An Initial Placement for Fruchterman–Reingold Layout with Coordinate Newton Direction

Hiroki Hamaguchi Naoki Marumo Akiko Takeda

Abstract—Graph drawing is a fundamental task for visualizing graph structures, with the Fruchterman–Reingold (FR) layout being one of the most widely used layouts. This layout can be interpreted as a solution of a continuous optimization problem. Typically, this problem is 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 obtained with subspace 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 Newton direction defined in the subspace. 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 subspace 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 subspace methods not only in graph drawing but also in broader graph-related applications.

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

1 Introduction

Graph is a mathematical structure representing pairwise relationships between objects, and graph drawing is one of the fundamental tasks in graph theory. Indeed, numerous kinds of layouts and algorithms have been proposed for graph drawing [1], [2], [3], [4]. Among these, one of the most popular strategies is force-directed algorithms.

In force-directed algorithms, we model a graph as a system of particles with forces acting between them. This class of algorithms includes the Kamada–Kawai (KK) layout [5], Eades' spring embedder [6], and the Fruchterman–Reingold (FR) layout [7], [8], which is the central focus of this study.

The FR layout, also known as spring layout or spring-electrical model, is one of the most widely used layouts. It is also implemented in many modern graph drawing libraries such as NetworkX [9], Graphviz [10], and igraph [11].

However, both the KK layout and the FR layout suffer from high computational complexity, specifically $\mathcal{O}(n^2)$ per iteration as it is, where n denotes the number of vertices. This computational burden makes it challenging to achieve high-quality visualizations for large-scale graphs.

To address this kind of burden, we can leverage several methods, such as approximating 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].

Another approach is to directly accelerate the optimization process, which aligns with the aim of our work. Recent research has accelerated the algorithms through various methods, including adaptation to GPU parallel architectures [16], utilizing numerical optimization techniques such as L-BFGS [17], and Stochastic Gradient Descent (SGD) [18].

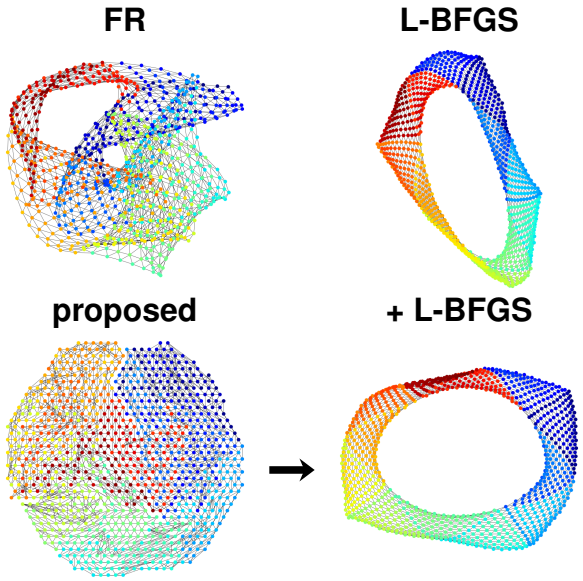

Fig. 1: Comparison of the FR algorithm, L-BFGS algorithm, and the proposed initial placement for the `jagmesh1` dataset.

Based on such advances, in this paper, we propose a new initial placement for the FR layout. Our goal is to accelerate the optimization process by leveraging the inherent structure of the problem. To achieve this, we optimize the position of n vertices one by one with the Newton direction, which is defined in the subspace $\mathbb{R}^2$ in the entire variable space $\mathbb{R}^{2\times n}$. In this way, we provide a high-quality initial placement in advance, improving the overall optimization

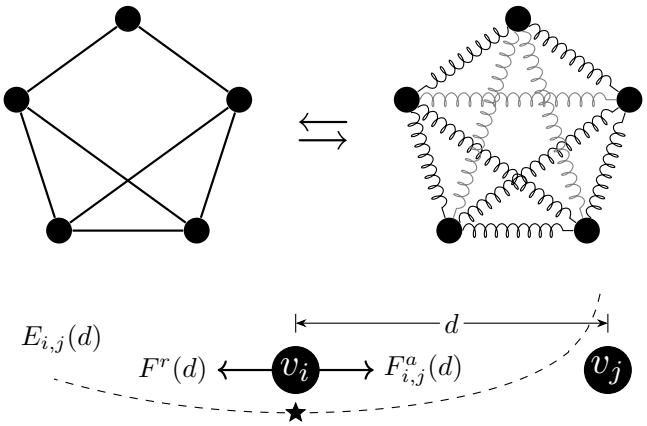

Fig. 2: (Top) Fruchterman–Reingold layout. It models $\mathcal{O}(n^2)$ forces between all pairs of vertices. (Bottom) Forces $F^{\mathrm{a}}_{i,j}(d)$ and $F^{\mathrm{r}}(d)$ work between v_i and v_j. The equilibrium of them is achieved at $d = k/\sqrt[3]{w_{i,j}}$, which equals k when $w_{i,j} = 1$.

