Cholesky factorization of matrices in parallel and ranking of graphs

Dariusz Dereniowski and Marek Kubale

Foundations of Informatics Department,
Gdańsk University of Technology, Poland,
kubale@eti.pg.gda.pl

Abstract. The vertex ranking problem has applications in the parallel Cholesky factorization of matrices. We describe the connection between this model of graph coloring and the matrix factorization. We also present a polynomial time algorithm for finding edge ranking of complete bipartite graphs. We use it to design an $O(m^{2+d})$ algorithm for edge ranking of graphs obtained by removing $O(\log m)$ edges from a complete bipartite graph, where d is a fixed number. Then we extend our results to complete k-partite graphs for any fixed $k > 2$. In this way we give a new class of matrix factorization instances that can be optimally solved in polynomial time.

1 Introduction

A k-ranking of the vertices of $G = (V, E)$ where $|V| = n, |E| = m$ is a labeling of its vertices with integers $1, \ldots, k$ such that every path between vertices with the same color contains a vertex with a greater color. If k is the smallest integer such that G has k-ranking then this number is denoted by $\chi_r(G)$. Edge ranking of graph G is a labeling of edges of G such that every path between two edges with color i contains an edge with color $j > i$. $\chi'_r(G)$ is used to denote the minimum k such that G has an edge ranking with k colors.

Vertex ranking and edge ranking problems are interesting because of their potential applications. Rankings are important in computing Cholesky factorizations of matrices in parallel [8] and in VLSI-layout [7]. Edge ranking of trees has applications in modeling the parallel assembly of a product from its components [3]. Also, vertex ranking problem is equivalent to the problem of finding a minimum height elimination tree of a graph.

Pothen [10] proved that vertex ranking problem is NP-hard. It remains NP-hard even if restricted to bipartite and cobipartite graphs [1]. On the other hand, efficient vertex ranking algorithms for some classes of graphs are known. An $O(n^3\chi_r(G)^{3d-3})$ algorithm for vertex ranking of d-trapezoid graphs has been presented in [2]. It has running time $O(n^3)$ for interval graphs and $O(n^3\chi_r(G)^3)$ for permutation graphs. In the same paper an $O(n^3)$ algorithm for vertex ranking of circular-arc graphs has been given. There exists a linear time algorithm for trees [11] and the problem is solvable in polynomial time for graphs with treewidth at most k, where k is a fixed integer [1].

Lam and Yue showed in [5] that the general problem of finding an optimal edge ranking of graphs is NP-hard. In [1] formula for computing $\chi'_r(K_n)$ for a complete graph K_n has been presented. Lam and Yue have described a linear time algorithm for edge ranking of trees [6].

In the next section we describe the connection between the vertex ranking problem and the problem of finding the minimum height elimination tree of a graph. We list some facts to show that the vertex ranking problem can be used in the parallel Cholesky factorization of matrices. Section 3 gives polynomial time algorithms for the edge ranking problem if restricted to some special classes of bipartite graphs. In particular, an $O(m^2)$ algorithm for complete bipartite graphs is described and an $O(m^{2+d})$ algorithm for graphs obtained by removing $O(\log m)$ edges from a complete bipartite graph, where d is a nonnegative constant number. We prove that these results can be generalized to complete k-partite graphs, where $k > 2$ is a fixed integer. In this way we give a new class of dense graphs, namely line graphs of complete and nearly complete k-partite graphs, that can be labeled in polynomial time.

2 Cholesky factorization of matrices in parallel

Let A be an $n \times n$ symmetric positive definite matrix. Consider a linear system of the form

$$Ax = b. \tag{1}$$

There exists a lower triangular matrix L such that

$$A = LL^T.$$

We obtain vector x by solving the following two linear systems:

$$Ly = b,$$

$$L^T x = y.$$

Note that if P is a permutation matrix then we can rewrite the linear system (1) to the form

$$PAP^T(Px) = Pb,$$

thus the matrix A can be reordered before the factorization is performed. Let us define graph $G(A)$ as follows:

$$V(G(A)) = \{1, \ldots, n\},$$

$$E(G(A)) = \{ij | A_{ij} \neq 0\}.$$

Let $p_1, \ldots, p_n$ be a permutation of the vertices of $G(A)$. To create the graph $F(A)$ repeat the following step for each $i = 1, \ldots, n$: add edges to G, such that neighbors of p_i in the set $\{p_{i+1}, \ldots, p_n\}$ form a complete subgraph.

