AMO - Advanced Modeling and Optimization, Volume 12, Number 2, 2010

An Optimal Algorithm to Find Maximum Independent Set and Maximum 2-Independent Set on Cactus Graphs

Kalyani Das

Department of Mathematics,
Ramnagar College, Purba Medinipur, West Bengal, India.
e-mail: kdkalyanidas@gmail.com

Abstract.
A cactus graph is a connected graph in which every block is either an edge or a cycle. An optimal algorithm is presented here to find a maximum independent set and maximum 2-independent set on cactus graphs in $O(n)$ time, where n is the total number of vertices of the graph. The cactus graph has many applications in real life problems, specially in radio communication system.

Keywords: Design of algorithms, analysis of algorithms, independent set, 2-independent set, Euler tour, cactus graph.

AMS Subject Classifications: 68Q22, 68Q25, 68R10.

1 Introduction

Let $G = (V, E)$ be a finite, connected, undirected, simple graph of n vertices and m edges, where V is the set of vertices and E is the set of edges. A vertex v is called a *cutvertex* if removal of v and all edges incident to v disconnect the graph. A *non-separable graph* is a connected graph which has no cut-vertex and a *block* means a maximum non-separable sub-graph. A block is a *cyclic block* or simply *cycle* in which every vertex is of degree two.

A *cactus graph* is a connected graph in which every block is either an edge or a cycle.

A subset of the vertices of a graph $G = (V, E)$ is an independent set if no two vertices in this subset are adjacent. The maximum independent set (MIS) problem on G is to determine a

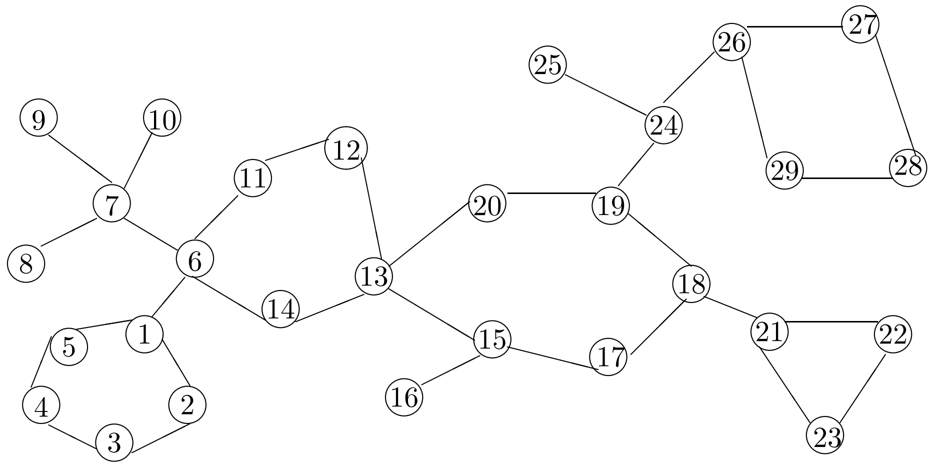

Figure 1: A cactus graph G.

maximum size independent set on G. The MIS problem is NP-complete for general graphs [2], but it can be solved in polynomial time for many special graphs [7].

The maximum k-independent set (MKIS) problem on G is to determine k disjoint independent sets $S_1, S_2, \ldots, S_k$ in G such that $S_1 \bigcup S_2 \bigcup \ldots S_k$ is maximum. The MKIS problem is NP-complete for general graphs [3].

The maximum 2-independent set (M2IS) problem, which is a special case of the MKIS problem, is also NP-complete for general graphs [15] and it applications have been studied in the last decade [6, 10, 15]. In [6], Hsiao et. al. have solved the two-track assignment problem by solving the M2IS problem on circular arc graph. In [10], Lou et. al. have solved the maximum 2-chain problem on a given point set, which is the same as the M2IS problem on permutation graph.

In this paper, MKIS problem is considered on a non-weighted cactus graph for $k = 1$ and $k = 2$.

Cactus graph has many applications. These graphs can be used to model physical setting where a tree would be inappropriate. Examples of such setting arise in telecommunications when considering feeder for rural, suburban and light urban regions [9] and in material handling network when automated guided vehicles are used in [8]. Moreover ring, star and bus structures are often used in local area networks. The combination of local area network forms a cactus graph.

