Fruchterman–Reingold Algorithm with Random Subspace Newton

Hiroki Hamaguchi

***Abstract*—The abstract goes here. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.**

***Index Terms*—Graph drawing, Fruchterman–Reingold algorithm, Random Subspace method.**

I. Introduction

GRAPH is a mathematical structure representing pairwise relationships between objects, and graph drawing is one of the most fundamental tasks in data science. Indeed, Numerous general 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, the graph is modeled as a physical system of particles. These include methods such as the Kamada–Kawai (KK) layout [5] using shortest path distances for a cost function, and the Fruchterman–Reingold (FR) layout [6], which is the primary focus of this paper.

The FR algorithm is one of the most widely used force-directed algorithms. It is also implemented in many modern graph drawing libraries such as NetworkX [7], Graphviz [8], and igraph [9]. The FR layout is based on a physical model of a system of particles and springs, and FR algorithm seeks the equilibrium of the forces between nodes.

However, since both the KK layout and the FR layout requires to compute the forces or energies between all pairs of vertices, they have a high computational cost of $\mathcal{O}(n^2)$ per iteration, where n is the number of vertices in the graph. To address this kind of computational burden, several methods have been proposed.

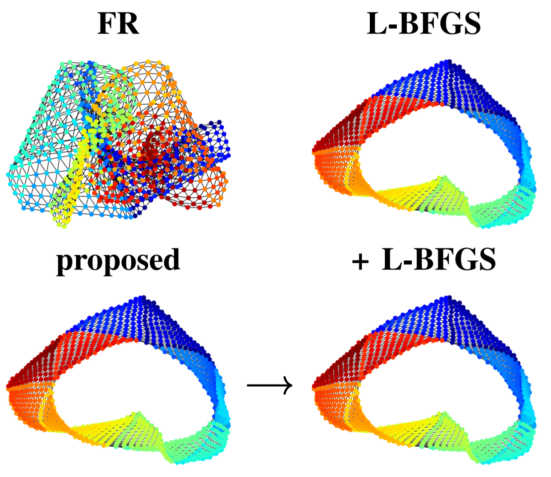

TABLE I: Comparison of the FR algorithm (left, NetworkX) and L-BFGS-B (right, my implementation) on the `jagmesh1` dataset.

One strategy is to approximate the n-body simulation using hierarchical methods such as the fast multipole method [10], the Barnes–Hut approximation [11], multilevel approaches [12], or stress majorization [13].

Another approach is to accelerate the optimization algorithm directly, which aligns with the spirit of our work. Recent researches have accelerated the algorithms for FR layout and KK layout through various methods, including GPU parallel architectures [14], numerical optimization techniques such as L-BFGS [15], and Stochastic Gradient Descent (SGD) [16].

Based on such advances, in this paper, we investigate the ability of another algorithm: Random Subspace Newton (RSN). The RSN method is a variant of the Newton method that uses a random subspace to approximate the Hessian matrix. The RSN method and its variant have been proposed in the context of optimization problems [17], [18], [19], [20].

The rest of the paper is organized as follows. In Section II, we define the optimization problem for the FR algorithm. In Section III, we introduce the Random Subspace method and report a its problem when directly applied to the FR algorithm. In Section IV, we propose a new algorithm that combines the FR algorithm with the

Random Subspace method. In Section V, we present the experimental results. Finally, we conclude and discuss future work in Section VI.

II. Preliminary

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

A. Fruchterman–Reingold layout

Let us define $\mathbb{R}_{>0} := \{x \in \mathbb{R} \mid x > 0\}$, $\mathbb{R}_{\geq 0} := \{x \in \mathbb{R} \mid x \geq 0\}$, and let $W = (w_{i,j}) \in \mathbb{R}_{\geq 0}^{n \times n}$ be an adjacency matrix of a undirected connected graph $G = (V, E)$. $V = \{v_i \mid 1 \leq i \leq n\}$ 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).