Definition 1. *Vertex p_i is the parent of a vertex p_j in the elimination tree if and only if $i = \min\{k | k > j$ and $p_k p_j \in E(F(G))\}$. The height of an elimination tree is the length of the longest path from the root to a leaf and is denoted by $h(T)$. Symbol $h(G)$ is used to denote the height of the elimination tree whose height is minimum.*

An elimination tree describes the dependencies between the columns of the lower triangular matrix during the Cholesky factorization, i.e if a vertex a is a descendant of a vertex b then numeric values in column b depend on values in column a, thus column a has to be computed before b [8].

The vertex ranking problem is closely related to the the problem of finding the elimination tree of minimum height. If T is an elimination tree of G then the vertex ranking of G can be obtained by assigning color i to the vertices in the i-th level in T, where level $h(T)+1$ is the root. Let z be the root of a subtree of T. If $c(x) = c(y)$ where $x, y \in T[z]$ and no descendant of z is a common ancestor of x and y then x and y are not adjacent in graph F because otherwise (assume that x is ordered before y) y or some node ordered between x and y should be the parent of x. So if $S = \{z_0 = z, \ldots, z_p = \text{root}(T)\}$ is the path from z to the root of T then S is a vertex separator, such that x and y belong to different components of the graph $G - S$. This means that each connected component of induced subgraph $G[\{v | c(v) \leq i\}]$ contains at most one vertex v such that $c(v) = i$, thus coloring c is a valid vertex ranking of G. Now let us assume that c is k-ranking of G. We can obtain an elimination tree whose height is equal to $k-1$ by choosing the following ordering of the vertices of G:

$$c^{-1}(1), \ldots, c^{-1}(k), \tag{2}$$

where $c^{-1}(i)$ is the set of vertices of G with color i under c. In this tree a node with color i is not a descendant of a node with color $j < i$. Suppose that $c(v) = c(u)$ and v is the parent of u. In this case u and v are adjacent in F which means that there exist a path between u and v in G containing vertices labeled with colors $\{1, \ldots, c(u)\}$, which is impossible because c is a proper vertex ranking of G. Thus the vertices with the same color are unrelated in T. This means that the elimination tree obtained by the permutation (2) has height equal to $k-1$.

If c is a k-ranking of G then we can compute the lower triangular matrix in k steps using $\max\{|c^{-1}(1)|, \ldots, |c^{-1}(k)|\}$ processors. As an example consider the following matrix.

$$A = \begin{bmatrix} a & * & * & & & * \\ * & b & * & & & \\ * & * & c & * & & \\ & & * & d & * & * \\ & & & * & e & * \\ * & & & * & * & f \end{bmatrix}$$

The symbol * was used to denote its nonzero elements. The adjacency graph G for matrix A is shown in Fig. 1(a). Fig. 1(b) gives an optimal vertex ranking of G with four colors, which means that the factorization can be performed in four steps when two processors are used for the computation.

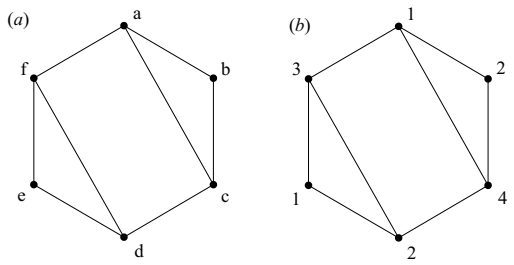

Fig. 1. (a) the adjacency graph G for matrix A; (b) optimal ranking of G.

Fig. 2 presents the elimination tree corresponding to the vertex ranking in Fig. 1(b), i.e. the elimination ordering was created on the basis of (2).

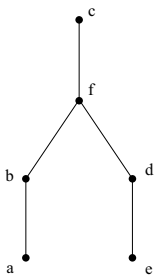

Fig. 2. An elimination tree of the graph in Fig. 1(a).