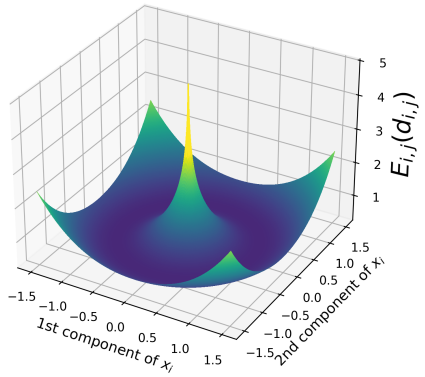

Fig. 3: Energy function $E_{i,j}(\|x_i - x_j\|)$ for $x_j = (0, 0)$, $w_{i,j} = 1$ and $k = 1$. Although $E_{i,j}$ is convex as a function of d, it is not convex as a function of x_i. As x_i approaches x_j, the energy function diverges.

process. We also demonstrate its effectiveness through various experiments.

The rest is as follows. In Sec. 2, we define the optimization problem for the FR layout. In Sec. 3, we present our research question based on previous works. In Sec. 4, we propose a new initial placement algorithm. In Sec. 5, we show the experimental results. In Sec. 6, we discuss the potential of the subspace methods, the key concept of our research. Finally, we discuss future work and conclude the paper in Sec. 7.

2 Preliminary

In this section, we formulate the FR layout as a continuous optimization problem and introduce the conventional approaches to this problem, namely the FR algorithm and the L-BFGS algorithm.

2.1 Fruchterman–Reingold layout

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 $W = (w_{i,j}) \in \mathbb{R}^{n \times n}_{\geq 0}$ 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. 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:

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

Fruchterman and Reingold [7] proposed a force-directed layout called the Fruchterman–Reingold (FR) layout, as known as a spring layout [9] or spring-electrical model [14]. 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 $d_{i,j} \coloneqq \|x_i - x_j\|$ 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

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

FR layout 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

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

As a remark, this energy function $E_{i,j}$ is convex and minimized when $d = k/\sqrt[3]{w_{i,j}}$, but $E_{i,j}$ is not Lipschitz continuous. 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).

2.2 Fruchterman–Reingold algorithm

The Fruchterman–Reingold algorithm [7] is the original force-directed algorithm for this layout, and is a most standard approach for solving the optimization Prob. (3). As mentioned in Ref. [19], the FR algorithm can be regarded as

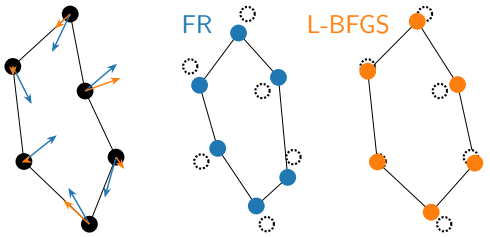

Fig. 4: Comparison of the FR algorithm and the L-BFGS algorithm. Although the FR algorithm moves vertices in a descent direction with a fixed step size (blue arrows), the L-BFGS algorithm adjusts them differently since it utilizes approximated inverse Hessian of f (orange arrows).

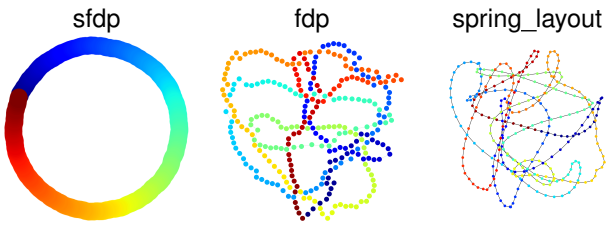

Fig. 5: Comparison of sfdp in Graphviz, fdp in Graphviz, and spring_layout in NetworkX for the cycle graph with $|V| = 300$, `cycle300`. We run every algorithm with the default parameters, but both fdp and spring_layout fail to visualize the cycle beautifully.

a variant of gradient descent (steepest descent) method for the energy function f with a cooling global temperature t.

