Fruchterman–Reingold Layout with Subspace Method as Initial Placement

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 to a continuous optimization problem. Typically, this problem is solved using the FR algorithm, a gradient descent-like method, or the L-BFGS, 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 by the subspace method. 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 method to obtain the final configuration.

We demonstrate the effectiveness of our proposed approach through experiments, highlighting the potential of the subspace method 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, Subspace Method, Random Subspace Newton.**

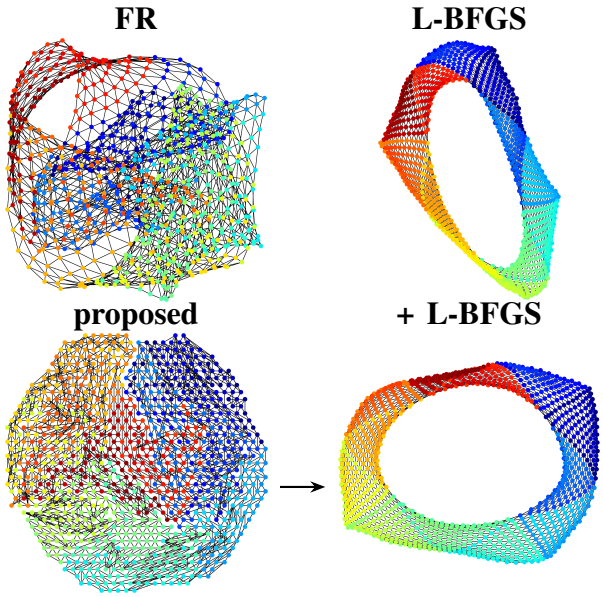

Fig. 1: Comparison of the FR algorithm, L-BFGS, and the proposed method for the `jagmesh1` dataset.

I. 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 $\mathcal{O}\left(|V|^2\right)$ per iteration as it is, where $|V|$ 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 for FR layout or KK layout 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].

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 employ a subspace method, which means randomly select a vertex and move it along the Newton direction defined in the subspace. In this way, we provide a high-quality initial configuration in advance, improving the overall optimization process of the FR layout. We also demonstrate its effectiveness through experiments.

The rest of the paper is organized as follows. In Sec. II, we define the optimization problem for the FR layout. In Sec. III, we present our research question based on previous works. In Sec. IV, we propose a new algorithm that utilizes the subspace method for the FR layout. In Sec. V, we present our experimental results. Finally, we discuss future work and

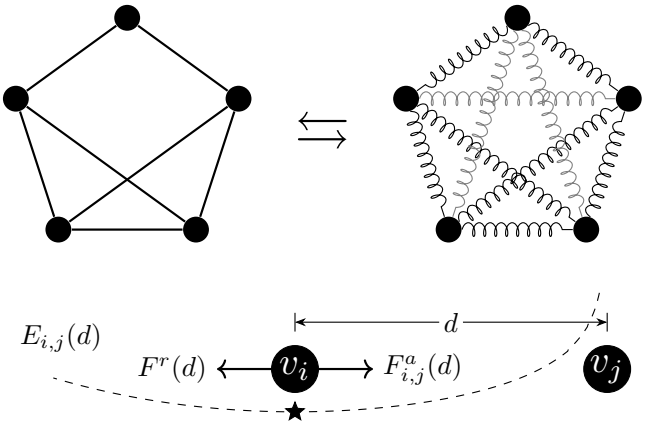

Fig. 2: (Top) Fruchterman–Reingold layout. It models $\mathcal{O}(n^2)$ forces between all pairs of vertices. (Bottom) Forces $F^a_{i,j}(d)$ and $F^r(d)$ work between v_i and v_j. The equilibrium of them is achieved at $d = k/\sqrt[3]{w_{i,j}}$, which equals k when $w_{i,j} = 1$.

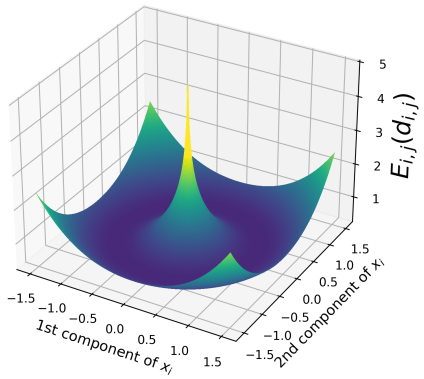

Fig. 3: Energy function $E_{i,j}(d_{i,j})$ for $x_j = (0, 0)$, $w_{i,j} = 1$ and $k = 1$. Although $E_{i,j}$ is convex as a function of $d_{i,j}$, it is not convex as a function of x_i. As x_i approaches x_j, the energy function diverges.

conclude the paper in Sec. VI.