3 Complete bipartite graphs

In this chapter we consider some classes of graphs for which the vertex ranking problem is polynomially solvable, namely line graphs of complete k-partite graphs and line graphs of almost complete k-partite graphs. For convenience, we will consider the edge ranking problem of graphs instead of the vertex ranking problem of the line graphs.

Let $K_{a,b}$ be a complete bipartite graph. We denote the vertices in $V_1(K_{a,b})$ by $v_1^a, \ldots, v_a^a$ and in $V_2(K_{a,b})$ by $v_1^b, \ldots, v_b^b$. $S_{a_1,b_1}^{a,b}$ where $a_1 \leq a, b_1 \leq b$ denotes the set of edges joining vertices $v_1^a, \ldots, v_{a_1}^a$ to $v_{b_1+1}^b, \ldots, v_b^b$ and $v_{a_1+1}^a, \ldots, v_a^a$ to $v_1^b, \ldots, v_{b_1}^b$.

Lemma 1 *Let c be an edge ranking of $K_{a,b}$ using k colors. Then there exists an edge ranking c' using the same k colors, and positive integers $a_1 < a$ and $b_1 < b$ such that colors $k - |S_{a_1,b_1}^{a,b}| + 1, \ldots, k$ are assigned to the edges in $S_{a_1,b_1}^{a,b}$.*

Proof: The edges with unique labels under c form an edge-separator S in $K_{a,b}$. Consider the graph $G = (V(K_{a,b}), E(K_{a,b}) - S^{a,b}_{a_1,b_1})$, which is disconnected. Let G' denote a connected subgraph of G containing a_1 vertices from set $\{v^a_1, \ldots, v^a_a\}$ and b_1 vertices from set $\{v^b_1, \ldots, v^b_b\}$. Since G is disconnected, $G' \neq G$. Edges between vertices in $V(G')$ and $V(G) - V(G')$ get unique colors under c, so we can shuffle the labels on the edges of S so that the edges joining graphs G' and $G - G'$ receive the biggest labels. This results in an edge ranking c'. The set of edges between $V(G')$ and $V(G) - V(G')$ is equal to $S^{a,b}_{a_1,b_1}$ and $|c'(K_{a,b})| = |c(K_{a,b})|$. This completes the proof. □

Theorem 1 *There exists an $O(m^2)$ algorithm for finding an edge ranking of complete bipartite graphs with m edges.*

Proof: The procedure described below computes the edge ranking number for a complete bipartite graph $K_{a,b}$.

```
begin
  for j := 1 to b do begin
    Opt[1, j] := j;
  for i := 2 to a do
    for j := i to b do begin
      Opt[i, j] := +∞;
      for k := 1 to i − 1 do
        for l := 1 to ⌊j/2⌋ do begin
          r1 := Opt[min{k, l}, max{k, l}];
          r2 := Opt[min{i − k, j − l}, max{i − k, j − l}];
          r := k(j − l) + (i − k)l + max{r1, r2};
          if r < Opt[i, j] then
            Opt[i, j] := r;
        end
    end
  return Opt[a, b];
end
```

Array Opt contains edge ranking numbers of subgraphs $K_{i,j}$, $i \leq a$, $j \leq b$, so we have to compute the edge ranking number only once for each graph $K_{i,j}$. Lemma 1 implies that to compute $\chi'_r(K_{i,j})$, $i > j$ we have to check the following separators: $S^{i,j}_{k,l}$, $k = 1, \ldots, i-1$, $l = 1, \ldots, \lfloor j \rfloor$. In order to compute edge ranking of $K_{a,b}$ we have to use another array to store the values of k, l for each i, j such that separator $S^{i,j}_{k,l}$ is optimal for finding edge ranking of $K_{i,j}$. This completes the proof of the correctness of the algorithm. Clearly, the algorithm has running time $O((ab)^2) = O(m^2)$. □