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

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

The gradient of f_i is

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

which is the sum of forces acting on the vertex v_i.

We show the pseudo-code of the FR algorithm in Algorithm 1. It is based on the original pseudo-code [7] and implementation in NetworkX [9] with some omitted details.

Algorithm 1: Fruchterman–Reingold algorithm

Input: Graph $G_W = (V, E)$
Output: Point placement $X = (x_1, \ldots, x_n)$
Define parameters $k, t, N_{\text{iter}}^{\text{FR}}, \Delta_t \coloneqq t/N_{\text{iter}}^{\text{FR}}$;
Define initial placement of $v_i \in V$ 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;

In Algorithm 1, the initial placement of the n points is determined randomly. Under proper input normalization, each point is uniformly distributed within 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.

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]. The L-BFGS algorithm is in a family of quasi-Newton methods, and iterates procedures at most $N_{\text{iter}}^{\text{L-BFGS}}$ times. Using only a few recent gradient vector, L-BFGS algorithm approximates the inverse Hessian of the objective function f, which is necessary to determine a descent direction [20]. 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.

3 Research Question

Now, we declare the research question of this paper: "Can we accelerate the optimization process for the FR layout by leveraging the inherent structure of the problem?".

In the previous section, we presented the FR algorithm and L-BFGS algorithm. However, both methods face specific challenges during optimization, leading to inefficiencies and slow convergence, as detailed in Sec. 3.1. Addressing these challenges forms a part of our research question, and we aim to develop a new algorithm capable of overcoming these limitations. To achieve this goal, we will review key prior studies in Sections 3.2 and 3.3.

3.1 "twist" in the optimization process

First, we observe the challenges that appeared in existing methods. One of the most challenging aspects of optimizing the FR layout is addressing the issue of "twist" in general.

The term "twist" does not refer to a mathematically rigorous concept; rather, it describes situations where unnecessary intersections of edges or tangled structures appear in visual representations. Although the term "twist" is uncommon, works such as Ref. [24] also mentioned this "twist" phenomenon. Fig. 5 illustrates cycle graph examples of this situation.

The presence or absence of "twists" can impact optimization efficiency. For example, when we draw a cycle graph, even if the current point placement is far from optimal, the optimization method proceeds relatively smoothly if

there are no "twists" in the graph. However, when "twist" exists like in Fig 5, mutual interactions may diminish the gradient, causing the optimization to stagnate. Therefore, we can predict that providing an initial placement of the graph that resolves "twist" as much as possible in advance can significantly influence the efficiency of the optimization process.

3.2 SGD for KK layout

Moreover, the effectiveness of Stochastic Gradient Descent (SGD) for Kama–Kawai (KK) layout is well-documented in [18]. This work has some implications for the FR layout.

The KK layout is an energy-based layout, which minimizes the energy function f^{KK} defined as

$$f_{i,j}^{\mathrm{KK}}(d) \coloneqq w_{i,j}(d - l_{i,j})^2,\ f^{\mathrm{KK}}(X) \coloneqq \sum_{i<j} f_{i,j}^{\mathrm{KK}}(\|x_i - x_j\|),$$

where $l_{i,j}$ is the optimal distance between v_i and v_j calculated from the shortest path distance in the graph G_W.

The SGD method is an optimization method often used in machine learning to minimize a sum of differentiable functions, and Ref. [18] applied its modified version to the KK layout. It randomly selects pairs of vertices (i, j) and adjusts their positions to minimize $f_{i,j}^{\mathrm{KK}}$ along the gradient direction with a learning rate η.

However, in contrast to the KK layout, the FR layout assigns the function $E_{i,j}(d) = -k^2 \log d$ to all $(i, j) \notin E (\iff w_{i,j} = 0)$, making SGD less effective for the FR layout. Nevertheless, the superiority of SGD in the KK layout, which focuses on "randomly selected edge," suggests that an optimization method focusing on "randomly selected vertex" may also be effective in the FR layout, especially when we reduce the "twist" in advance. This observation motivates us to propose a new approach combined with idea in the next subsection.

3.3 Newton Direction and Subspace Newton Direction

In this subsection, we will explain Newton direction and subspace Newton direction, which are the key concepts of our research. Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. The second order approximation of f at $x \in \mathbb{R}^n$ is

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

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

$$\nabla f(x) + \nabla^2 f(x)(x^* - x) = 0$$
$$\iff x^* = x - \nabla^2 f(x)^{-1} \nabla f(x).$$

We call the direction $d = -\nabla^2 f(x)^{-1} \nabla f(x)$ as the Newton direction. Although Newton direction is the optimal direction for the approximation, it requires the computation of the inverse Hessian $\nabla^2 f(x)^{-1} \in \mathbb{R}^{n \times n}$, 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.

