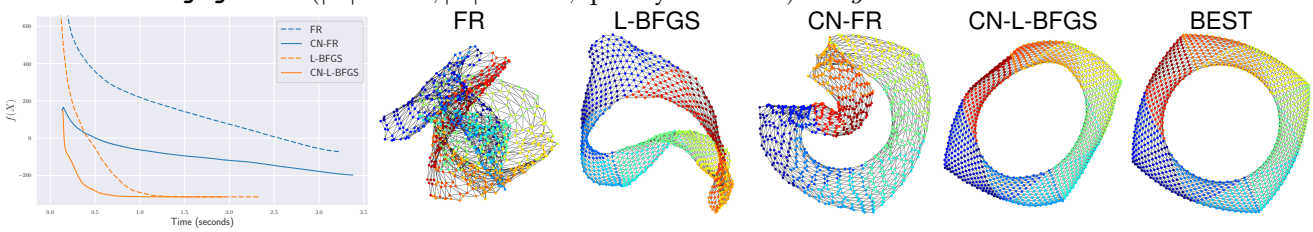
We first explain what Fig. 12 represents. The plots on the left illustrate the objective function values $f(X)$ on the vertical axis versus execution time on the horizontal axis, using 10 trials for each algorithm. Faint crosses represent the results of non-CN algorithms, while faint circles represent the results of CN algorithms, plotted every 10 iterations for each trial. The solid and dashed lines represent the average of these values across the 10 trials for each algorithm. If one of the trials terminated before reaching $N_{\text{iter}}^{\text{FR}}$ or $N_{\text{iter}}^{\text{L-BFGS}}$ iterations, the line was plotted up to the minimum trial count achieved across all trials. Each trial was conducted with a different seed, meaning that even deterministic algorithms, such as FR or L-BFGS, converged to different local optima due to variations in the initial random placement. The graphs on the right illustrates the placement at the 150th iteration (or the final iteration if it concluded before then) for each algorithm with the seed 0.

The observations and implications of Fig. 12 are as follows. First, the solid plots for CN generally demonstrates superior performance compared to non-CN, validating the efficacy of the proposed method. Some exception of the superiority arise with FR, which exhibits oscillations in the plot, likely due to excessive step sizes leading to overshooting. Although we refrained from altering FR for fairness, adjusting the step size could enable the proposed method to achieve its intended performance. In fact, the initial $f(X)$ of CN-FR is clearly small enough compared to the objective function values produced by FR alone, suggesting that the

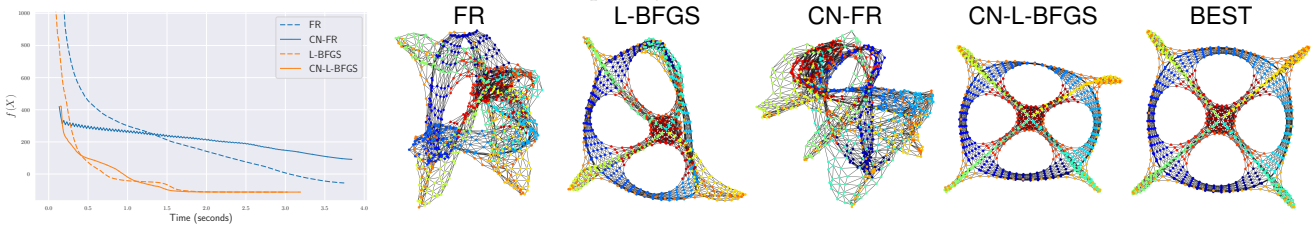
cycle300 ($|V| = 300, |E| = 300, \text{sparsity} = 0.669\%$) $Figures are at 150 iterations.$



