Initial Placement for Fruchterman–Reingold force model with Coordinate Newton Direction

Hiroki Hamaguchi Naoki Marumo Akiko Takeda

Abstract—Graph drawing is a fundamental task in information visualization, with the Fruchterman–Reingold (FR) force model being one of the most popular choices. We can interpret this visualization task as a continuous optimization problem, which can be solved using the FR algorithm, a gradient descent-like method, or the L-BFGS algorithm, a 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, we propose a new initial placement based on the stochastic coordinate descent to accelerate the optimization process. We first reformulate the problem as a discrete optimization problem using a hexagonal lattice and then iteratively move a randomly selected vertex along the coordinate Newton direction. We can use the FR or L-BFGS algorithms to obtain the final placement. We demonstrate the effectiveness of our proposed approach through experiments, highlighting the potential of coordinate descent methods for graph drawing tasks. Additionally, we suggest combining our method with other graph drawing techniques for further improvement. We hope that this work will inspire future research of coordinate descent methods not only in graph drawing but also in broader graph-related applications.

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

1 Introduction

Graph is a mathematical structure representing pairwise relationships between objects, and graph drawing is a fundamental task in information visualization. Indeed, numerous kinds of models and algorithms have been proposed [1], [2], [3], and among these, one of the most popular choices is the force-directed graph drawing.

In force-directed graph drawing, we model a graph as a system of particles with forces acting between them. By simulating the system and seeking the equilibrium of the forces, we can obtain a visualization of the graph. Among the various force models [4], [5], the Fruchterman–Reingold (FR) force model [6], [7] is the central focus of our study.

FR algorithm is the original algorithm for this force model, and it can be regarded as a variant of the gradient descent method for a energy function of the model. FR algorithm is implemented in many graph drawing libraries such as NetworkX [8], Graphviz [9], and igraph [10].

However, the FR algorithm has several issues that make it challenging to achieve high-quality visualizations for large-scale graphs. First, it suffers from high computational complexity when directly applied, $\mathcal{O}(n^2)$ per iteration, with n being the number of vertices. Secondly, the occurrence of "twist" [11] is crucial for the force model, as it can significantly impact the simulation efficiency. We refer to "twist" as unnecessary edge intersections or tangled structures, as shown in the upper half of Fig. 1. When "twist" exists, mutual interactions may weaken or diminish the forces, causing the simulation process to stagnate.

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

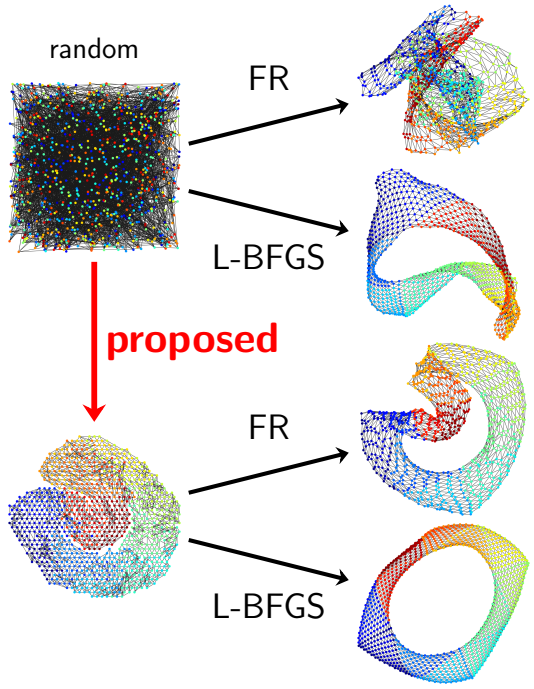


Fig. 1: Comparison of the algorithms for the `jagmesh1`.

Another strategy is to directly accelerate the simulation process with the forces, in other words, the optimization process of the energy function. This aligns with the aim of our work. Recent research has accelerated the process in various ways, such as adapting to GPU parallel architectures [16] or utilizing numerical optimization methods. L-BFGS algorithm, a family of quasi-Newton methods, is

one such approach and is reported to be effective for graph drawing [17]. However, since this method treats the problem just as a general optimization problem, adding some pre-processing based on the inherent graph structure of the problem could be effective. Indeed, Simulated Annealing (SA) is also known to be effective when used as a pre-processing step [18], since SA can deal "twist" issues and leads to a better visualization combined with the FR algorithm. However, this work only uses a circle initial placement for unweighted graphs with a simple method, leaving significant room to improve the effectiveness and extend the applicability. Refer to Sec. 2.4 for more details.

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

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

2 Preliminary

In this section, we formulate the simulation process of the force model as a continuous optimization problem and introduce the conventional approaches to this problem, namely the FR algorithm and the L-BFGS algorithm. We also briefly review the pre-processing step by Ref. [18].

2.1 Formulation of the Force Model

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

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

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

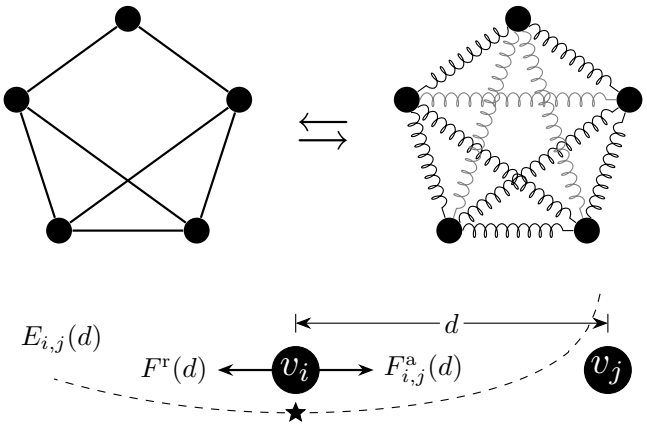


Fig. 2: (Top) The illustration of the force model. Forces act on every pair of vertices. (Bottom) Forces $F^{\mathrm{a}}_{i,j}(d)$ and $F^{\mathrm{r}}(d)$ work between v_i and v_j. The equilibrium of them is achieved at $d = k/\sqrt[3]{w_{i,j}}$, which equals k when $w_{i,j} = 1$.