To illustrate the problem we consider the cactus graph of Figure 1.

In the following section, we construct a tree T_{BC} whose nodes are the blocks of G and edges are defined between two nodes if they are *adjacent blocks, i.e.*, they have at least one common vertex of the graph G.

2 Construction of the Tree T_{BC}

As described in [12] the blocks as well as cut vertices of a graph G can be determined by applying DFS technique. Using this technique we obtain all blocks and cut vertices of the cactus graph $G = (V, E)$. Let the blocks be B_1, B_2, B_3,..., B_N and the cut vertices be C_1, C_2, $C_3, \dots, C_R$ where N is the total number of blocks and R is the total number of cut vertices.

The blocks and cut vertices of the cactus graph shown in Figure 1 are respectively $B_1 = (1,2,3,4,5), B_2 = (6,1), B_3 = (7,6), B_4 = (7,8), B_5 = (7,10), B_6 = (9,7), B_7 = (6,11,12,13,14)$, $B_8 = (13,15,17,18,19,20), B_9 = (15,16), B_{10} = (18,21), B_{11} = (21,22,23), B_{12} = (24,19), B_{13} = (24,25), B_{14} = (24,26), B_{15} = (26,27,28,29)$ and $C_1 = 1, C_2 = 6, C_3 = 7, C_4 = 13, C_5 = 15, C_6 = 18, C_7 = 19, C_8 = 21, C_9 = 24, C_{10} = 26$.

Now we have in a position to construct the tree T_{BC}. Before constructing the tree we define *an intermediate graph* G' whose vertices are the blocks of G and if two blocks are adjacent in G then they are connected by an edge in G'.

i.e., $G' = (V', E')$ where $V' = \{B_1, B_2, \dots, B_N\}$

and $E' = \{(B_i, B_j) : i \neq j, i, j = 1, 2, \dots, N,\ B_i$ and B_j are adjacent blocks $\}$.

The graph G' for the graph G of Figure 1 is shown in Figure 2.

Two properties of the graph G' are described below.

Lemma 1 *In G' there exists no cycle of length more than 3.*

Lemma 2 *The three vertices of G' forming a triangle must have a common cut vertex of G.*

Now the tree T_{BC} is constructed from G' as follows:

We discard some suitable edges from G' in such a way that the resultant graph becomes a tree. The procedure for such reduction is given below:

Let us take any arbitrary vertex of G', containing at least two cut-vertices of G, as root of the tree T_{BC} and mark it. All the adjacent vertices of this root are taken as children of level one and mark them. If there are edges between the vertices of this level, then discard these edges. Each vertices of level one is considered one by one to find the vertices which are adjacent to them but unmarked. These vertices are taken as children of the corresponding vertices of level one and put them at level two. These children at level two are marked and if there be any edge between them then remove them. This process is continued until all the vertices are marked.

Thus the tree $T_{BC} = (V', E'')$ where $V' = \{B_1, B_2, \dots, B_N\}$ and $E'' \subset E'$ is obtained.

For convenience, we refer the vertices of T_{BC} as nodes.