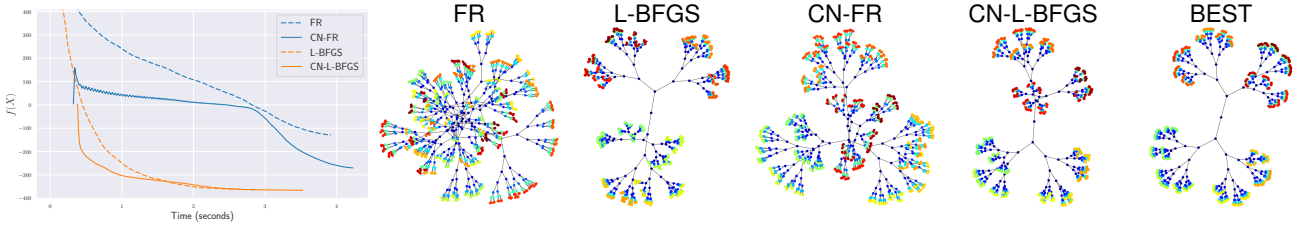
jagmesh1 ($|V| = 936, |E| = 2664, \text{sparsity} = 0.609\%$) $Figures are at 50 iterations.$



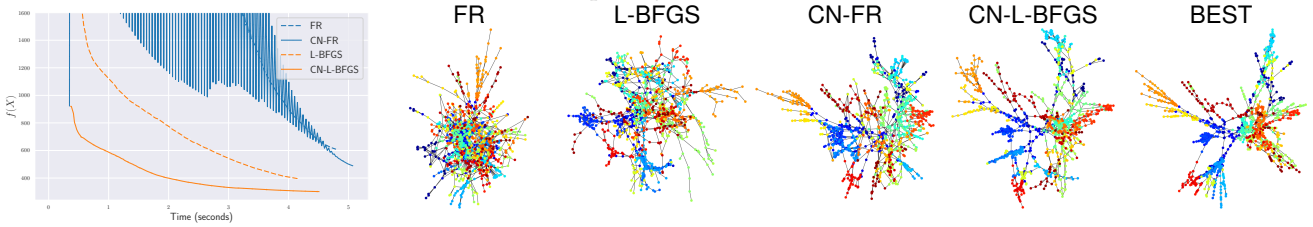
dwt_1005 ($|V| = 1005, |E| = 3808, \text{sparsity} = 0.755\%$) $Figures are at 100 iterations.$



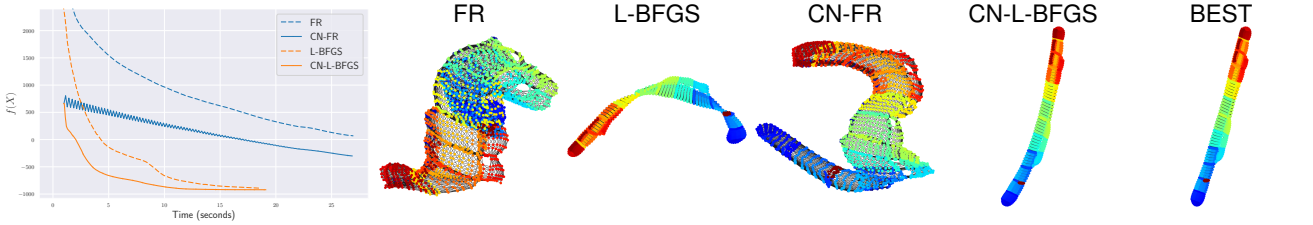
btree9 ($|V| = 1023, |E| = 1022, \text{sparsity} = 0.196\%$) $Figures are at 150 iterations.$



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

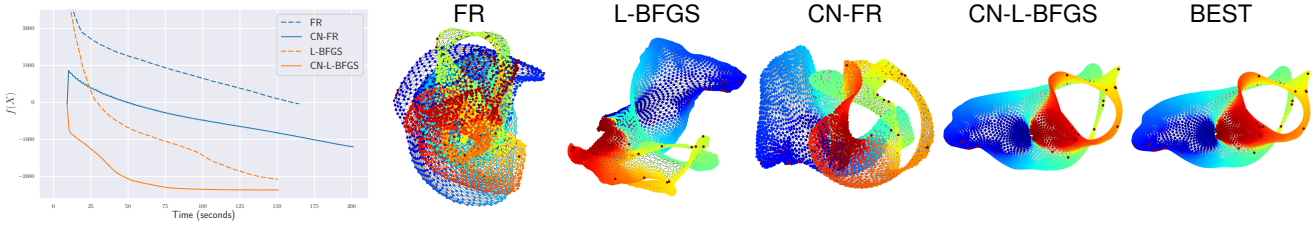


Fig. 12: Numerical experiment results for various graphs. Please refer Sec. 4.3 for details.