II. 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 method.

A. 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 problem (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 configuration of the graph. For a parameter k and a distance $d_{i,j} \coloneqq \|x_i - x_j\|_2$ between two vertices v_i and v_j, the attraction force $F^a_{i,j} : \mathbb{R}_{>0} \to \mathbb{R}$ and the repulsion force $F^r : \mathbb{R}_{>0} \to \mathbb{R}$ is defined as

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

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

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

As a remark, this energy function $E_{i,j}$ is convex as a function of d and minimized when $d^* = k/\sqrt[3]{w_{i,j}}$. Though, $E_{i,j}$ is not Lipschitz continuous and is not convex as a function of x_i for a fixed x_j. Refer to Fig. 3.

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

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

Seeking an equilibrium of the forces is equivalent to minimizing the energy function $E_{i,j}$ for all pairs of vertices. In the following, we will discuss the optimization of this problem.

B. Fruchterman–Reingold algorithm

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

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

The gradient of f_i is

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

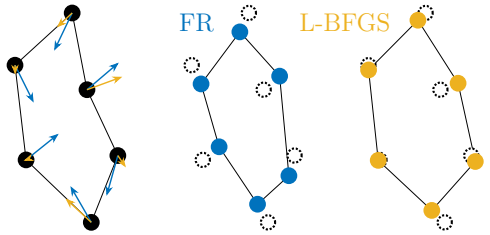

Fig. 4: Comparison of the FR algorithm and the L-BFGS method. Although the FR algorithm moves vertices in a descent direction with a fixed step size (blue arrows), the L-BFGS method adjusts them differently since it utilizes approximated inverse Hessian of f (red arrows).

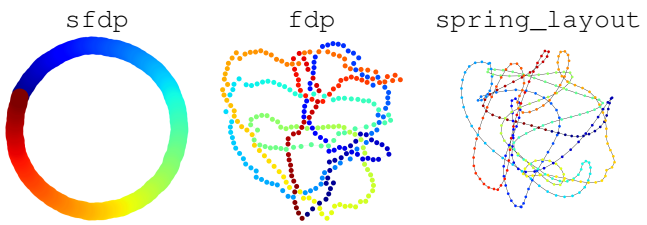

Fig. 5: Comparison of `sfdp` in Graphviz, `fdp` in Graphviz, and `spring_layout` in NetworkX for the cycle graph with $|V| = 300$. We run every algorithm with the default parameters, but both `fdp` and `spring_layout` fail to visualize the cycle beautifully.

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

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

Algorithm 1: Fruchterman–Reingold algorithm

Input: Graph $G_W = (V, E)$
Output: Point configuration $X = (x_1, \ldots, x_n)$
Define parameters k, t, dt, iterations;
Define initial placement of $v_i \in V$ randomly;
for $j \leftarrow 0$ **to** *iterations* **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 $v_i \in V$;
  $t \leftarrow t - dt$;
  if *convergence condition* **then**
    break;
return X;

In Algorithm 1, the initial placement of 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 configuration, though this configuration is not necessarily the optimal solution.

C. L-BFGS method

Another approach to solve the optimization problem (3) is to use the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [17]. The L-BFGS method is an optimization technique used for large-scale problems, which approximates the inverse Hessian matrix using only a few recent gradient vectors, making it more memory-efficient than the standard BFGS method (quasi-Newton method) [20]. Thus, L-BFGS is particularly suitable for high-dimensional optimization tasks. L-BFGS is more sophisticated than the gradient descent method, and the superior performance of the L-BFGS method to the FR algorithm reported in Ref.[17] also indicates this fact. Also refer to Fig. 4.

There are many implementations of L-BFGS available, such as SciPy [21] and C++ L-BFGS [22], [23], which we used in our experiments.

For the optimization problem (3), we can apply the L-BFGS method via flattening the configuration $X \in \mathbb{R}^{2\times n}$ 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 method.