The FR algorithm in NetworkX, for example, can handle directed unconnected graphs with $w_{i,j} < 0$, but we will not consider such cases here. For directed graphs, slight modifications of algorithms or converting them to undirected graphs may be effective. For unconnected graphs, algorithms can be applied to each connected component independently. When some weights $w_{i,j}$ are negative, the optimization problem may become unbounded, but with $w_{i,j} > 0$, the problem is always bounded and solvable.

Each vertex $v_i \in V$ is assigned to a point on a plane $x_i \in \mathbb{R}^2$ and we define $x = (x_1, \ldots, x_n) \in \mathbb{R}^{2\times n}$ as the configuration of the graph. For an parameter k and a distance $d_{i,j} := \|x_i - x_j\|_2$ between two vertices v_i and v_j, Fruchterman and Reingold [6] defined the power of attraction $F^a_{i,j} : \mathbb{R}_{>0} \to \mathbb{R}$ and the power of repulsion $F^r : \mathbb{R}_{>0} \to \mathbb{R}$ as

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

We refer this Force-Directed layout as the Fruchterman–Reingold (FR) layout. The energy for these powers $E_{i,j}(d)$ can be defined as

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

As a remark, the energy function $E_{i,j}$ is convex for d and minimized when $d^* = k/\sqrt[3]{w_{i,j}}$, but it is not Lipschitz continuous, and is not convex for x_i (?correct expression?). Refer to Figure 2.

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:

$$\underset{x \in \mathbb{R}^{2\times n}}{\text{minimize}} \quad f(x) := \sum_{i<j} E_{i,j}(d_{i,j}) \tag{1}$$

B. Fruchterman–Reingold algorithm

The most standard approach to solve the optimization problem (1) is the Fruchterman–Reingold algorithm [6], the original force-directed algorithm for graph drawing. Let us consider this algorithm in the context of the optimization problem (1). As pointed out in [21], the FR algorithm can be regarded as a gradient descent method for the energy function f with cooling schedule.

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

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

The gradient and Hessian (for later sections) of f_i are

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

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

The pseudo code of the FR algorithm is as follows:

Algorithm 1: Fruchterman–Reingold algorithm

```
Input: Graph $G = (V, E)$ with its weights $W$
Output: Point configuration $x \in \mathbb{R}^{2\times n}$
define parameters $k, t, dt$, iterations;
$x_i \leftarrow$ random for all $v_i \in V$;
for $j \leftarrow 0$ to iterations do
    compute gradient $\nabla f_i(x_i)$ for all $v_i \in V$;
    $x_i \leftarrow x_i + t\frac{\nabla f_i(x_i)}{\|\nabla f_i(x_i)\|_2}$ for all $v_i \in V$;
    $t \leftarrow t - dt$;
    if convergence condition then
        break;
return pos
```

Algorithm 1 is based on the original implementation [6] and implementation in NetworkX [7]. The t in the parameter represents the temperature, which is used to control the step size of the gradient descent. This temperature assures the convergence to a certain configuration, which might not be a local minimum of the energy function. There are some ways to define the parameters or cooling schedule, such as linear cooling, exponential cooling, or adaptive cooling.

C. L-BFGS algorithm

Another approach to solve the optimization problem (1) is to use the L-BFGS algorithm [15], which is a quasi-Newton method that approximates the Hessian

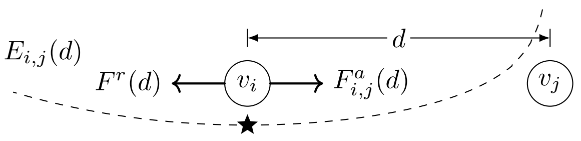

Fig. 1: Fruchterman–Reingold model. The equilibrium of the attractive force $F^a_{i,j}(d)$ and the repulsive force $F^r(d)$ is the optimal distance $d = k/\sqrt[3]{w_{i,j}}$.