Fruchterman and Reingold proposed a force model for graph drawing, based on the physical analogy of the system of particles [6]. Let $x_i \in \mathbb{R}^2$ be the position of the vertex $v_i \in V$, and $X = (x_1, \ldots, x_n) \in \mathbb{R}^{2\times n}$ be the placement of the graph. Let $\|\cdot\|$ denote the Euclidean distance in $\mathbb{R}^2$. For a parameter k and a distance d between two vertices v_i and v_j, the attraction force $F^{\mathrm{a}}_{i,j} \colon \mathbb{R}_{>0} \to \mathbb{R}$ and the repulsion force $F^{\mathrm{r}} \colon \mathbb{R}_{>0} \to \mathbb{R}$ is defined as

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

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

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

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

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

Now, the optimization problem can be formulated as the minimization of the energy function $f \colon \mathbb{R}^{2\times n} \to \mathbb{R}$, 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 stress f. In the following, we will discuss how to optimize this minimization problem, Prob. (3).

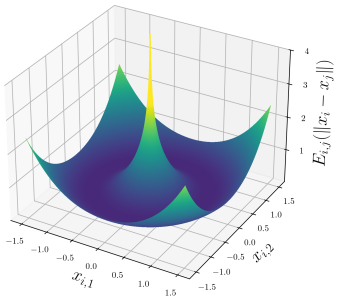


Fig. 3: Energy function $E_{i,j}(\|x_i - x_j\|)$ for $x_i = (x_{i,1}, x_{i,2})$, $x_j = (0,0)$, $w_{i,j} = 1$ and $k = 1$. Although $E_{i,j}$ is convex as a function of $d = \|x_i - x_j\|$, it is not convex as a function of x_i. As x_i approaches x_j, the energy function diverges.

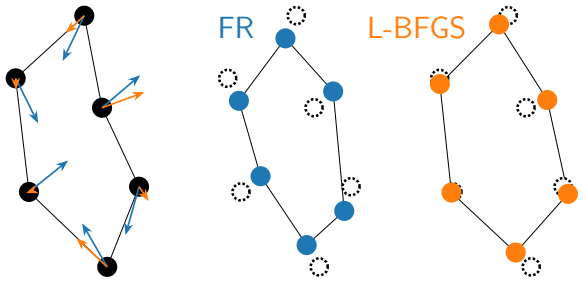


Fig. 4: Comparison of the FR algorithm and the L-BFGS algorithm. The FR algorithm moves vertices in a descent direction with a fixed step size (blue arrows). In contrast, the L-BFGS algorithm adjusts them differently since it utilizes approximated inverse Hessian (orange arrows).

2.2 Fruchterman–Reingold Algorithm

The Fruchterman–Reingold algorithm [6] is the original force-directed algorithm and the most standard approach for this force model.

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

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

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

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

The pseudo-code of the FR algorithm is shown in Algorithm 1, which can be regarded as a variant of gradient (steepest) descent method for the function f [19].

Algorithm 1: Fruchterman–Reingold algorithm

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

Alg. 1 is based on the original code [6] and implementation in NetworkX [8] with some omitted details. The initial placement X is sampled from a uniform distribution on a unit square in general. The parameter t denotes the temperature, which governs the step size along the steepest descent. As the temperature gradually decreases, the algorithm converges to a particular placement, though this placement is not necessarily the optimal solution.

2.3 L-BFGS Algorithm

Another approach to solving the optimization Prob. (3) is to use the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm [17]. Using only a few recent gradient vectors, the L-BFGS algorithm approximates the inverse Hessian of the objective function f, which is necessary to determine a descent direction [20]. We fix the maximum number of iterations to $N_{\text{iter}}^{\text{L-BFGS}}$. L-BFGS is known to be very efficient for large-scale optimization problems, and the superior performance of the L-BFGS algorithm to the FR algorithm reported in Ref.[17] also indicates this fact. Refer to Fig. 4 for a comparison to the FR algorithm.

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

2.4 Pre-Processing by Simulated Annealing

As a related work, we briefly introduce and discuss the pre-processing step proposed by Ref. [18]. Let $Q^{\text{circle}} \coloneqq \{(\cos(2\pi i/n), \sin(2\pi i/n)) \mid 1 \leq i \leq n\}$ be the points on a unit circle in $\mathbb{R}^2$. For an unweighted graph G_W, let E_2 be a set of vertex pairs with a theoretical distance equal to 2. Let $\angle(a, b)$ be the angle between two points a and b in $[-\pi, \pi]$. This study defines the problem for pre-processing as follows (we change some notations for consistency):

$$\begin{array}{ll} \underset{\pi \colon V \to Q^{\text{circle}}}{\text{minimize}} & \displaystyle\sum_{(i,j) \in E \cup E_2} |\angle(\pi(i), \pi(j))|. \\ \text{subject to} & \pi \text{ is injective.} \end{array} \tag{6}$$

The Prob. (6) is a discrete optimization problem to find the best assignment π just using angles, not the function f. Ref. [18] reports that we can obtain a faster and better visualization by setting the result of Simulated Annealing (SA) for Prob. (6) as the initial placement for the FR algorithm.