III. 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 method. However, both methods face specific challenges during optimization, leading to inefficiencies and slow convergence, as detailed in Sec. III-A. 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 III-B and III-C.

A. "twist" in the optimization process

First, we observe the challenges that appeared in existing methods. One of the more 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 graph configuration 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.

B. 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

$$f_{i,j}^{\mathrm{KK}}(d_{i,j}) \coloneqq w_{i,j}(d_{i,j} - l_{i,j})^2, \quad f^{\mathrm{KK}}(X) \coloneqq \sum_{i<j} f_{i,j}^{\mathrm{KK}}(d_{i,j}),$$

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:

$$x_i \leftarrow x_i - \frac{\min(w_{i,j}\eta, 1)}{4w_{i,j}} \frac{\partial}{\partial x_i} f_{i,j}^{\mathrm{KK}}(d_{i,j}),$$
$$x_j \leftarrow x_j - \frac{\min(w_{i,j}\eta, 1)}{4w_{i,j}} \frac{\partial}{\partial x_j} f_{i,j}^{\mathrm{KK}}(d_{i,j})$$

with a learning rate η.

However, in contrast to the KK layout, the FR layout assigns the function $E_{i,j}(d_{i,j}) = -k^2 \log d_{i,j}$ to all $(i, j) \notin E(\iff w_{i,j} = 0)$, making SGD less effective for the FR layout. However, 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. This observation motivates us to explore the application of Random Subspace Newton (RSN) to the FR layout, as discussed in the next subsection.

C. Introduction of Random Subspace Newton

We introduce the RSN method, which is NOT a proposed method but a concept that heavily inspired our proposed algorithm. The RSN method and its variant have been proposed in the context of optimization problems [25], [26], [27], [28], [29].

First, we briefly explain the Newton's method. This method updates variable $x \in \mathbb{R}^n$ of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ by

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

Since f is convex, the Hessian matrix $\nabla^2 f(x) \in \mathbb{R}^{n \times n}$ is positive semi-definite, and the update direction (Newton direction) is a descent direction. Plus, the updated x is an optimal solution of the quadratic approximation of f at x, ensuring Newton's method's fast convergence. Newton's method requires the computation of the inverse Hessian matrix $\nabla^2 f(x)^{-1} \in \mathbb{R}^{n \times n}$ at each iteration, posing a high computational cost for large-scale problems.

In contrast, 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

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

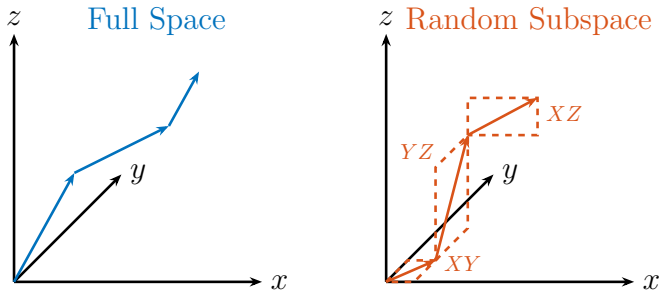

Fig. 6: Visual explanation of Random Subspace Newton TODO

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 [26], using a regularization term to ensure the convergence of the method.

In particular, our problem (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 $v_i \in V$. Although its direct application to the FR layout is ineffective, as we depicted in Sec. B, we exploit this idea in the proposed algorithm, as discussed in the next section.

IV. Proposed Algorithm

To answer the research question above, we propose a new algorithm for the FR layout that utilizes the subspace method. We describe our proposed method in three stages. Firstly, in Section IV-A, we reformulate the optimization problem (3) into a simplified discrete optimization problem using a hexagonal lattice. Secondly, in Section IV-B, 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 Section IV-C, we show the complete framework of the proposed method, where we use the solution obtained from the previous step as the initial solution.