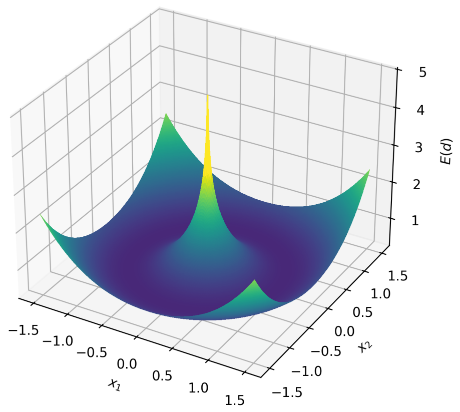

Fig. 2: Energy function $E_{i,j}(d)$ for $w_{i,j} = 1$ and $k = 1$. Although $E_{i,j}$ is convex for d, but not for x_i and x_j.

matrix using the gradient information. The L-BFGS algorithm is widely used in optimization problems, and there are many implementations available, such as SciPy [**?**] and C++ L-BFGS [**?**].

For the optimization problem (1), the L-BFGS algorithm can be applied via flattening the configuration x to a vector $x \in \mathbb{R}^{2n}$.

III. Random Subspace Newton

In this section, we introduce the Random Subspace Newton (RSN) method and report a its problem when directly applied to the FR algorithm.

A. *What is Random Subspace Newton*

First, we introduce the RSN method. RSN is a variant of the Newton's method. For N variables problem, the Newton's method requires to compute the Hessian matrix of size $N \times N$ at each iteration, which poses a high computational cost for large-scale problems. One way to address this issue is to employ quasi-Newton methods, such as the L-BFGS method, which approximates the Hessian directly, and has also been applied in graph drawing [15]. In contrast, RSN focuses on a subspace of dimension S randomly selected from the solution space and utilizes the exact Hessian matrix of size $S \times S$ defined on this subspace. Since $S \ll N$, the computational cost per iteration is significantly reduced.

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 [18].

B. *Random Subspace Newton for FR Algorithm*

In this way, the problem exhibits a natural affinity with the RSN method, as it inherently defines a subspace of the solution space for the FR layout.

Moreover, the effectiveness of Stochastic Gradient Descent (SGD) for KK layout is well-documented in [16]. In contrast to the KK layout, where each function is individually determined by $\frac{k_{i,j}}{2}(d_{i,j} - l_{i,j})^2$ based on the actual distance between vertices $d_{i,j}$ and their optimal distance $l_{i,j}$, the FR layout assigns the same value, $-k^2 \log d_{i,j}$, to all pairs of points without direct edges, making SGD relatively less effective for the FR layout.

However, the report in the KK layout, which highlights that optimization focusing on each edge yields favorable results, suggests that optimization focusing on each vertex, such as methods utilizing RSN, may also be effective in the FR layout.

In this situation, our research aims to investigate the effectiveness of the RSN method for the FR layout.

C. *problem of RSN for FR Algorithm*

However, unfortunately, the RSN method is not necessarily effective for the FR layout. As shown in Figure 3, although the energy function f_i with respect to the position x_i of each vertex is convex in the FR layout, the overall energy function f with respect to x_i is not convex.

IV. Algorithm

V. Experiment

We used dataset from [22] and MatrixMarket [23].