However, the following limitations remain:

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

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

3 Proposed Algorithm

In this section, we propose a new initial placement algorithm for Prob. (3). We first introduce the graph drawing by stochastic gradient descent [21] and the concept of the coordinate Newton direction, the key concept of our research. Then, we define the discrete optimization problem for initial placement and propose a new algorithm to optimize it, based on stochastic coordinate descent.

3.1 Related Work

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

3.1.1 Graph Drawing by Stochastic Gradient Descent

When we regard graph drawing as an optimization problem, Stochastic Gradient Descent (SGD) for Kamada–Kawai (KK) layout [5] is one of the most notable works [21]. In the KK layout, we regard G as a complete graph and assign the energy function $E^{\mathrm{KK}}_{i,j}$ to all edges. SGD in this context means to repeat randomly selecting an edge $\{v_i, v_j\}$ and updating x_i and x_j with the gradient of $E^{\mathrm{KK}}_{i,j}$. This algorithm is known to be effective in various optimization problems in general.

Although applying SGD to our problem Prob. (3) is straightforward, it is ineffective for this problem. This is because the force model we consider assigns exactly the same function $E_{i,j}(d) = -k^2 \log d$ to all (i,j) such that $w_{i,j} = 0$. Optimizing $E_{i,j}$ only increases the distance between vertices v_i and v_j, no matter how close they are in the optimal solution. Thus, the gradient of $E_{i,j}$ is not informative enough to find the optimal solution, and we need to develop a new optimization method for Prob. (3).

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

3.1.2 Newton Direction and Coordinate Newton Direction

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

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

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

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

We call the direction $d = -\nabla^2 f(x_0)^{-1} \nabla f(x_0)$ as the Newton direction. Although Newton direction is the optimal direction for the approximated f, it requires the computation of the inverse Hessian $\nabla^2 f(x_0)^{-1} \in \mathbb{R}^{n \times n}$, posing a high computational cost for large-scale problems. Actually, the reason why the L-BFGS algorithm approximates the inverse Hessian is to avoid this computational cost.

Still, we can leverage the concept of the Newton direction in a different manner, the coordinate Newton direction. Instead of computing the inverse Hessian $\nabla^2 f(x_0)^{-1}$ in the entire variable space $\mathbb{R}^n$, we limit the variable x to its coordinate block x_i with fewer dimensions, and compute $\nabla^2 f_i(x_i)^{-1} \nabla f_i(x_i)$ where f_i is a restricted function of f to x_i. Since the computation of the coordinate Newton direction is much cheaper than that of the Newton direction, we can repeat this procedure many times. In general, this idea is known as stochastic coordinate descent [22] or Randomized Subspace Newton (RSN) [23] in a broader context.

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

3.2 Reduction to Discrete Optimization Problem

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

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

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

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

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

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

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

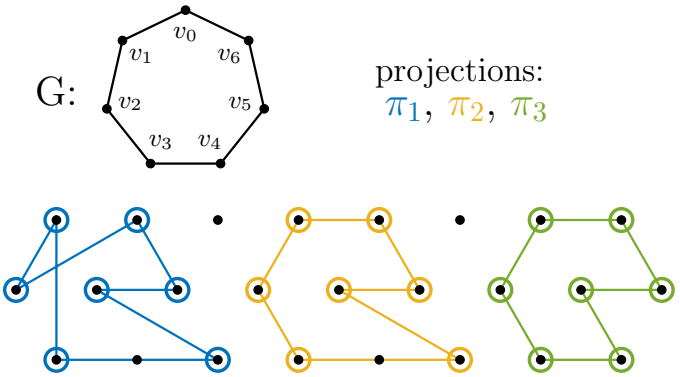


Fig. 5: Concept of Q and π. The injection π maps vertices V to a discrete point placement Q, especially a hexagonal lattice Q^{hex}. Apparently, among π_1, π_2, π_3, the right one π_3 is the best mapping for the Prob. (9).

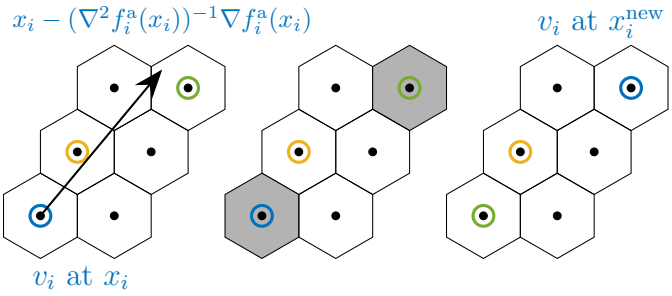


Fig. 6: Visual explanation of the one iteration of the proposed algorithm. Step1. Compute the coordinate Newton direction for a randomly selected vertex (blue). Step2. Decide x_i^{new} by rounding the direction and adding a random vector. Step3. Move the vertex and swap the vertices if there is a collision.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

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

For such a discrete point placement Q, we can consider various placements, including Q^{circle} [18]. However, in this study, we adopt a hexagonal lattice Q^{hex} [24], [25]. When minimizing the attraction energy f^{a}, it is advantageous for the points to cluster as closely as possible. In this context, the hexagonal lattice is known for its densest packing structure in space with the least distance ϵ between points and offers computational simplicity. Thus, Q^{hex} is a suitable choice for our purpose.