A. reduction to the discrete optimization problem

First, we transform the optimization problem (3) into a constrained continuous optimization problem. As written in Sec. III-B, the energy function $E_{i,j}$ is $-k^2 \log d_{i,j}$ for all $(i, j) \notin E$. Considering the sparsity of many practical graphs ($|E| \ll |V|^2$), simplifying as follow is a reasonable approach:

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

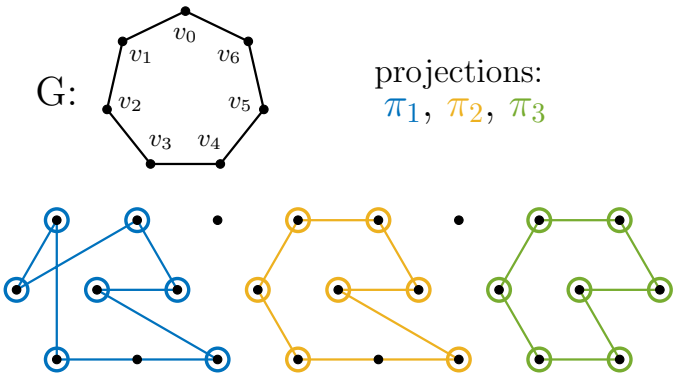

Fig. 7: concept of π. The injection π maps vertices V to a discrete point configuration Q. Apparently, among π_1, π_2, π_3, π_3 on the right one is the best mapping for the Problem (7).

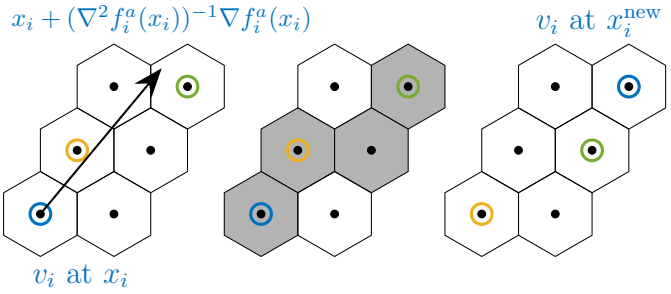

Fig. 8: Visual explanation of the one iteration of the proposed algorithm. Step1. Compute the Newton direction for a randomly selected vertex. Step2. decide the path from the original position to the new position. Step3. move the vertex along the path.

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 $|V|^2$:

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

where ϵ is a suitably chosen positive constant. This conversion is reasonable because $-k^2 \log d_{i,j}$ is a convex function such that it decreases monotonically concerning $d_{i,j}$. Thus, for sufficiently large $d_{i,j}$, the value of $-k^2 \log d_{i,j}$ 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 configurations 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), we obtain the following discrete optimization problem:

$$\begin{aligned} \underset{\pi\,:\,V \to Q}{\text{minimize}} \quad & f^a(X) = \sum_{(i,j)\in E} \frac{w_{i,j} d_{i,j}^3}{3k} \\ \text{subject to} \quad & \|q_i - q_j\| \geq \epsilon, && \forall q_i, q_j \in Q\,(q_i \neq q_j), \\ & x_i = \pi(v_i), && \forall v_i \in V, \\ & \pi(v_i) \neq \pi(v_j), && \forall v_i, v_j \in V\,(v_i \neq v_j). \end{aligned} \tag{7}$$

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 configuration 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 configuration Q, one could consider n circles of radius ϵ packed in $\mathbb{R}^2$. In this study, however, we adopt a hexagonal lattice. The hexagonal lattice is known for its densest packing structure in space and offers computational simplicity. See Figure 7 for reference. Although not directly related to this study, prior research [30] has also pointed out the connection between FR layouts and hexagonal lattices.

B. 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. III-C.

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

$$f_i^a(x_i) \coloneqq \sum_{j\neq i} \frac{w_{i,j}\|x_i - x_j\|_2^3}{3k}.$$

Using the Newton direction, we can solve the discrete optimization problem through continuous relaxation, i.e., by updating the position of a vertex v_i as

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

Specifically, we update the position x_i of the vertex v_i according to the Newton direction and project this new position onto the nearest point on a hexagonal lattice, the new position of the vertex x_i^{new}. Additionally, by sequentially moving vertices from their original positions to the new ones, we can satisfy the constraints of the discrete optimization problem. We illustrated the overall process in Fig. 8.

This approach can obtain a high-quality initial solution for the optimization problem (3).

C. pseudo code

We presented the overall framework of the proposed method in Algorithm 2. The output of this algorithm is not a solution to Problem (3), but rather a solution to the discrete optimization problem (7). To obtain the solution to Problem (3), the output must be further refined using the FR algorithm or the L-BFGS method.

