JAIST Repository
https://dspace.jaist.ac.jp/

Title	バンド幅問題の効率のよいアルゴリズムの開発に関する研究
Author(s)	中西，朗裕
Citation	
Issue Date	2010-03
Type	Thesis or Dissertation
Text version	author
URL	http://hdl.handle.net/10119/8965
Rights	
Description	Supervisor:上原隆平，情報科学研究科，修士

Japan Advanced Institute of Science and Technology

Efficient Algorithm on Bandwidth Problem

Akihiro Nakanishi (0810044)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 9, 2010

Keywords: linear time algorithm, bandwidth problem, bipartite permutation graphs.

A layout of a graph $G = (V, E)$ is a bijection σ between the vertices in V and the set $\{1, 2, \ldots, |V|\}$. Bandwidth $bw(\sigma)$ of layout σ is $\max\{|\sigma(u) - \sigma(v)| \mid \{u, v\} \in E\}$. Bandwidth $bw(G)$ of the graph G is $\min_\sigma\{bw(\sigma)\}$. A layout whose bandwidth is equal to $bw(G)$ is called an optimal layout. Intuitively, computing $bw(G)$ is to find a linear ordering of vertices with the minimum maximum distance between two vertices that are adjacent in G.

Bandwidth problem is an optimization problem to compute $bw(G)$ for given G. On the other hand, k-bandwidth problem is a decision problem which asks if G has a bandwidth at most k for given G and k.

The bandwidth problem has wide applications including sparse matrix computations and molecular biology. For instance, assume that symmetric sparse matrix S whose all diagonal elements are zero is given. In this case, there is a graph G whose adjacency matrix is obtained from S by replacing all the nonzero elements with 1. Let σ be an optimal layout of G. The symmetric matrix obtained from S by reordering its rows and columns according to the ordering of σ has minimum bandwidth among all the symmetric matrices obtained from S by reordering its rows and columns. Therefore, solving the bandwidth problem on graph G, we expect that we can execute calculations such as multiplication of matrices quickly.

Copyright © 2010 by Akihiro Nakanishi

The bandwidth problem is one of the NP-complete problems. The problem is NP-complete even if the input is a tree. Cygan and Pilipczuk developed an $O(n^5)$ time exact algorithm for the bandwidth problem on general graphs in 2008, where n is the number of the vertices in the input graph. If we restrict graph classes, there are some polynomial time algorithms for the bandwidth problem and the k-bandwidth problem. There are liner time algorithms for the bandwidth problem on threshold graphs and chain graphs. There is an $O(n \log n)$ time algorithm for the k-bandwidth problem on interval graphs. There is an $O(n^2)$ time algorithm for the k-bandwidth problem on bipartite permutation graphs. In this paper, we propose a liner time algorithm for the k-bandwidth problem on bipartite permutation graphs, which improves the known best time complexity $O(n^2)$ to optimal.

We can decompose a bipartite permutation graph to a sequence of chain graphs $G_1 = (V_1, V_2, E_1), G_2 = (V_2, V_3, E_2), \ldots, G_m = (V_{m-1}, V_m, E_m)$. An existing algorithm for the k-bandwidth problem on bipartite permutation graphs uses this property.

Given a bipartite permutation graph $G = (X, Y, E)$, the existing algorithm first computes the $V_0, V_1, V_2, \ldots, V_m$ of the vertices $X \cup Y$, and obtains a sequence of chain graphs $G_1 = (V_1, V_2, E_1), G_2 = (V_2, V_3, E_2), \ldots, G_{m-1} = (V_{m-1}, V_m, E_{m-1})$. Then, it computes a layout σ_i whose bandwidth is less than or equal to k for each G_i. It arranges each $\sigma_i (1 \leq i \leq m)$ from left to right. Even if every bandwidth of G_i is less than k, distance in the layout between two vertices in G_i and G_{i-1} may exceed k. Thus, it is necessary to permute the arrangement of vertices. If we do this permuting of the arrangement of vertices in a naive way, since the number of permutations of the arrangement of vertices is exponentially large, the time complexity becomes exponential. Existing algorithm repeatedly repairs $\sigma_1, \ldots, \sigma_i$ for $i = 1, 2, \ldots, m$ so that a bandwidth of $G_1 \cup \ldots \cup G_i$ is less than k. Using a table of size proportional to the number of vertices in G_{i-1}, an existing algorithm uses $O(n^2)$ time in repairing $\sigma_1, \ldots, \sigma_i$. This is a good improvement. But there is still room for an improvement.

In this paper, we show that there are at most constant places a vertex is replaced. Therefore, using an appropriate data structure, we can find vertices that must be replaced and we can replace vertices quickly. Hence, we can improve an existing algorithm to liner time.