3.3 Newton Direction for Discrete Optimization

Next, using the coordinate Newton direction of a randomly selected vertex, as mentioned in Sec. 3.1, we optimize the discrete optimization problem, Prob. (9).

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

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

Its gradient and Hessian matrix are

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

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

The ordinary updated rule with the coordinate Newton direction is

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

However, x_i^{new} may not be in the hexagonal lattice Q^{hex}, a constraint of Prob. (9). Thus, we must round to the nearest point in Q^{hex}. Plus, we empirically found that adding a random vector to the Newton direction is effective for the optimization process, which is a similar strategy to the SA in Sec. 2.4. This randomness can help to escape from local minima and to explore the solution space more effectively. In conclusion, the updated rule for the vertex v_i is

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

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

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

3.4 Optimal Scaling

When we optimize the initial placement for the original continuous optimization problem Prob. (3), scaling the placement at first can often yield better results than directly using the unmodified one. This subsection explains how to find the optimal scaling factor that minimizes f 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 scale it as $x_i \leftarrow cx_i$ for all i. This problem is to minimize the energy function $\phi(c)$ defined by

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

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

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

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

This value can be computed in $\mathcal{O}(|E|)$ complexity. We can obtain a better initial placement for Prob. (3) using this scaling factor.

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

3.5 Pseudo Code

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

Algorithm 2: Proposed algorithm as initial placement

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

3.6 Alternative Approach

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

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

4 Numerical Experiment

In this section, we evaluate the proposed algorithm using various numerical experiments. We also confirm that the proposed algorithm extends the applicability of the pre-processing step in Ref. [18], as mentioned in Sec. 1.

4.1 Experimental Setup

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

To implement the FR and L-BFGS algorithm, we referenced NetworkX version 3.3 [8], SciPy 1.14.1 [26], and C++ L-BFGS [27], [28]. In particular, we used almost the same parameters as NetworkX's spring_layout for the FR algorithm. We also referenced the open-source code of the hexagonal grid from [24] and Graphviz version 2.43.0 [9].

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

- Random initialization (no prefix),
- The circle initialization obtained by Simulated Annealing (SA-) [18],
- The proposed initialization obtained with coordinate Newton direction (CN-).

As an algorithm to solve Prob. (3), we used

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

As parameters, we used $N^{\mathsf{CN}}_{\mathrm{iter}} = 2|V|^3/|E|$ for initial placement, and we also set the total number of iterations of Simulated Annealing (SA) to the same value. Since the amortized time complexity per iteration of Algorithm 2 is $\mathcal{O}(|E|/|V|)$, we can roughly expect that the computational time of the proposed algorithm is $\mathcal{O}\left(2|V|^2\right)$, equivalent to a few iterations of the FR or L-BFGS algorithm. All the codes are available at GitHub [29].

4.2 Comparison with Other Initializations

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

As a dataset, we used matrices from Sparse Matrix Collection [30] satisfying the condition (1) with $|V| \le 1000$, in total 124 graphs. For fairness, when we compare with SA, we converted the graph to a unweighted one. We set weights $w_{i,j}$ to 1 if $\{i, j\} \in E$; otherwise, we set it to 0.

We set $N^{\mathsf{FR}}_{\mathrm{iter}} = N^{\mathsf{L-BFGS}}_{\mathrm{iter}} = 50$, the default parameter of NetworkX [8], except for the CN algorithms compared with random initialization. We set them as 45 in this case, since it contains the pre-processing step. We can roughly expect that the computational time is equivalent to a few iterations of FR or L-BFGS algorithms, as mentioned in Sec. 4.1.

The results are shown in Fig. 7 and Fig. 8. In almost all cases, the proposed algorithm performed better than random or circle initialization. A few cases where the proposed

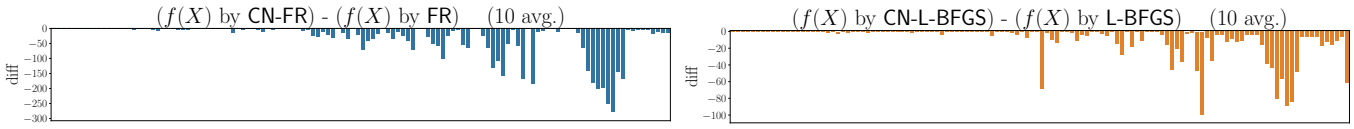


Fig. 7: Comparison of the proposed initialization (CN) with random initialization (no prefix). In almost all cases, the difference is negative, meaning the proposed algorithm performed faster and better than random initialization.

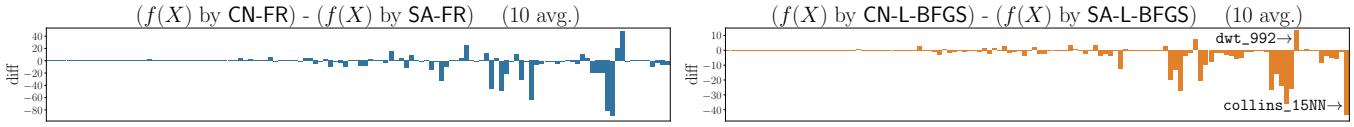


Fig. 8: Comparison of the proposed initialization (CN) with circle initialization (SA) in Ref. [18]. In most cases, the proposed algorithm performed better than the circle initialization.

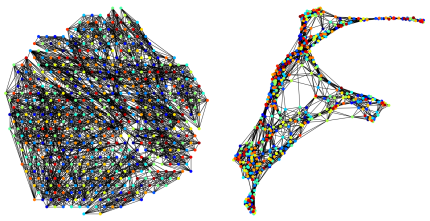


Fig. 9: An example of a case where the proposed algorithm performed worse than random initialization. Left: Initial placement by CN. Middle and Right: 50th and 200th iteration of CN-L-BFGS.