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 compared to non-CN placements.

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

5 Challenges of the Coordinate Newton Direction

Here, we explain the rationale for why we took a roundabout approach to optimize Prob. (3). As we have explained, we first transformed it into a discrete optimization problem, Prob. (9), and then optimized it using the coordinate Newton direction. However, is it possible to directly leverage the coordinate Newton direction to optimize Prob. (3)? We think the answer is no, and in this section, we explain the reasons behind this conclusion.

The important point is that these issues do not necessarily preclude the possibility of achieving global optimization with the coordinate Newton direction. Rather, we hope that an accurate understanding of these challenges will lead us to develop better optimization methods in the future.

5.1 Possible Approach with Coordinate Newton Direction

In this subsection, we explain a possible approach to optimize Prob. (3) using the coordinate Newton direction. As mentioned, our approach resembles to the RSN method, one of the subspace methods, and RSN and its variant have been proposed in the context of continuous optimization [25], [30], [31], [32], [33].

We take the following algorithm as a direct application of the subspace methods to the FR layout; Namely, we randomly select a vertex v_i, apply Newton's method or its regularized variant to f_i using the gradient in Eq. (5) and its Hessian:

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

Then, we update the position of vertex v_i, and repeat this process until convergence. However, this approach fails to work effectively in practice. In the following, we explain the reasons behind this difficulty.

5.2 Inaccuracy of quadratic approximation

This subsection discusses the inaccuracy of quadratic approximation used in the Newton's method, particularly a specific issue arises when restricting the optimization to a subspace.

Fig. 13: The inaccurate quadratic approximation. Assume that the optimal placement of the graph is as shown on the left. For the red vertex on the right graph, its Newton direction is the red arrow, which is apparently a bad direction.

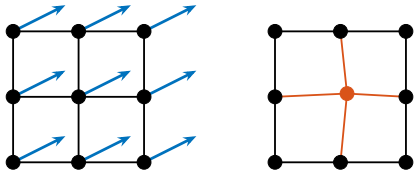


Fig. 14: The ignorance of other vertex movements. Assume that the blue arrows show the forces to the vertices and the optimal movements in this situation. Nevertheless, the red vertex will only move a little by the subspace method.

When we apply the Newton's method to a randomly selected vertex v_i, we approximate the energy function f_i as a quadratic function $\overline{f}_i$ at x_i. If this approximated function is convex, then the Newton's method update v_i to the optimal solution of the approximated function. Otherwise, by adding regularization terms, such as the cubic regularization [34], we can update v_i to a descent direction of $\overline{f}_i$.

Nevertheless, the update of x_i, which locally improves $\overline{f}_i$, does not necessarily improves the overall energy function f.

We show an example of this issue in Fig. 13. Let a graph G be as shown in the left (optimal), where k and all positive edge weights $w_{i,j}$ are set to 1. In a successful case such as in the middle (success), the subspace method works effectively, since the Newton direction defined on f_i (blue arrow) leads to the improvement of the overall energy function f.

However, in the situation depicted on the right (failure), the Newton direction for x_1 (red arrow) is apparently a bad direction, leading to a significant deviation from the global optimal solution as it is. The points of G are set as

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

and the Hessian $\nabla^2 f_1(x_1)$ is approximately

$$\begin{pmatrix} 1.841 & 0 \\ 0 & 1.159 \end{pmatrix},$$

which is positive definite and not ill-conditioned. Despite of such a good property of the Hessian, the Newton direction for x_1 (red arrow) is a bad direction, since the repulsive force between x_1 and x_2, x_3 are too strong to diverge from the global optimal solution. This kind of illness cannot be resolved by just modifying the Newton's method we use for f_i, and it is a inherent problem of the subspace method.

5.3 Ignorance of other vertex movements

This subsection discusses another issue of the subspace method, which is the ignorance of other vertex movements

when optimizing each vertex individually. When optimizing for a vertex v_i, the subspace method treat all other vertices $v_j (j \neq i)$ as fixed. However, both the FR algorithm and the L-BFGS algorithm do not, which represents a significant difference.