Theorem 2 *Let $K_{a,b}$ be a complete bipartite graph, let c be a nonnegative integer and let $E^{a,b}_c$ denote any set of edges such that $E^{a,b}_c \subset E(K_{a,b})$, $|E^{a,b}_c| \leq c$. If c is a constant number then there exists an $O(t^c m^2)$ algorithm to compute edge ranking of graph $K_{a,b} - E^{a,b}_c$, where $t > 0$ is a constant number.*

Proof: Let us consider a set $E_c^{a,b}$. Let

$$\{v_1^{c_a}, \ldots, v_{c_a}^{c_a}\} = \{v \in V_1(K_{a,b}) \mid \deg(v) < b\},$$

$$\{v_1^{c_b}, \ldots, v_{c_b}^{c_b}\} = \{v \in V_2(K_{a,b}) \mid \deg(v) < a\}.$$

Let us consider fixed values of variables i, j, k, l in the procedure given in the proof of Theorem 1. Let G' be a graph induced by i vertices in set $V_1(K_{a,b} - E_c^{a,b})$ and j vertices in $V_2(K_{a,b} - E_c^{a,b})$. If $v_1, v_2 \in V_1(G'), \deg(v_1) = \deg(v_2) = j$ and $U = V(G') \setminus \{v_1, v_2\}$, then graphs induced by vertices $U \cup \{v_1\}$ and $U \cup \{v_2\}$ are isomorphic. Thus to find edge ranking of all subgraphs containing i vertices from $V_1(K_{a,b} - E_c^{a,b})$ and j vertices from $V_2(K_{a,b} - E_c^{a,b})$ we have to consider all subsets of type $\{v_1^{c_a}, \ldots, v_{c_a}^{c_a}, v_1^{c_b}, \ldots, v_{c_b}^{c_b}\}$ in $K_{i,j}$. Therefore there are at most $2^{c_a} 2^{c_b}$ cases to consider. Since the cardinality of the separator $S_{k,l}^{i,j}$ can be computed in linear time so for fixed values of variables i, j, k, l the running time of the algorithm is $O(c 2^{2c_a} 2^{2c_b}) = O(t^c)$. Thus, $O(t^c m^2)$ is the complexity of the procedure similar to the procedure described in the proof of Theorem 1. The dimension of array *Opt* is $a \times b \times 2^{c_a} \times 2^{c_b}$, which is a polynomial if c is constant. □

From Theorem 2 we obtain the following

Corollary 1 *There exists an $O(m^{2+d})$ algorithm for the edge ranking problem of almost complete bipartite graphs $K_{a,b} - E_{d \log(m)}^{a,b}$, where $d \geq 0$ is a constant number.* □

The above results can be extended to complete k-partite graphs for fixed $k > 2$. Let $K_{s_1,\ldots,s_k}$ denote the complete k-partite graph. Define a set of edges

$$S_{p_1,\ldots,p_k} = \{v_q^i v_r^j | i, j = 1, \ldots, k, i \neq j, q = 1, \ldots, p_i, r = p_j + 1, \ldots, s_j\}.$$

Note that

$$|S_{p_1,\ldots,p_k}| = \sum_{i=1}^{k} \sum_{j=1, j \neq i}^{k} p_i(s_j - p_j).$$

Lemma 2 *If c is an edge ranking of $K_{s_1,\ldots,s_k}$ using l colors then there exists an edge l-ranking c' and positive integers $0 \leq p_i \leq s_i$, $i = 1, \ldots, k$ such that unique colors $l - |S_{p_1,\ldots,p_k}| + 1, \ldots, l$ are assigned to the edges in $S_{p_1,\ldots,p_k}$.* □

We omit the proof because it is analogous to the proof of Lemma 1. As before, we store the values of $\chi'_r(K_{s'_1,\ldots,s'_k})$, $s'_i \leq s_i$ in k-dimensional matrix *Opt*, i.e.