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

Input: Graph $G_W = (V, E)$, subspace dimension s
Output: Point configuration $X = (x_1, \ldots, x_n)$
define parameters k, *iterations* and hexagonal lattice Q;
set π as a injection from V to Q randomly;
for $j \leftarrow 0$ **to** *iterations* **do**
  $v_i \leftarrow$ randomly selected vertex from V;
  $q_i \leftarrow \pi(v_i)$;
  $x_i^{\text{new}} \leftarrow x_i - \nabla^2 f_i(x_i)^{-1} \nabla f_i(x_i)$;
  $q_i' \leftarrow$ nearest point of x_i^{new} in Q;
  path $\leftarrow$ path from q_i to q_i';
  update π by moving q_i along the path;
  if *convergence condition* **then**
    break;
$x_i \leftarrow q_i$ for all $v_i \in V$;
return X

V. Numerical Experiment

In this section, we evaluate the proposed algorithm by various numerical experiments based on the setup described in Sec. V-A. 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. V-B, and detailed experiments to investigate the behavior of the proposed algorithm in detail in Sec. V-C.

A. 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.

We used Graphviz version 2.43.0 [10] and NetworkX version 3.3 [9] during the experiments, including Fig. 5. To implement FR-algorithm and L-BFGS method, we referenced NetworkX and C++ L-BFGS [22], [23], respectively. We also referenced the open-source code of the hexagonal grid from [31].

All the codes are available at our GitHub [**?**].

B. Exhaustive Experiment

First, 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 [32] satisfying the condition (1) with $|V| \leq 1000$.

C. detailed Experiment

Secondly, we conducted a detailed experiment to investigate the behavior of the proposed algorithm in detail.

TODO

提案手法に基づく初期配置を用いた方が、高速に良質な解へと収束していることが確認できる。

VI. Discussion

In this section, we assess the potential of the proposed algorithm and 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. VI-A. Secondly, we explain some rationale of our approach in Sec. VI-B. Thirdly, we explore the applications beyond the scope of graph drawing in Sec. VI-C. Finally, we conclude this paper in Sec. VI-D.

A. 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 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.

B. Why We Proceed Through Discrete Optimization Problems

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

First of all, we consider that directly applying the subspace method to the problem (3) is quite challenging. Initially, we attempted such a direct application, but encountered several issues and found the optimization process to be inefficient. While these reasons are elaborated in Appendix B, we will summarize the key points here as well.

We believe that, while the subspace method is effective for reducing the "twist" in a rough sense, it is not suitable for optimizing the overall placement. In the subspace method, optimization is performed by focusing on a single vertex v_i, ignoring forces acting on other vertices v_j and v_k. This ignorance of interactions between other vertices can lead to suboptimal improvements, or negative effects on the overall placement. This is a significant drawback of the subspace method, and thus applying the subspace method directly did not efficiently yield a global optimal solution.

However, by converting the problem as stated in Section IV-A, we can take this ignored interaction into account to some extent. The advantage of problem (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.

It is important to note that these considerations do not necessarily preclude the possibility of achieving global optimization with the subspace method. Focusing on optimizing each vertex individually is a natural and valid approach, and we believe that with further refinements, the subspace method holds the potential to efficiently yield global optimal solutions.

C. Application to Other Problems

Although we proposed a subspace-based algorithm only for the optimization problem (3), we can see that the application of the subspace method is not necessarily limited to the FR layout alone. In this section, we briefly discuss and explore the potential applicability of the subspace method to a wider range of problems.

In general, the optimization problem (3) is more broadly treated as "objective functions arising from graphs" [33]:

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

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

We can expect the subspace method to be effective for such problems, as shown in this study. The authors of [33] 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. III-C, this claim supports the potential of the subspace method for a wider range of problems.

For instance, we suspect that the subspace method can also be applied to the graph isomorphism problem. 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. 9 for reference. When we relax it to a continuous optimization problem on Riemannian manifolds as in [34], [34], 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.

D. Conclusion

In this study, we proposed a new algorithm for the FR Layout that utilizes the subspace method. To demonstrate its effectiveness, we conducted numerical experiments, which revealed that the proposed method is effective across a variety of graphs.