However, by combining this idea with the concept of subspace, the computational cost issue can be addressed in other ways. The computational cost of computing the inverse Hessian $\nabla^2 f(x)^{-1}$ increase with respect to the dimension n. Thus, by limiting the variable space to its subspaces, we can reduce the computational cost as well. We call the Newton direction defined in the subspace as the subspace Newton direction. This idea is well known as the subspace method or Randomized Subspace Newton (RSN) [25]. Refer to Fig. 6 for an illustration of the subspace method.

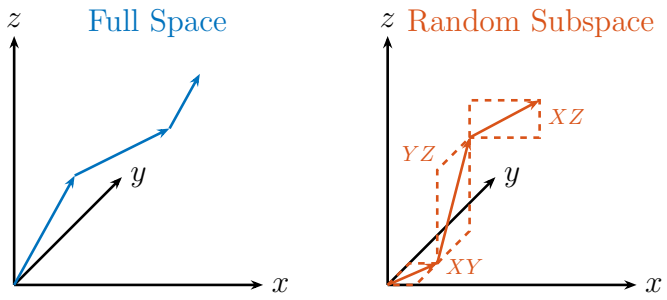

Fig. 6: TODO

Although directly applying this idea to the FR layout is challenging as we will discuss in Sec. 6, we leverage this subspace Newton direction to propose a new algorithm for the FR layout in the next section.

4 Proposed Algorithm

To answer the research question above, we propose a new algorithm for the FR layout that utilizes the subspace Newton direction. We describe our proposed method in three stages. Firstly, in Sec. 4.1, we reformulate the optimization problem (3) into a simplified discrete optimization problem using a hexagonal lattice. Secondly, in Sec. 4.2, we present a method to solve the discrete optimization problem through continuous relaxation, using the Newton direction for vertices selected randomly, which is the core of our proposed algorithm. Finally, in Sec. 4.3, we show the complete framework of the proposed method, where we use the solution obtained from the previous step as the initial solution.

4.1 reduction to the discrete optimization problem

First, we transform the optimization problem (3) into a constrained continuous optimization problem. As written in Sec. 3.2, 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 a reasonable approach:

$$\underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad \sum_{(i,j)\in E} \frac{w_{i,j}\|x_i - x_j\|^3}{3k} - \sum_{i<j} k^2 \log \|x_i - x_j\|. \tag{5}$$

Further, by converting the second term into a constraint, the problem (5) can be approximated such that the objective function can be computed with a complexity dependent on $|E|$ rather than $|V|^2$:

$$\begin{aligned} \underset{X \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad & f^{\mathrm{a}}(X) \coloneqq \sum_{(i,j)\in E} \frac{w_{i,j}\|x_i - x_j\|^3}{3k} \\ \text{subject to} \quad & d_{i,j} = \|x_i - x_j\| \geq \epsilon, \quad \forall (i,j)\,(i<j) \end{aligned} \tag{6}$$

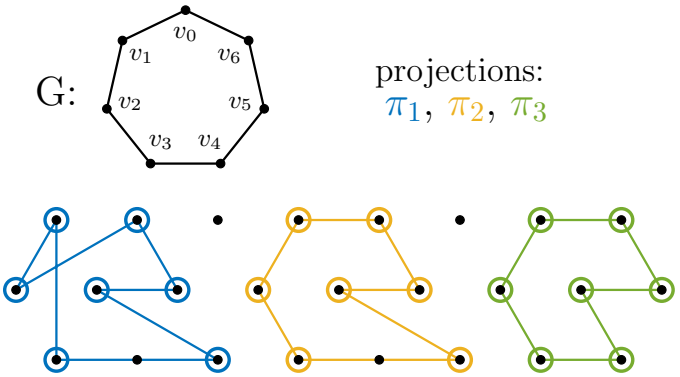

Fig. 7: concept of π. The injection π maps vertices V to a discrete point placement Q. Apparently, among π_1, π_2, π_3, π_3 on the right one is the best mapping for the Prob. (7).

where ϵ is a suitably chosen positive constant. This conversion is reasonable 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, ensuring the validity of the approximation.