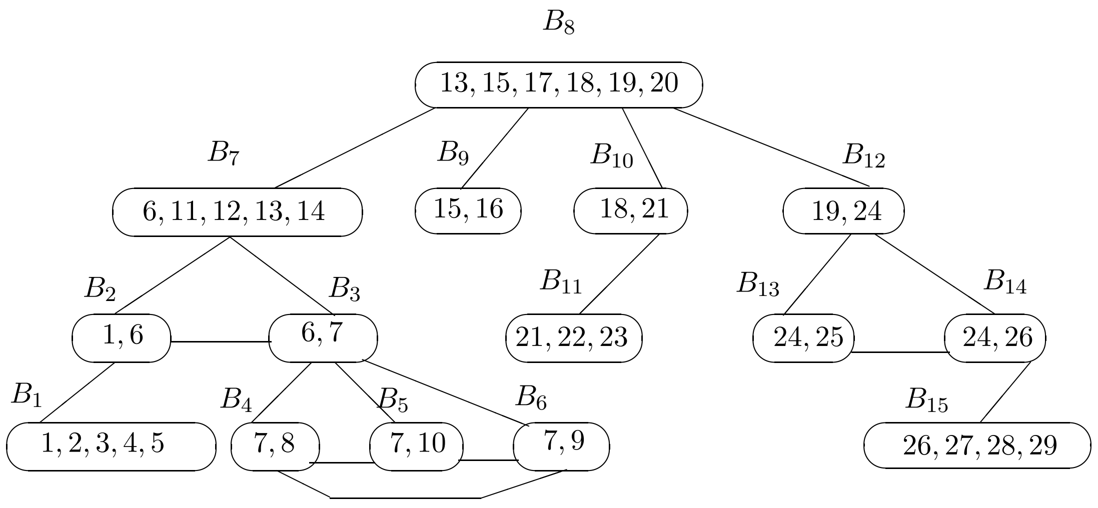

Figure 2: The intermediate graph G' of G.

We note that each node of this tree is a block of the graph $G = (V, E)$.

The parent of the node B_i in the tree T_{BC} will be denoted by *Parent(B_i)*. The tree T_{BC} constructed from G' is given in Figure 3.

3 Euler Tour

Euler tour produces an array of nodes. The tour proceeds with a visit to the root and there after visits to the children of the root one by one from left to right returning each time to the root using tree edges in both directions. Algorithm GEN-COMP-NEXT of Chen et al. [1] implements this Euler tour on a tree starting from the root. The input to the algorithm is the tree represented by a 'parent of' relation with explicit ordering of the children. The output of the algorithm is the tour starting from the root of the tree and ending also at the root. The tour is represented by an array $S(1 : 2N - 1)$ that stores information connected to the visits during the tour. The element $S(i)$ of the array S is a record consisting of two fields, one of which, denoted by $S(i).node$, is the node visited during the ith visit while the other, denoted by $S(i).subscript$ is the number of times the node $S(i).node$ is visited during the first i visits of the tour. Two fields of an element of S are written together using the notation $(node)_{subscript}$.

Also, we consider an array $f(j)$ which stores the total number of occurrence of the block $B_j, j = 1, 2, 3, \ldots, N$ in the array $S(i), i = 1, 2, 3, \ldots, 2N - 1$. Thus $f(j)$ represents the number of visits of the block B_j in the Euler tour, *i.e.*, $f(j)$ is the maximum subscript of B_j in the array $S(i)$.