This is illustrated in Figure 14. Consider a subset of vertices in a mesh-like structure G, all vertices receive forces to the blue arrow directions and can minimize f by moving in that direction. In this setting, both the FR algorithm and the L-BFGS algorithm progress the optimization without issue. On the other hand, if we attempt to optimize only for the red vertex in the right graph, heavily influenced by its directly connected neighbors, v_0 barely moves. As a result, the overall optimization barely advances, in contrast to the alignment of the blue arrows. The same problem occurs for other vertices.

In this way, ignoring the direction of forces on other vertices, or more precisely, neglecting the values of $\frac{\partial^2 f}{\partial x_i \partial x_j}$, is a major shortcoming of the subspace method.

5.4 Rationale for the Proposed Method

As explained, we suspect that, while the subspace method is effective for reducing the "twist" in a rough sense, it is not suitable for optimizing the overall placement.

However, by converting the problem as stated in Section 3.2, we can take this ignored interaction into account to some extent. The advantage of Prob. (9) is that not only reduction of the computational complexity from $\mathcal{O}\left(|V|^2\right)$ to $\mathcal{O}(|E|)$, but also allowing us to more easily take account for vertex interactions and prevent them from becoming too close to each other, since the mapping π is injection. Thus, making the problem discrete brings the acceleration of each iteration as well as the high quality of the updated solution.

The important point is that, combined with some appropriate ideas, the subspace method can be very effective for the FR layout. Focusing on optimizing each vertex individually is a natural and valid approach, and with further refinements, the subspace method holds the potential to efficiently yield global optimal solutions.

6 Discussion

In this section, we discuss the future directions of this research. Firstly, the proposed method would be better to combine with the some conventional techniques such as Multilevel approach. We explain this in Sec. 6.1. Secondly, we explore the applications beyond the scope of graph drawing in Sec. 6.2. Finally, we conclude this paper in Sec. 6.3.

6.1 Combination with Other Techniques

This paper has demonstrated the effectiveness of the proposed method on relatively small-scale graphs, but it is believed that it can also be applied to more larger-scale problems. For instance, the Scalable Force-Directed Placement (sfdp) of Graphviz [10], based on [14], employs a multilevel approach to accelerate processing for larger graphs by progressively coarsening vertices. The result of sfdp shown in Fig. **??** validate the utility of this technique.

The coarsening operation does not 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 to groups of vertices consolidated through coarsening, it is possible to extend its applicability to larger-scale problems. This approach is expected to yield faster and higher-quality solutions. Addressing this integration is one of the challenges for future research.

In addition, the FR layout sometimes used not only in 2d but also in 3d [35]. Although we have to modify some parts of the proposed algorithm to apply to 3d FR layout, such as the hexagonal lattice, its application would be not so difficult.

6.2 Application to Other Problems

In this subsection, we briefly discuss and explore the potential applicability of the subspace method to a wider range of problems. Although we employed the idea of subspace methods only for the optimization Prob. (3), we can see that its application is not necessarily limited to the FR layout alone.

In general, the optimization Prob. (3) is more broadly treated as "objective functions arising from graphs" [36]:

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

where Ω_i is a regularization term for the i-th vertex, and λ is a regularization parameter. The FR layout Prob. (3) is obviously a special case of this problem class.

We expect that a variant of subspace method to be effective for such problems, as shown in this study. The authors of [36] claim that coordinate descent is an effective method for solving such problems. Given the similarity between the subspace method and coordinate descent as depicted in Sec. **??**, this claim supports the potential of the subspace method for a wider range of problems.

For instance, we suspect that the graph isomorphism problem is one of the candidates for the application of the subspace method. The graph isomorphism problem is a well-known combinatorial optimization problem to determine whether two graphs G_1, G_2 are isomorphic, i.e., $G_1 \cong G_2$. In fact, 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 two graphs G_1 and G_2 can be drawn in the same way, it becomes evident that $G_1 \cong G_2$. See Fig. 15 for reference. When we relax it to a continuous optimization problem on Riemannian manifolds as in [37], [37], we might be able to apply the subspace method to this problem as well. Investigating the applicability of the subspace method to these problems constitutes one of the future research directions.