However, problem (6) 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 fixed initial placements for the FR layout [4]. This study reports that a cycle initial placement obtained by Simulated Annealing (SA) brings a rapid convergence to a better solution in the FR layout. Similarly, by simplifying problem (6) with a fixed initial placement Q whose points are separated by at least ϵ, we obtain the following discrete optimization problem:

$$\begin{aligned} &\underset{\pi : V \to Q}{\text{minimize}} && \sum_{(i,j)\in E} \frac{w_{i,j}\|\pi(v_i) - \pi(v_j)\|^3}{3k} \\ &\text{subject to} && \pi(v_i) \neq \pi(v_j), \quad \forall (i,j)\,(i<j) \end{aligned} \tag{7}$$

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.

As an example of such a discrete point placement Q, one could consider n circles of radius ϵ packed in $\mathbb{R}^2$. In this study, however, we adopt a hexagonal lattice [27], [26]. The hexagonal lattice is known for its densest packing structure in space and offers computational simplicity. See Figure 7 for reference.

4.2 the Newton direction for discrete optimization

Next, using the Newton direction of a randomly selected vertex, we solve the discrete optimization problem through continuous relaxation. The computation of the Newton direction in this part is essentially identical to the RSN method described in Sec. 6.1.

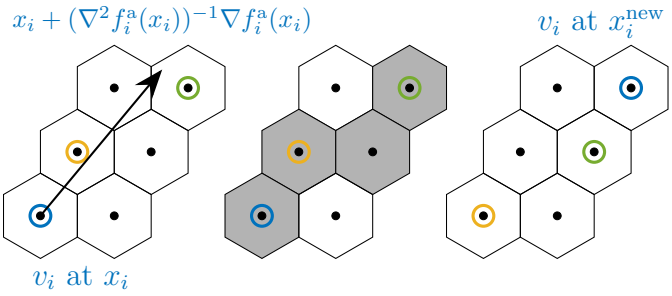

Fig. 8: Visual explanation of the one iteration of the proposed algorithm. Step1. Compute the Newton direction for a randomly selected vertex. Step2. decide the path from the original position to the new position. Step3. move the vertex along the path.

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

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

Using the Newton direction, we can define

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

where $\mathrm{round}(\hat{x})$ denotes the operation assigning $\hat{x}$ to the nearest point in the hexagonal lattice Q.

Simply changing the assignment $\pi(v_i)$ from vertex v_i to Q, updating from x_i to x_i^{new}, may not preserve the injectivity of π. To address this, consider the path from x_i to x_i^{new}, which can be determined using linear interpolation for line drawing [27]. If each point on this path within Q is assigned to vertices such as $v_i, v_j, \mathrm{NaN}, v_k$, then by a cyclic shift, we reassign in the sequence $v_j, \mathrm{NaN}, v_k, v_i$. This approach ensures that vertex v_i can be reassigned to its new position x_i^{new} while maintaining the injectivity of π. We illustrated the overall process in Fig. 8. This approach can obtain a high-quality initial solution for the optimization Prob. (3).

4.3 pseudo code

We presented the overall framework of the proposed method in Algorithm 2. The output of this algorithm is a solution to Prob. (7). To obtain the solution to Prob. (3), the output must be further refined using the FR algorithm or the L-BFGS algorithm.

4.4 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 as

$$\phi(c) \coloneqq \left(\sum_{i<j} \frac{w_{ij}(cd_{ij})^3}{3k}\right) - k^2 \sum_{i<j} \log(cd_{ij})$$

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 k, $N_{\text{iter}}^{\text{SN}}$ and hexagonal lattice Q;
set π as a injection from V to Q randomly;
for $j \leftarrow 0$ **to** $N_{\text{iter}}^{\text{SN}}$ **do**
  $v_i \leftarrow$ randomly selected vertex from V;
  $x_i \leftarrow \pi(v_i)$;
  $x_i^{\text{new}} \leftarrow \text{round}(x_i - \nabla^2 f_i(x_i)^{-1} \nabla f_i(x_i))$;
  path $\leftarrow$ path from x_i to x_i^{new};
  update π by cyclic shifting along the path;
  if *convergence condition* **then**
    break;
$x_i \leftarrow \pi(v_i)$ for all $v_i \in V$;
return X

$$= c^3 \left(\sum_{i<j} \frac{w_{ij} d_{ij}^3}{3k} \right) - k^2 n(n-1) \log(c) - k^2 \sum_{i<j} \log(d_{ij}),$$

$$\phi'(c) = 3c^2 \left(\sum_{i<j} \frac{w_{ij} d_{ij}^3}{3k} \right) - \frac{k^2 n(n-1)}{c}.$$

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

$$c^* = \left(\frac{k^2 n(n-1)}{3 \sum_{i<j} \frac{w_{ij} d_{ij}^3}{k}} \right)^{1/3}.$$

5 Numerical Experiment

In this section, we evaluate the proposed algorithm by various numerical experiments based on the setup described in Sec. 5.1. 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. 5.2, and detailed experiments to investigate the behavior of the proposed algorithm in detail in Sec. 5.3.