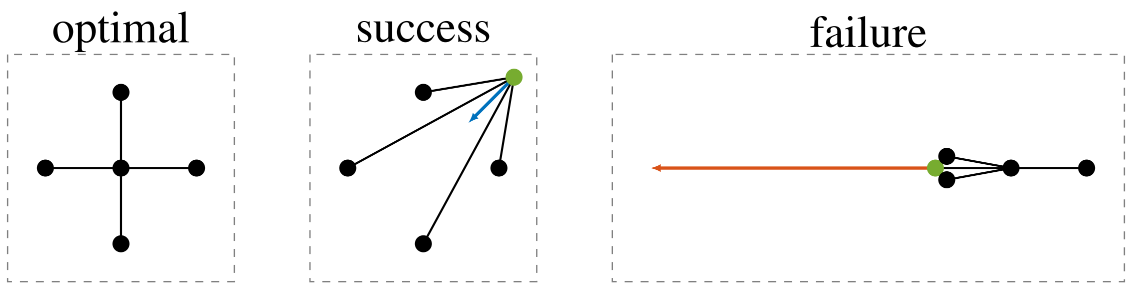

Fig. 3: Visualization of the problem of the subspace method. Let a graph as shown in the left (optimal), where k and all positive edge weights $w_{i,j}$ are set to 1. For the situation in the middle (success), the RSN works effectively. However, in the situation depicted on the right (failure), where the points are set as $x_0 = (0, 0), x_1 = (-1, 0), x_2 = (-0.85, 0.155), x_3 = (-0.85, -0.155), x_4 = (1, 0)$, the Hessian for the subspace of x_1 is approximately $\begin{pmatrix} 1.841 & 0 \\ 0 & 1.159 \end{pmatrix}$. This Hessian, while positive definite and not ill-conditioned, leads to a Newton direction that clearly deviates significantly from the global optimal solution.

https://reference.wolfram.com/language/tutorial/GraphDrawingIntroduction.html

VI. Discussion

In this paper, we investigated the effectiveness of the Random Subspace Newton method for the Fruchterman–Reingold algorithm.

A. Combination with Other Methods

B. Application to Other Problems

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.

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.

[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, May 1995.

[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, Feb. 1981.

[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. New York, NY, USA: Association for Computing Machinery, Jul. 2016, pp. 997–1000.

[5] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, no. 1, pp. 7–15, Apr. 1989.

[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.

[7] A. Hagberg, P. J. Swart, and D. A. Schult, "Exploring network structure, dynamics, and function using NetworkX," Jan. 2008.

[8] 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. Berlin, Heidelberg: Springer, 2002, pp. 483–484.

[9] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006.

[10] L. Greengard and V. Rokhlin, "A fast algorithm for particle simulations," *Journal of Computational Physics*, vol. 73, no. 2, pp. 325–348, Dec. 1987.

[11] J. Barnes and P. Hut, "A hierarchical O(N log N) force-calculation algorithm," *Nature*, vol. 324, no. 6096, pp. 446–449, Dec. 1986.

[12] Y. Hu, "Efficient, high-quality force-directed graph drawing," *The Mathematica journal*, vol. 10, pp. 37–71, 2006.

[13] 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. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, vol. 3383, pp. 239–250.

[14] 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, Feb. 2016.

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

[16] 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.

[17] 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.

[18] T. Fuji, P.-L. Poirion, and A. Takeda, "Randomized subspace regularized Newton method for unconstrained non-convex optimization," Sep. 2022.

[19] C. Cartis, J. Fowkes, and Z. Shao, "Randomised subspace methods for non-convex optimization, with applications to nonlinear least-squares," Nov. 2022.

[20] R. Higuchi, P.-L. Poirion, and A. Takeda, "Fast Convergence to Second-Order Stationary Point through Random Subspace Optimization," Jun. 2024.

[21] D. Tunkelang, "A numerical optimization approach to general

graph drawing," Ph.D. dissertation, Carnegie Mellon University, 1999.

[22] 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.

[23] R. F. Boisvert, R. Pozo, K. Remington, R. F. Barrett, and J. J. Dongarra, "Matrix Market: A web resource for test matrix collections," in *Quality of Numerical Software: Assessment and Enhancement*, R. F. Boisvert, Ed. Boston, MA: Springer US, 1997, pp. 125–137.

APPENDIX A
OPTIMAL SCALING

When we optimize a placement for FR-layout with an initial placement obtained, for instance through KK-layout, scaling the initial placement at first can often yield better results than directly using the unmodified initial placement. In this section, we address the problem of finding the optimal scaling factor that minimizes the energy function for a given configuration.

A. *Optimal Scaling Algorithm*

Formulating the optimization through scaling, the task reduces to selecting an appropriate scaling factor $x \in \mathbb{R}_{>0}$ that minimizes the following energy function:

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

The function $\phi(x)$ is convex, and we can find the optimal scaling factor x by using Newton's method.

This algorithm achieves sufficient convergence within a few iterations, and when we pre-compute the coefficients of $\phi(x)$ with $w_{i,j} > 0$, the time complexity is just $\mathcal{O}(|E|)$.

APPENDIX B
ANOTHER APPROACH BASED ON THE HEXAGONAL LATTICE

todo