$$Opt[s'_1, \ldots, s'_k] = \chi'_r(K_{s'_1,\ldots,s'_k}).$$

Consider the graph $K_{s'_1,\ldots,s'_k}$. Assume that edge rankings of all graphs $K_{s''_1,\ldots,s''_k} \neq K_{s'_1,\ldots,s'_k}$ have been already computed, $0 \leq s''_i \leq s'_i$. From Lemma 2 we have

$$\chi'_r(K_{s'_1,\ldots,s'_k}) = \min\{|S_{p_1,\ldots,p_k}| + \max\{\chi'_r(K_{p_1,\ldots,p_k}), \chi'_r(K_{s'_1-p_1,\ldots,s'_k-p_k})\}\}, \quad (3)$$

where $p_i = 0, \ldots, s'_i, i = 1, \ldots, k$. Thus, in order to compute $\chi_r(K_{s'_1,\ldots,s'_k})$ using equation (3), we have to consider $(s'_1 + 1) \cdot \ldots \cdot (s'_k + 1)$ separators. Note that computing an address in the *Opt* array requires time proportional to k. We find the value of $|S_{p_1,\ldots,p_k}|$ in time $O(k^2)$, so we can find the cardinalities of all minimal separators and store them in additional array in time $O(k^3 s_1 \cdot \ldots \cdot s_k)$. After this initialization, we obtain values of $|S_{p_1,\ldots,p_k}|$, $\chi'_r(K_{p_1,\ldots,p_k})$ and $\chi'_r(K_{s'_1-p_1,\ldots,s'_k-p_k})$ performing $O(k)$ operations in the main loop. So the running time of the algorithm is

$$O(k^3 s_1 \cdot \ldots \cdot s_k + k(s_1 \cdot \ldots \cdot s_k)^2) = O(k^3 m^{\lceil k/2 \rceil} + k m^{2\lceil k/2 \rceil}) = O(k m^{2\lceil k/2 \rceil}).$$

Thus we obtained the polynomial time algorithm for computing the edge ranking of complete k-partite graph. This algorithm can be extended to the algorithm for computing the edge ranking of graphs obtained by removing $O(\log m)$ edges from a complete k-partite graph. On the basis of the discussion above we can write the following theorem.

Theorem 3 *There exists an $O(km^{2\lceil k/2 \rceil})$ time algorithm for computing edge ranking of complete k-partite graph, where $k \geq 2$ is fixed. If G is a graph obtained from the complete k-partite graph by removing $O(\log m)$ edges then $\chi'_r(G)$ can be computed in polynomial time.* □

References

1. H. Bodlaender, J.S. Deogun, K. Jansen, T. Kloks, D. Kratsch, H. Müller, Z. Tuza, Rankings of graphs, *SIAM J.Discrete Math.* **11** (1998) 168-181.
2. J.S. Deogun, T. Kloks, D. Kratsch, H. Müller, On the vertex ranking problem for trapezoid, circular-arc and other graphs, *Discrete Appl. Math.* **98** (1999) 39-63.
3. A.V. Iyer, H.D. Ratliff, G. Vijayan, Parallel assembly of modular products - an analysis, *Tech. Report* 88-06, Georgia Institute of Technology, 1988
4. M. Katchalski, W. McCaugh, S. Seager, Ordered colourings, *Discrete Math.* **142** (1995) 141-154.
5. T.W. Lam, F.L. Yue, Edge ranking of graphs is hard, *Discrete Appl. Math.* **85** (1998) 71-86
6. T.W. Lam, F.L. Yue, Optimal edge ranking of trees in linear time, *Proc. of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, (1998) 436-445.
7. C.E. Leiserson, Area-efficient graph layout (for VLSI) *Proc. 21st Ann. IEEE Symp. on Foundations of Computer Science* (1980) 270-281.
8. J.W.H. Liu, The role of elimination trees in sparse factorization, *SIAM J.Matrix Analysis and Appl.* **11** (1990) 134-172.
9. F. Manne, Reducing the height of an elimination tree through local reorderings. *Tech. Report* 51, University of Bergen, Norway, 1991.
10. A. Pothen, The complexity of optimal elimination trees, *Tech. Report* CS-88-13, The Pennsylvania State University, 1988
11. A.A. Schäffer, Optimal node ranking of trees in linear time, *Inform. Process. Lett.* **33** (1989/90) 91-96.
12. P. de la Torre, R. Greenlaw, A.A.Schäffer, Optimal edge ranking of trees in polynomial time, *Algorithmica* **13** (1995) 529-618