5.1 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. 5.

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

As a parameter, we used $N_{\text{iter}}^{\text{SN}} = 3\frac{|V|^3}{|E|}$ 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].

5.2 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 $|V| \leq 1000$, in total 124 graphs.

The result is shown in Fig. 14. 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. 10, Fig. 11, Fig. 12, and Fig. 13. We can observe that the reasons why it was worse though these figures. todo

5.3 detailed Experiment

We also conducted a detailed experiment to investigate the behavior of the proposed algorithm in detail. The result is shown in Fig. 14.

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. 14 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-SN algorithms, while faint circles represent the results of SN 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. 14 are as follows. First, the solid plots for SN generally demonstrates

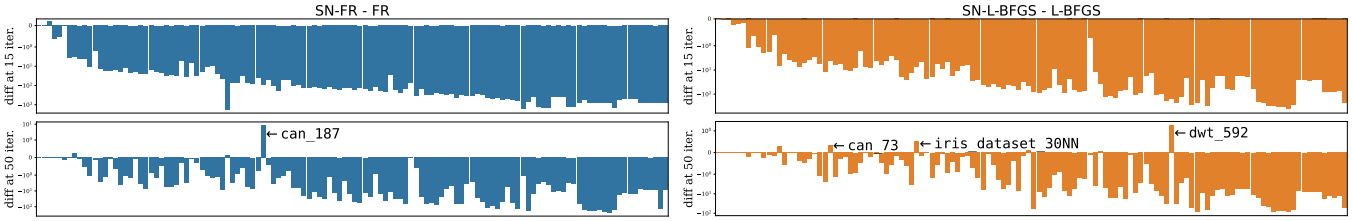

Fig. 9: Comparison of the proposed initialization with random initialization. Since $N_{\text{iter}}^{\text{SN}} = 3\frac{|V|^3}{|E|}$, we set $N_{\text{iter}}^{\text{FR}}$ and $N_{\text{iter}}^{\text{L-BFGS}}$ as 12(=15-3), 47(=50-3), respectively for the SN algorithms. For non-SN algorithms, we set them as 15 and 50, respectively. The y axis is symlog scale. Almost all of the cases, the difference are negative, meaning that the proposed algorithm performed better than random initialization.

Fig. 10: `can_187`

Fig. 11: `can_73`

Fig. 12: `iris_dataset_30NN`

Fig. 13: `dwt_592`

superior performance compared to non-SN, 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 SN-FR is clearly 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 SN better represent the intended geometric arrangement compared to non-SN placements.

As a side note, regardless of SN or non-SN, 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.

6 Challenges of the Subspace Methods

Here, we explain the rationale for why we took such a roundabout approach to solve the optimization Prob. (3). As we have explained, we first transformed it into a discrete optimization Prob. (7) and then solved it using the subspace Newton direction.

Initially, we attempted a direct application of the subspace method, one of the general optimization methods, to the FR layout as described in Section 3.3. However, we encountered several issues that made this approach challenging: the inaccuracy of the quadratic approximation and the ignorance of other vertex movements. We state these issues in Sec. 6.2 and Sec. 6.3 Based on these challenges, we explain the rationale for our approach in Sec. 6.4.

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

6.1 Introduction of Randomized Subspace Newton

In this subsection, we introduce the Randomized Subspace Newton (RSN), which is NOT a proposed method but a concept that heavily inspired our proposed algorithm. The RSN method is one of the subspace methods, and RSN and its variant have been proposed in the context of optimization problems [25], [30], [31], [32], [33].