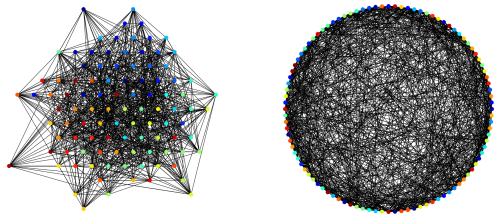


Fig. 10: An example of a case where $w_{i,j} \notin \{0, 1\}$. This result shows the proposed algorithm is extending the applicability of the pre-processing step. Left: the result of CN. Separated by colors. Right: the result of SA. No separation.

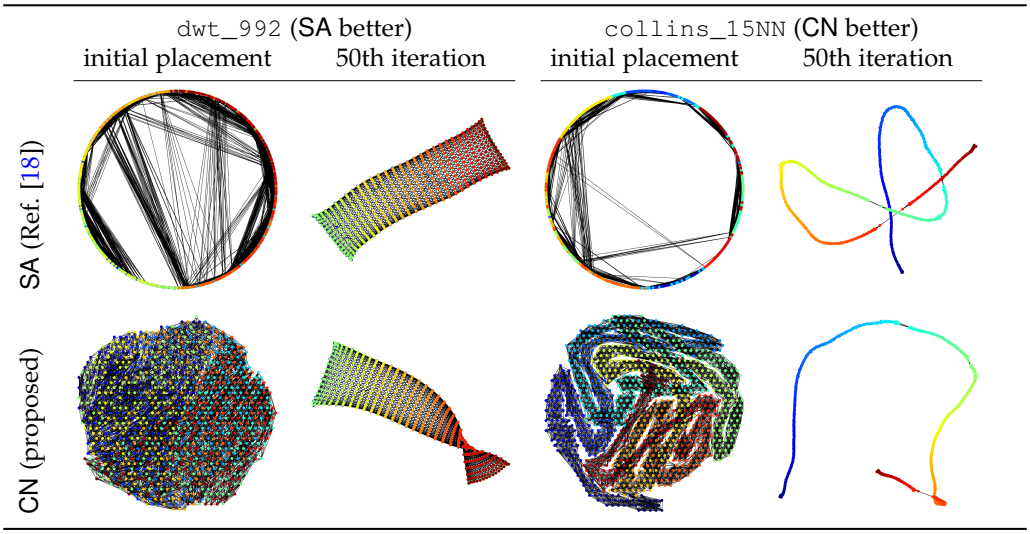
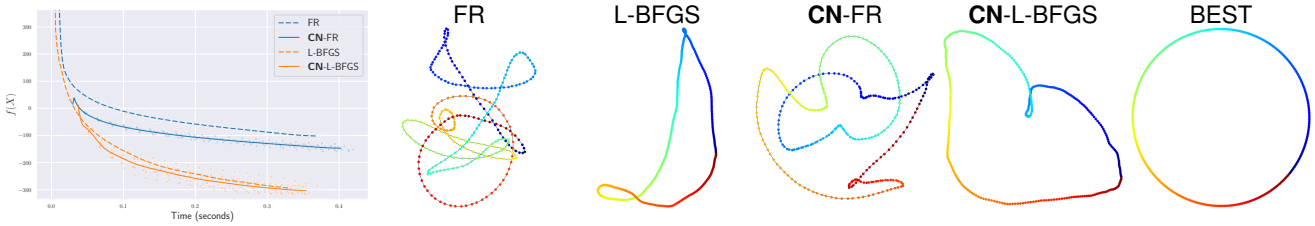


Fig. 11: Visualization results showing initial and 50th iteration placements for dwt_992 and collins_15NN.

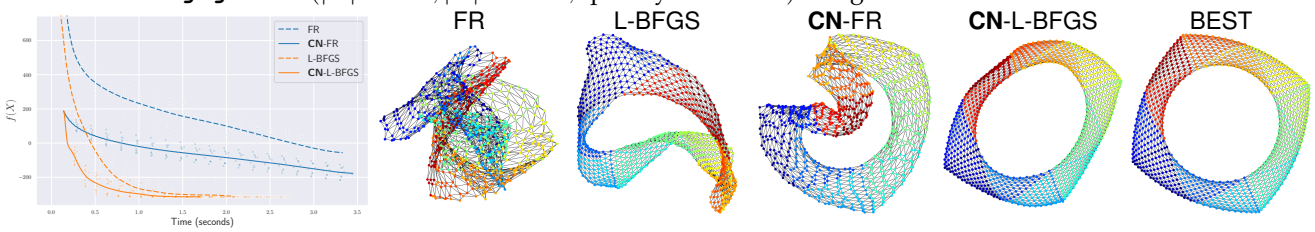
algorithm performed worse than other methods. As for random initialization, one of such cases is Spectro_10NN shown in Fig. 9. The proposed algorithm might fail to resolve the "twist" in the initial placement, shown in the figure, leading to a worse result. However, the average performance for this case was still almost the same as that of the random initialization. As for circle initialization, we showed examples in Fig. 11. The proposed algorithm outperforms in collins_15NN and underperforms in dwt_992. The colors of the vertices in the graphs presented in this section are assigned according to vertex indices using a color map that provides a gradient from blue to red.