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

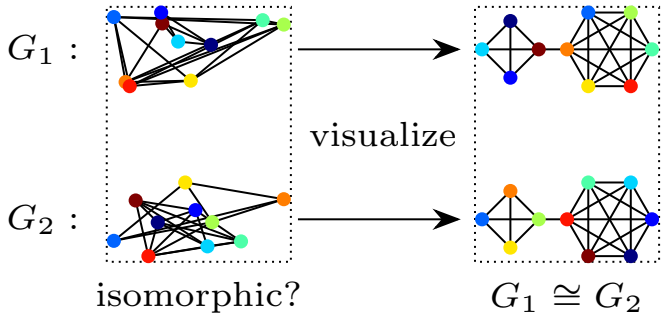

Fig. 9: Illustration of the relationship between graph drawing and graph isomorphism. If we can draw graphs G_1 and G_2 symmetrically, then it is clear that $G_1 \cong G_2$.

VII. 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] 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

[27] 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

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

[29] 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

[30] 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

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

[32] 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.

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

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

[35] 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

APPENDIX A
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 section, we explain how to find the optimal scaling factor that minimizes the energy function for a given configuration.

A. Optimal Scaling Algorithm

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

$$\begin{aligned}
\phi(c) &:= \left(\sum_{i<j} \frac{w_{ij}(cd_{ij})^3}{3k}\right) - k^2 \sum_{i<j} \log(cd_{ij}) \\
&= c^3 \left(\sum_{i<j} \frac{w_{ij} d_{ij}^3}{3k}\right) - k^2 n(n-1)\log(c) \\
&\quad - k^2 \sum_{i<j} \log(d_{ij}), \\
\phi'(c) &= 3c^2 \left(\sum_{i<j} \frac{w_{ij} d_{ij}^3}{3k}\right) - \frac{k^2 n(n-1)}{c}, \\
\phi''(c) &= 6c \left(\sum_{i<j} \frac{w_{ij} d_{ij}^3}{3k}\right) + \frac{k^2 n(n-1)}{c^2}.
\end{aligned}$$

The function $\phi(c)$ is convex, and we can find the optimal scaling factor c by using Newton's method. This algorithm achieves sufficient convergence within a few iterations, and by pre-computing the coefficients of $\phi(c)$ with $w_{i,j} > 0$, the time complexity is just $\mathcal{O}(|E|)$.

APPENDIX B
CHALLENGES OF THE SUBSPACE METHOD FOR THE FR LAYOUT

In this paper, we did NOT propose a direct application of the subspace method to the FR layout; instead, we proposed a initial placement algorithm based on the subspace method.

This is because we consider that directly applying 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

Fig. 10: 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.

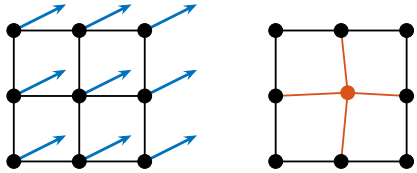

Fig. 11: The ignorance of other vertex movements. Assume that the blue arrows show the forces to the vertices and the optimal movements in this situation. Nevertheless, the red vertex will only move a little by the subspace method.

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:

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

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.

It is important to note that these challenges do not necessarily imply fundamental limitations of the subspace method. On the contrary, improvements based on these identified issues could potentially enhance the effectiveness of the subspace method.

A. *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 [35], 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. 10. 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.

In order to address this issue and enhance the effectiveness of the subspace method for the FR layout, we believe that it may be necessary to consider a fundamentally new approach.

B. *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 method do not, which represents a significant difference.

This is illustrated in Figure 11. 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 method 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.

Addressing the issues mentioned in these subsections would be crucial for the direct application of the subspace method.

Appendix C
Another approach based on the Proposed Method

本論文では六方格子に基づいた Subspace Method を提案したが、六方格子だけのアイデアを活用した方が、実装上早いという場合がある。(Python の numpy 配列に index でアクセスするのが遅い)

その内の一つは、各頂点毎に最適化する際に、ランダムに頂点を選んで最適化していくのではなく、$|V|$ 点全てを同時に最適化する方法である。

こうすることによって、六方格子という制約をよりラフに扱うことが出来る。

六方格子への射影は、MinCostFlow などで $\mathcal{O}\left(|V|^3\right)$ で厳密解が出せる。ソートによる擬似的な射影で、$\mathcal{O}(|V|\log|V|)$ で近似解が出せる。

基本的に性能は落ちると考えているが、実装が簡略化するなどの利点や、イテレーション毎の計算コストが多少減るという利点があり、検討の余地が残されている。

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.