RSN focuses on a subspace of dimension s randomly selected from the solution space by a projection matrix $S \in \mathbb{R}^{n\times s}$ and utilizes the exact Hessian matrix of size $s \times s$ defined on this subspace: $S^\top \nabla^2 f(x) S$. At each iteration, RSN updates the solution by

$$x \leftarrow x - S(S^\top \nabla^2 f(x) S)^{-1} S^\top \nabla f(x)$$

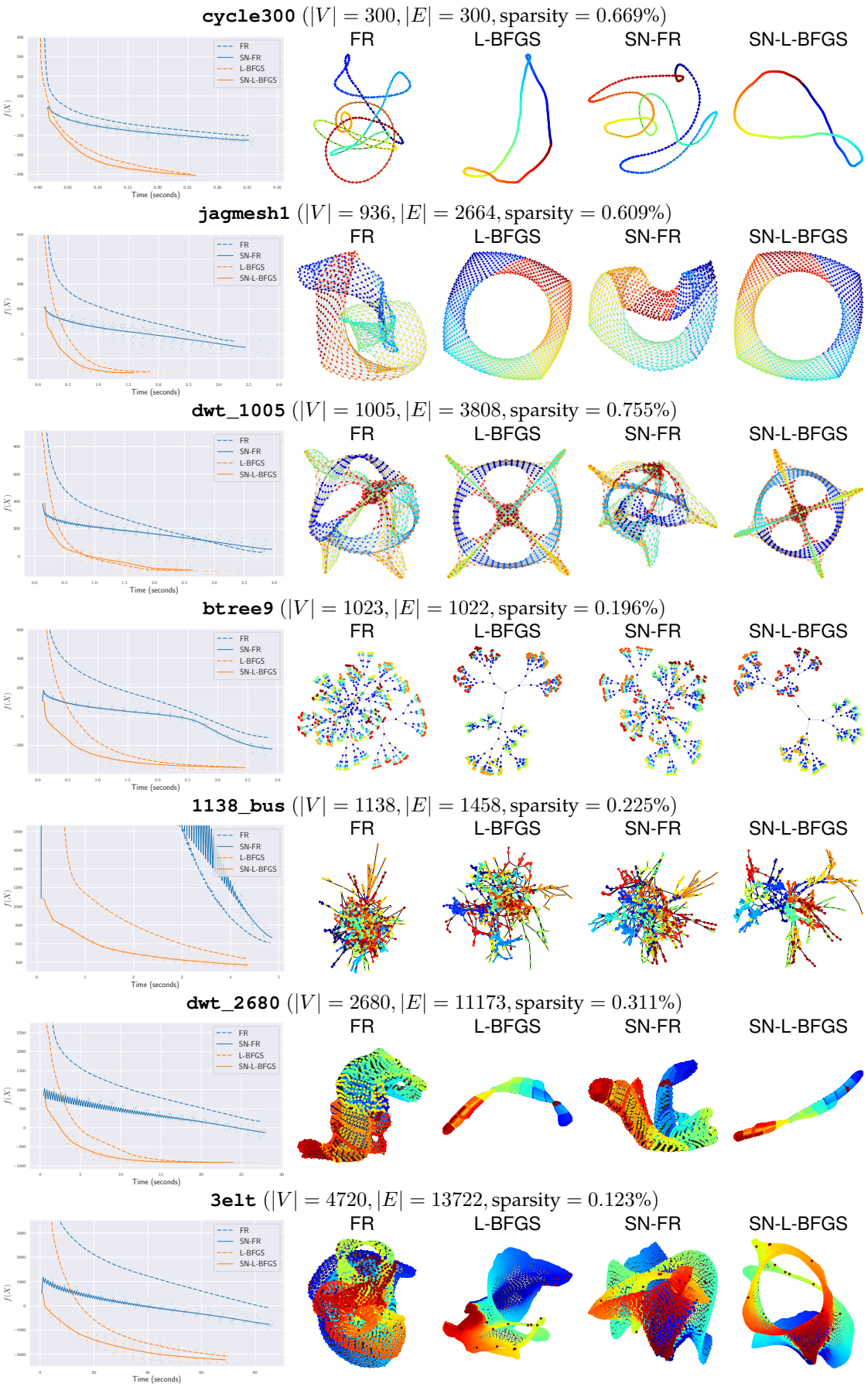

Fig. 14: Numerical experiment results for various graphs. Please refer Sec. 5.3 for details.

Fig. 15: 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.

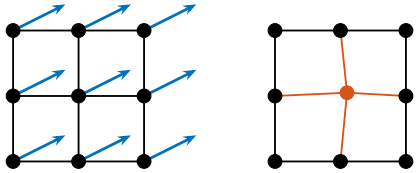

Fig. 16: The ignorance of other vertex movements. Assume that the blue arrows show the forces to the vertices and the optimal movements in this situation. Nevertheless, the red vertex will only move a little by the subspace method.

if $S^\top \nabla^2 f(x) S$ is non-singular. Since $s \ll n$, the computational cost per iteration is significantly reduced when we disregard the cost of selecting the subspace.