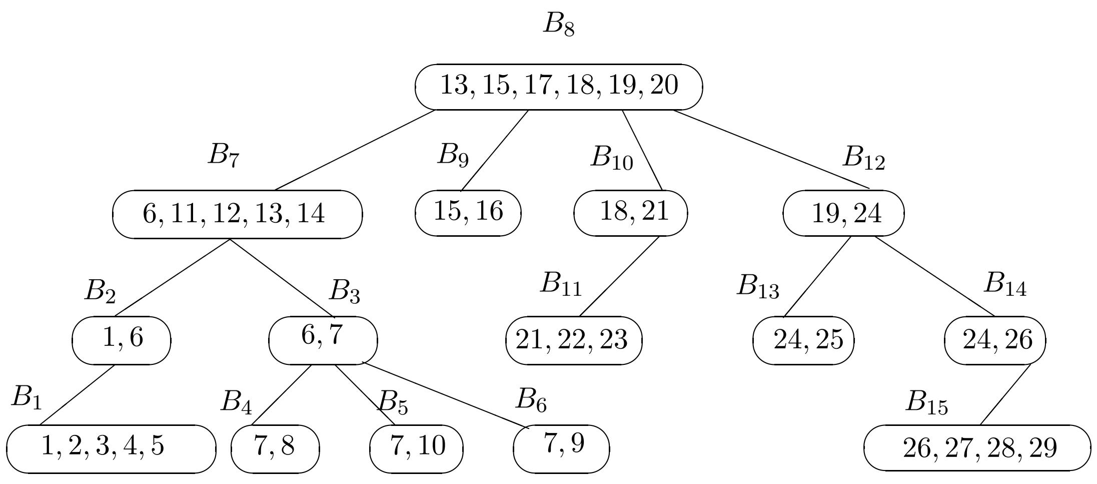

Figure 3: The tree T_{BC} of the graph G

i :	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S(i)$:	$(B_8)_1$	$(B_7)_1$	$(B_2)_1$	$(B_1)_1$	$(B_2)_2$	$(B_7)_2$	$(B_3)_1$	$(B_4)_1$	$(B_3)_2$	$(B_5)_1$	$(B_3)_3$	$(B_6)_1$	$(B_3)_4$	$(B_7)_3$	$(B_8)_2$
i :	16	17	18	19	20	21	22	23	24	25	26	27	28	29	
$S(i)$:	$(B_9)_1$	$(B_8)_3$	$(B_{10})_1$	$(B_{11})_1$	$(B_{10})_2$	$(B_8)_4$	$(B_{12})_1$	$(B_{13})_1$	$(B_{12})_2$	$(B_{14})_1$	$(B_{15})_1$	$(B_{14})_2$	$(B_{12})_2$	$(B_8)_5$	

Table 1: The sequence of nodes obtained from Euler tour.

The array S for the graph of Figure 1 is shown in Table 1.

For each j, $j = 1, 2, \ldots, N$, $(B_j)_{f(j)}$ occurs only once in the array $S(i)$ and before $(B_j)_{f(j)}$ all of $(B_j)_1, (B_j)_2, \ldots, (B_j)_{f(j)-1}$ occur in order of increasing subscripts of B_j.

The following important lemma is proved in [11].

Lemma 3 *If $S(i).subscript = 1$ and $S(i+1).subscript \neq 1$, then $S(i).node$ is a leaf node of the tree.*

In a tree there are three types of nodes. One is leaf node and others are interior and root node. Leaf node contains one cutvertex where as interior node contains at least two cutvertices and the root node contains one or more than one cutvertices.

4 Computation of Independent Set from a Leaf and an Interior Node

Since the blocks of the cactus graphs are cyclic the vertices others than the cutvertices are adjacent with only two vertices and cutvertices are adjacent with at least two vertices. Thus when we consider a block, say B we try to exclude the cutvertex of B and Parent(B) in the independent set I.

4.1 When the node is a leaf node

Suppose B be a leaf node (*block*) containing the vertices $\{v_1, v_2, \ldots, v_n\}$ where v_1 be the cutvertex of B and Parent(B).

Then we select the vertices $v_2, v_4, \ldots, v_n$ as member of the independent set I. The adjacent vertices of I are then marked by '$*$'.

4.2 When the node is an interior node

Suppose B be an interior node (*block*). Here some of the vertices are marked due to the consideration of its children nodes. So the free vertices occurred as one or more sequence of edges and vertices. Suppose such a sequence contains the vertices $\{v_1, v_2, \ldots, v_r\}$.

Then if r is even select either $v_1, v_3, \ldots, v_{r-1}$ or $v_2, v_4, \ldots, v_r$ as the members of I so that the set does not contain the cutvertex of B and Parent(B).

If r be odd then select $v_1, v_3, \ldots, v_r$ as members of I. If this set contains the cutvertex of B and Parent(B) then we does not include it when we consider B. It is included in I or not it is evident when we consider Parent(B).

Lemma 4 *A cycle with* $2m$ *and* $2m+1$ *vertices contribute* m *vertices in* I *and no more.*

Proof: Let us proof by contradiction. Suppose there exist more than m say, $m+1$ vertices in the independent set from a cycle with $2m$ and $2m+1$ vertices. Since these vertices form an independent set for the cycle they are not adjacent. There exist at least one vertex which are not belongs to the independent set between two of these vertices. Thus between $m+1$ vertices there exist another $m+1$ vertices to form the cycle. Hence the number of vertices of the block (*cycle*) becomes at least $2m+2$ which contradicts our assumption that the cycle contains $2m$ and $2m+1$ vertices. $\square$

Lemma 5 *A sequence of* $2m$ *vertices contribute* m *vertices and a sequence of* $2m+1$ *vertices contribute* $m+1$ *vertices in* I *and no more.*

Proof: Let us proof by contradiction. Suppose, there exist $m+1$ vertices for the sequence containing $2m$ vertices and $m+2$ vertices for the sequence containing $2m+1$ vertices. As in Lemma 4 between these two vertices of the independent set there exist at least one non independent vertex. Thus for the even case the number of vertices in the sequence becomes at least $2m+1$ and for the odd case the minimum number of vertices becomes $2m+2$ which contradicts our initial assumption. Hence the proof. □