6.3 Conclusion

In this study, we proposed a new initial placement with the subspace Newton direction. To demonstrate its effectiveness, we conducted numerical experiments, which revealed that the proposed method is effective across a variety of

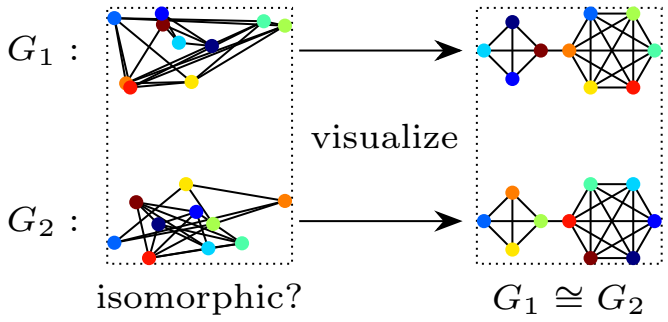


Fig. 15: Illustration of the relationship between graph drawing and graph isomorphism. If we can draw graphs G_1 and G_2 symmetrically, then it is clear that $G_1 \cong G_2$.

graphs. As for the research question posed in Sec. **??**, we can answer that "We can accelerate the optimization process for the FR layout by focusing on each vertex, especially with the subspace Newton direction".

We conclude this paper expecting that the proposed method may advance graph drawing of FR layout and highlight the potential of the subspace methods for addressing a wider range of graph-related optimization problems.

7 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 on the library 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] W. T. Tutte, "How to Draw a Graph," *Proceedings of the London Mathematical Society*, vol. s3-13, no. 1, pp. 743–767, 1963. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1112/plms/s3-13.1.743

[2] M. Chrobak and T. H. Payne, "A linear-time algorithm for drawing a planar graph on a grid," *Information Processing Letters*, vol. 54, no. 4, pp. 241–246, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/002001909500020D

[3] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for Visual Understanding of Hierarchical System Structures," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 2, pp. 109–125, 1981. [Online]. Available: https://ieeexplore-ieee-org.utokyo.idm.oclc.org/document/4308636

[4] 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

[5] 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

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

[7] 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

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

[9] 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.

[10] 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.

[11] 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

[12] 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

[13] 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

[14] 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

[15] E. R. Gansner, Y. Koren, and S. North, "Graph Drawing by Stress Majorization," in *Graph Drawing*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, and J. Pach, Eds. Springer Berlin Heidelberg, 2005, vol. 3383, pp. 239–250. [Online]. Available: http://link.springer.com/10.1007/978-3-540-31843-9_25

[16] 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*, vol. 26, pp. 56–63, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650215000644

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

[18] 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.

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

[20] 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

[21] 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*, vol. 17, pp. 261–272, 2020.

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

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

[24] 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

[25] 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

[26] 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
[27] A. J. Patel, "Hexagonal Grids," Red Blob Games, Tech. Rep., 2013. [Online]. Available: https://www.redblobgames.com/grids/hexagons/
[28] H. Hiroki, "Hari64boli64/FruchtermanReingoldByRandomSubspace." [Online]. Available: https://github.com/hari64boli64/FruchtermanReingoldByRandomSubspace
[29] 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.
[30] 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
[31] 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
[32] R. Nozawa, P.-L. Poirion, and A. Takeda, "Randomized subspace gradient method for constrained optimization," 2023. [Online]. Available: http://arxiv.org/abs/2307.03335
[33] 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
[34] Y. Nesterov and B. Polyak, "Cubic regularization of Newton method and its global performance," *Mathematical Programming*, vol. 108, no. 1, pp. 177–205, 2006. [Online]. Available: https://doi.org/10.1007/s10107-006-0706-8
[35] 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
[36] B. Recht and S. J. Wright, "Optimization for modern data analysis," 2019. [Online]. Available: https://people.eecs.berkeley.edu/~brecht/opt4ml_book/
[37] S. Klus and P. Gelß, "Continuous optimization methods for the graph isomorphism problem," 2023. [Online]. Available: http://arxiv.org/abs/2311.16912

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.