The RSN method resembles the stochastic coordinate descent method, which updates only a subset of the variables at each iteration using gradient information. The difference is that RSN uses the Hessian matrix to determine the update direction, bringing the method closer to the Newton method.

Moreover, recent studies have explored its application not only to convex optimization problems but also to non-convex optimization problems [30], using a regularization term to ensure the convergence of the method.

In particular, our Prob. (3) exhibits a natural affinity with the RSN method, as it inherently defines a subspace of the solution space X for the FR layout: $x_i \in \mathbb{R}^2$ for all $v_i \in V$. If take S as a projection matrix that selects a random vertex v_i from V, we can consider to apply the RSN method or its variant to the FR layout.

Despite of such high expectation for the subspace methods, we found that direct applying of the subspace method to the FR layout is quite challenging. 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. (4) 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.

6.2 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.

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. 15. 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.

6.3 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 16. 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.

6.4 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 4.1, we can take this ignored interaction into account to some extent. The advantage of Prob. (7) 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.

7 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. 7.1. Secondly, we explore the applications beyond the scope of graph drawing in Sec. 7.2. Finally, we conclude this paper in Sec. 7.3.

7.1 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. 5 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.

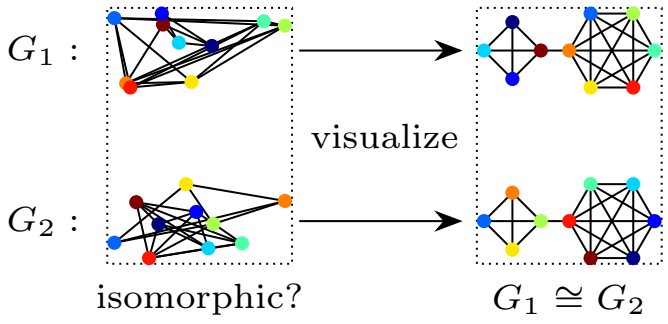

Fig. 17: Illustration of the relationship between graph drawing and graph isomorphism. If we can draw graphs G_1 and G_2 symmetrically, then it is clear that $G_1 \cong G_2$.

7.2 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. 6.1, 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. 17 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.

7.3 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 graphs. As for the research question posed in Sec. 3, 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.

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 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

APPENDIX A
ANOTHER APPROACH BASED ON THE PROPOSED METHOD

In this section, we discuss alternative approaches to the Prob. (7) other than the proposed method.

First, one might consider using only gradient descent, not the Newton direction in Algorithm 2. However, we were unable to achieve satisfactory results with this approach in our preliminary experiments. Given that the subspace dimension is as small as 2, employing the Newton direction seems considerably more efficient.

Secondly, we can also consider an approach that updates all vertices simultaneously, updating all $|V|$ vertices simultaneously like the FR algorithm. First, we compute the search direction for each vertex v_i using the subspace Newton direction, as in Algorithm 2. Then, move all the points $\{x_i\}_{1\leq i\leq |V|} \subseteq Q$ on the hexagonal grid to arbitrary points $\{\hat{x}_i := x_i - \nabla^2 f_i(x_i)^{-1}\nabla f_i(x_i)\}_{1\leq i\leq |V|} \subseteq \mathbb{R}^2$, and then assign all these $|V|$ points to the nearest points in Q. The assignment problem from the $\mathbb{R}^2$ to Q can be formalized as finding the optimal mapping $\pi : V \to Q$ that minimizes the distance between the new positions $\hat{x}_i$ and the assigned points in Q, which can be formulated as:

$$\begin{aligned} \underset{\pi\,:\,V\to Q}{\text{minimize}} \quad & \sum_{i=1}^{|V|} \|\hat{x}_i - \pi(v_i)\|^2 \\ \text{subject to} \quad & \pi(v_i) \neq \pi(v_j), \quad \forall v_i, v_j \in V, (v_i \neq v_j) \end{aligned}$$

This assignment problem can be solved in $\mathcal{O}\left(|V|^3\right)$ using methods such as the minimum-cost flow or Hungarian algorithm. Additionally, we can obtain a heuristic solution in $\mathcal{O}(|V|\log|V|)$ by appropriately sorting $\{\hat{x}_i^{\text{new}}\}_{1\leq i\leq |V|}$.

While we are not expecting as good performances as the proposed method from the approaches above, they offer certain advantages, such as simplifying implementation or avoiding random access to arrays.

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.