5 The Algorithm and its Complexity

In the following algorithm we compute independent set from each node as well as from the graph G.

```
Algorithm MIS
Input: The cactus graph $G = (V, E)$.
Output: Independent set $I$.

    Step 1: Compute the blocks and cut vertices of $G$ and construct a tree $T_{BC}$.
    Step 2: Apply Euler tour on $T_{BC}$ and store the output in the array $S(1 : 2N - 1)$, $N$ is the
            total number of nodes of $T_{BC}$.
    Step 3: Compute $f(j)$ which stores total number of occurrences of the node $B_j$ in the array
            $S, j = 1, 2, \ldots, N$.
    Step 4: Note the order in which $(B_j)_{f(j)}$, $j = 1, 2, \ldots, N$ occurs in the array $S$.
    Step 5: Consider the nodes $B_j$ one by one following the order of Step 4 and
            (i) If $f(j) = 1$, i.e., for a leaf node $B_j$, find the vertices for the set $I$ using the
            method described in Section-3.1.
            (ii) If $f(j) \neq 1$, i.e., for an interior and root node $B_j$ find the vertices for the set $I$
            using the method described in Section-3.2.
end MIS
```

For the graph of Figure 1, the maximum independent set I is $\{2, 4, 8, 9, 10, 6, 12, 16, 22, 27, 29, 25, 17, 19\}$.

Lemma 6 *The independent set I obtain from the algorithm MIS is maximal.*

Proof: In the algorithm MIS we find the independent set from each cyclic block and from sequence of edges and vertices so that a cycle with $2m$ and $2m+1$ vertices contribute m vertices in I (from Lemma 4) and sequence containing $2m$ and $2m+1$ vertices contribute m vertices and $m+1$ vertices (from Lemma 5). No more vertices are obtained from G for the set I. Hence the set I becomes the maximum independent set for the graph G. □

Theorem 1 *The independent set obtained from the algorithm MIS is computed in $O(n)$ time.*

Proof: The blocks and cut vertices of any graph can be computed in $O(m+n)$ time [12]. For cactus graph $m = O(n)$, hence Step 1 of Algorithm MIS takes $O(n)$ time. As the array S is obtained by applying Euler's tour on the tree T_{BC}, Step 2 takes $O(n)$ time. Step 3 takes only $O(n)$ time. Step 5 can be perform by comparing $f(j)$ with 1 for $j = 1, 2, \ldots, n$, so this step takes only $O(n)$ time. Obviously, Step 4 takes $O(n)$ time. Hence the total time complexity of Algorithm MIS takes $O(n)$ times. □

6 Determination of Maximum 2-independent Set in Cactus Graph

In this section, we describe an algorithm to find a maximum 2-independent set from the graph G.

Algorithm M2IS
Input: The cactus graph $G = (V, E)$.
Output: Two disjoint independent set S_1 and S_2.

```
Step 1: Determine the blocks and cutvertices of $G$ using the method described in section-2
        and number of vertices of each block.
Step 2: Find the odd and even blocks according as the block contain odd and even number
        of vertices respectively. Define $S_O$ be the set of all odd blocks and $S_E$ be the set of
        all even blocks.
Step 3: For the blocks of the set $S_E$ there is no need of deletion of any vertex. For the blocks
        of the set $S_O$ we follow the following method:
        (i) Let $B_i \in S_O$ be a fixed block and if for all $B_j \in S_E$ where $B_i \neq B_j$ and
        $B_i \cap B_j \neq \phi$, then delete a non-cutvertex from $B_i$ and remove the blocks $B_i$ and $B_j$
        from $S_O$ and $S_E$ respectively.
        (ii) Let $B_i \in S_O$ be a fixed block and if for some $B_j \in S_E$ and some $B_k \in S_O$ or
        all $B_j \in S_O$ where $B_j, B_k \neq B_i$ and $B_i \cap B_j \neq \phi$ and $B_i \cap B_k \neq \phi$, then delete
        the cutvertex which contained in maximum number of odd blocks including $B_i$ and
        remove the blocks $B_i$ and $B_j$ from $S_O$ and $S_E$ respectively.
Step 4: After deletion of the vertices from odd blocks, label the rest vertices of $G$ as $R$ and
        $M$ so that two consecutive vertices do not label with same symbol.
Step 5: Take all the vertices with label $R$ in $S_1$ and the vertices with label $M$ in $S_2$.
        Thus we get two disjoint independent set $S_1$ and $S_2$ and the maximum 2-independent
        set is $S_1 \bigcup S_2$.
end M2IS
```