The interpretation of the results is as follows. First, this result strongly supports the effectiveness of the proposed algorithm. It successfully untangles the "twists," leading to better outcomes with faster convergence. Even when the proposed algorithm performed worse, the difference was insignificant in almost all cases. Secondly, the choice of initial placement can lead to advantages or disadvantages depending on the optimal placement. In particular, since the optimal shape of collins_15NN is a linear shape, which differs from a circle, SA's performance can be worse than

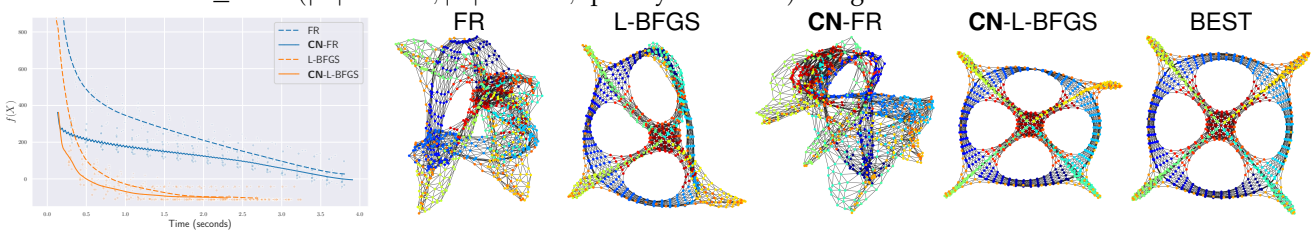
cycle300 ($|V| = 300, |E| = 300, \text{sparsity} = 0.669\%$) Figures are at 150 iterations.



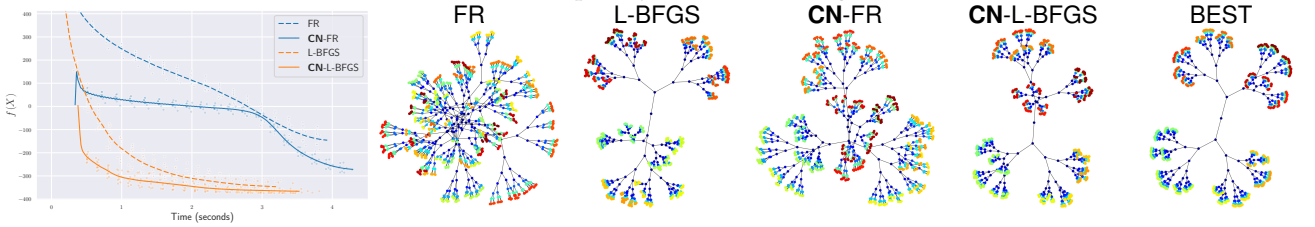
jagmesh1 ($|V| = 936, |E| = 2664, \text{sparsity} = 0.609\%$) Figures are at 50 iterations.



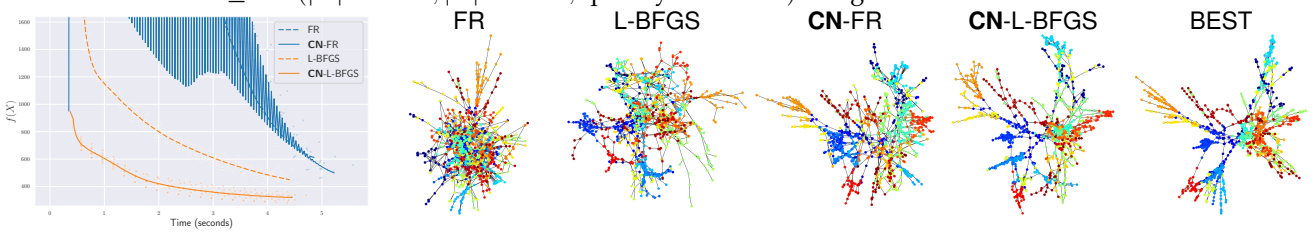
dwt_1005 ($|V| = 1005, |E| = 3808, \text{sparsity} = 0.755\%$) Figures are at 100 iterations.



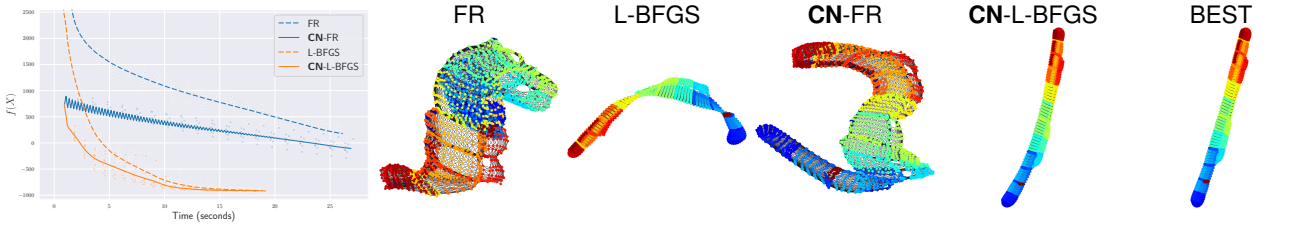
btree9 ($|V| = 1023, |E| = 1022, \text{sparsity} = 0.196\%$) Figures are at 150 iterations.



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



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



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

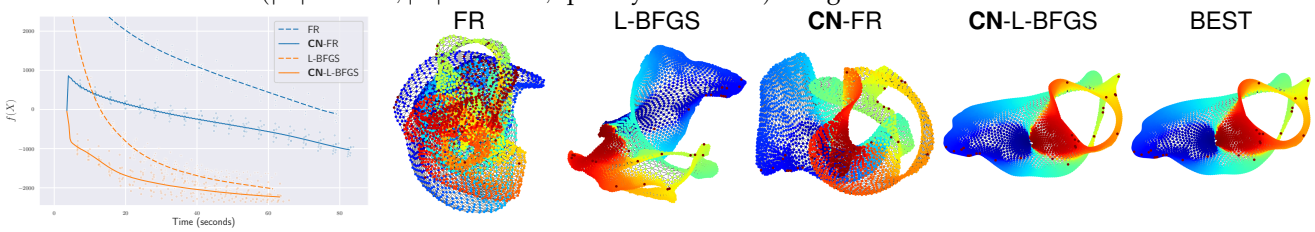


Fig. 12: Numerical experiment results for various graphs. Please refer to Sec. 4.3 for details.

the proposed algorithm. Such difference likely contributed to the advantage of the proposed algorithm. Although the proposed method uses a hexagonal lattice Q^{hex}, it can be adapted to other fixed placement Q. If the shape of the optimal solution is roughly known in advance, selecting a different Q, such as a Q^{circle}, could further enhance the performance of the proposed algorithm.

Additionally, although this is a case not considered in Ref. [18], we also conducted experiments with CN and SA for a case where the edge weight $w_{i,j}$ is not necessarily in $\{0,1\}$. We made a weighted graph having 100 vertices and 1000 edges with three groups of vertices. We generated every edge randomly, and if the two vertices were in the same group, we set the edge weight to 1.0; otherwise, we set it to 0.1. It exhibits both strong and weak connections. Fig. 10 shows the difference between the two initialization. When we ignore the edge weights and just solve Prob. (6), the graph is just an Erdos–Renyi graph, and thus SA cannot find any meaningful structure in the initial placement. On the other hand, the proposed algorithm can find the graph's structure, and we can observe that the left graph in Fig. 10 is separated by the groups, i.e., the node color. This result suggests that our proposed algorithm is effective even for weighted graphs, extending the applicability of the pre-processing step.

4.3 detailed Experiment

We also conducted a detailed experiment to investigate the behavior of the proposed algorithm in detail. Fig. 12 shows the results.

The experiment details are as follows. As parameters, we used $N_{\mathrm{iter}}^{\mathsf{FR}} = N_{\mathrm{iter}}^{\mathsf{L-BFGS}} = 200$ 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 [30], and these choices are based on the experiments conducted in Ref. [21]. Thus, although all the graphs are quite sparse so that $|E|/(|V|(V-1)/2)$ is less than 1%, this is not an arbitrary choice. The important graphs often have such a sparsity.

We first explain what Fig. 12 represents. The plots on the left illustrate the objective function values $f(X)$ on the vertical axis versus execution time on the horizontal axis, using ten trials for each algorithm. Faint crosses represent the results with random initialization, while faint circles represent the results of CN, plotted every ten iterations for each trial. The solid and dashed lines represent the average of these values across the ten trials for each algorithm. If one of the trials terminated before reaching $N_{\mathrm{iter}}^{\mathsf{FR}}$-th or $N_{\mathrm{iter}}^{\mathsf{L-BFGS}}$-th iterations, we plotted up to the minimum trial count achieved across all trials. Each trial was conducted with a different seed. The graphs on the right are at the iteration in which the most significant difference appeared among {50,100,150}-th iterations (or at the last iteration if it ended earlier), using seed 1.

The observations and implications of Fig. 12 are as follows. First, the solid plots for CN generally demonstrate superior performance compared to non-CN, validating the efficacy of the proposed method. Some exceptions of the superiority arise with FR, which exhibits oscillations in the plot, likely due to excessive step sizes leading to overshooting. Although we refrained from altering FR for fairness, adjusting the step size (or using adaptive step size) could enable the proposed method to achieve its intended performance. Still, the initial $f(X)$ of CN-FR is small enough compared to the objective function values produced by FR alone, suggesting that the proposed method is yielding a good initial placement. Additionally, the visualization results support the effectiveness of the proposed initial placement. In most cases, placements obtained with CN better represent the intended geometric arrangement showed in the "BEST" column, obtained by running at most 500 iterations of L-BFGS, than non-CN placements.

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

5 Challenges of Coordinate Newton Direction

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

5.1 Possible Approach with Coordinate Newton Direction

We first explain a possible approach using the coordinate Newton direction to optimize Prob. (3). As mentioned, our approach is based on the stochastic coordinate descent and is also resembles the randomized subspace Newton, one of the subspace methods [23], [31], [32], [33], [34]. However, our approach differs from these because we convert the problem to the discrete optimization problem.

The direct application of such method to Prob. (3) is as follows: we randomly select a vertex v_i, apply Newton's method or its regularized variant to f_i using the gradient in Eq. (5) and its Hessian:

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

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

Fig. 13: The inaccurate quadratic approximation. For the red vertex on the right graph, its coordinate Newton direction is the red arrow, which is a bad direction.

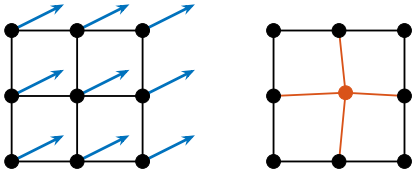


Fig. 14: The ignorance of other vertex movements. Although the blue arrows show the forces in this situation, the red vertex barely moves by the coordinate Newton direction.

5.2 Inaccuracy of Quadratic Approximation

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

We show an example of this issue in Fig. 13. Let a graph G be the one on the left, where k and all positive edge weights $w_{i,j}$ are 1. In the situation depicted on the right, the position of points are

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

The key point of this example is that the Hessian

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

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

5.3 Ignorance of Other Vertex Movements

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

Figure 14 illustrates this issue. Consider a subset of vertices that form a mesh-like structure in G, and all vertices receive forces to the blue arrow directions. In this setting, the FR and L-BFGS algorithms progress the optimization without issue. On the other hand, if we attempt to optimize only for the red vertex v_1 in the right graph, due to its fixed directly connected neighbors, v_1 barely moves. As a result, the overall optimization barely advances, in contrast to the alignment of the forces (blue arrows).

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

5.4 Rationale for Proposed Method

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

However, by converting the problem as stated in Section 3.2, we can mitigate these issues. The advantage of Prob. (9) is that not only reduction of the computational complexity from $\mathcal{O}\left(|V|^2\right)$ to $\mathcal{O}(|E|)$. It also brings the convexity of the problem, making the quadratic approximation more accurate than the original problem. Moreover, the discrete point set Q mitigates the adverse effects caused by ignoring the movement of other vertices. This is because each point is always separated by at least ϵ, making minor movements negligible and non-critical. The step size in the optimization is also assured to be more than ϵ, evading the stagnate of the optimization. These advantages are the key to preventing the issues of Sec. 5.3 and Sec. 5.2. Thus, making the problem discrete is essential to provides a high-quality initial placement.

The crucial point is that, when combined with suitable strategies, the stochastic coordinate descent or randomized subspace Newton can be effective for the original continuous problem, Prob. (3). Focusing on each vertex separately is a natural and valid approach. With further refinements and by addressing the issues discussed in Sec. 5.2 and Sec. 5.3, these methods hold the potential to efficiently provide good solutions for Prob. (3).

6 Discussion

In this section, we discuss the future directions of this research. Firstly, we discuss to combine our algorithm with some conventional techniques, such as the Multilevel approach. Secondly, we explore the applications beyond the scope of graph drawing. Finally, we conclude this paper.

6.1 Combination with Other Techniques

This paper has demonstrated the effectiveness of the proposed method on various graphs, but we can also apply it to larger-scale problems. For instance, the Scalable Force-Directed Placement (sfdp) of Graphviz [9], based on [14], employs a multilevel approach to accelerate processing for larger graphs by progressively coarsening vertices.

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

In addition, the FR force model is sometimes used not only in $\mathbb{R}^2$ but also in $\mathbb{R}^3$ [35]. Although we have to modify

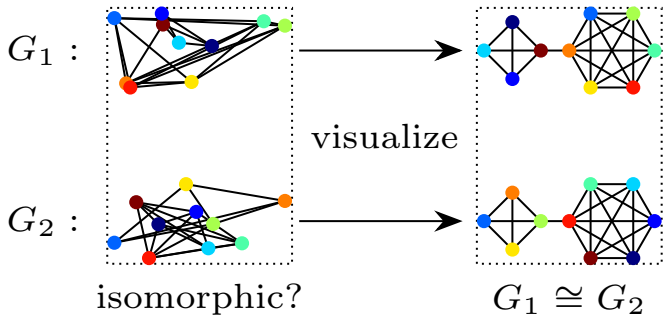


Fig. 15: Illustration of the relationship between graph drawing and graph isomorphism. If we can draw graphs G_1 and G_2 symmetrically, then it is clear that $G_1 \cong G_2$.

some parts of the proposed algorithm for $\mathbb{R}^3$, such as the hexagonal lattice, its application would be easy.

6.2 Application to Other Problems

In this subsection, we briefly discuss and explore the potential applicability of stochastic coordinate descent to a broader range of problems. Although we utilized the coordinate Newton direction only for the optimization Prob. (3), we can see that its application is not necessarily limited to the FR force model alone.

In general, the optimization Prob. (3) is more broadly treated as "objective functions arising from graphs" [22]:

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

where Ω_i is a regularization term for the i-th vertex, and λ is a regularization parameter. Our optimization problem, Prob. (3), is a special case of this problem class. The authors of [22] claim that coordinate descent, based only on coordinate gradients, is effective for solving such problems. A variant of the proposed method utilizing the coordinate Newton direction can also be effective for such problems.

For instance, we suspect that the graph isomorphism problem is one of the candidates for the application. The graph isomorphism problem is a well-known combinatorial optimization problem to determine whether two graphs G_1, G_2 are isomorphic, i.e., $G_1 \cong G_2$. The graph isomorphism problem is closely related to the graph drawing. Drawing a graph in a way that reveals its symmetry is at least as difficult as the graph isomorphism problem [4]. Indeed, if we can draw two graphs G_1 and G_2 in the same way, it becomes evident that $G_1 \cong G_2$. See Fig. 15 for reference. When we relax it to a continuous optimization problem on Riemannian manifolds [36], [36], we might be able to apply the stochastic coordinate descent or coordinate Newton direction to this problem as well. Investigating the the variant of our proposed algorithm to these problems constitutes one of the future research directions.

6.3 Conclusion

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

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

7 Acknowledgment

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

The author also thanks the developers of NetworkX and Graphviz. Their excellent work 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] P. Eades, "A heuristic for graph drawing," *Congressus numerantium*, vol. 42, no. 11, pp. 149–160, 1984.

[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] 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

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

[8] 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.

[9] 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.

[10] 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

[11] 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

[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] 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

[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] J. X. Zheng, S. Pawar, and D. F. M. Goodman, "Graph Drawing by Stochastic Gradient Descent," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 9, pp. 2738–2748, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8419285

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

[23] 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

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

[25] 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

[26] 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.

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

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

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

[30] 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.

[31] 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

[32] 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

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

[34] 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

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

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

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

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