For the graph of Figure 1, the sets S_O, S_E, S_1, S_2 and maximum two-independent sets are

$S_O = \{B_1, B_7, B_{11}\}$
$S_E = \{B_2, B_3, B_4, B_5, B_6, B_8, B_9, B_{10}, B_{12}, B_{13}, B_{14}, B_{15}\}$
$S_1 = \{2, 4, 6, 8, 9, 10, 13, 16, 17, 19, 25, 21, 26, 28\}$,
$S_2 = \{1, 3, 7, 11, 14, 15, 18, 20, 22, 24, 27, 29\}$,
$S_1 \bigcup S_2 = G - \{5, 12, 23\}$.

Lemma 7 *The two sets S_1 and S_2 obtained from algorithm M2IS are maximal and disjoint.*

Proof: Using the algorithm M2IS, we divide the graph in terms of even blocks and odd blocks. In an even block the vertices are marked with two symbols alternatively and the vertices of different symbols are not adjacent. So marking the vertices with two symbols clearly give two disjoint sets. But for an odd block if we use two symbols, it is obvious that there exist two adjacent vertices which have same symbol. For this reason we delete one vertex from each odd block or common vertex of two or more odd blocks in Step 4 of the algorithm M2IS. Hence in algorithm we describe a method which shows that two adjacent vertices of the graph G have not marked with same symbol and since the set S_1 and S_2 are the sets containing vertices with different symbol these two sets are disjoint.

Now for the deletion of one vertex from each odd block we always try to delete the cutvertex which contained in maximum numbers of odd blocks instead of deletion of one vertex from each odd block. Thus deletion of vertices is minimized and therefore the number of vertices in S_1 and S_2 are maximized. Hence the theorem. □

Theorem 2 *The 2-independent set obtained from the algorithm M2IS is computed in $O(n)$ time.*

Proof: The blocks and cutvertices of any graph can be computed in $O(m + n)$ time [12]. For cactus graph $m = O(n)$, hence Step 1 of algorithm M2IS takes $O(n)$ time. Counting the number of vertices of each blocks, putting them into S_E and S_O obviously take $O(n)$ time. In Step 4, we perform comparison in which each even block is considered only one time and some odd blocks taken more time to find adjacent blocks but it does not exceed the number of blocks and number of block in cactus graph is $O(n)$. Hence Step 4 takes $O(n)$ time. Step 5 and Step 6 obviously take $O(n)$ time. Hence the complexity of the algorithm M2IS is $O(n)$. □

References

[1] Chen, C. C. Y., Das, S. K., and Akl, S. G., A unified approach to parallel depth-first traversals of general trees, *Information Processing Letters,* 41 (1991) 49-55.

[2] Garey, M. R. and Johnson, D. S., *Computer and Intractability: A Guide to the theory of NP-Completeness* (Freeman, San Francisco, CA, 1978).

[3] Gavril, F. and Yannakakis, M., The maximum k-colorable subgraph problem for chordal graphs, *Information Processing Letter*, 24 (1987) 133-137.

[4] Hota, M., Pal, M. and Pal, T. K., An efficient algorithm for finding a maximum weight k-independent set on trapezoid graphs, *Computational Optimization and Applications*, 18 (2001), 49-62.

[5] Hota, M., Pal, M. and Pal, T. K., An efficient algorithm to generate all maximal independent set on trapezoid graphs, *Intern. J. Computer Mathematics*, 70 (1999), 587-599.

[6] Hsiao, J. Y., Tang, C. Y. and Chang, R.S., An efficient algorithm for finding a maximum weight 2-independent set on interval graphs, *Information Processing Letters*, 43 (1992), 229-235.

[7] Johnson, D. S., The NP-completeness Column: An on going guide, *J.Algorithms*, 6 (1985) 434-451.

[8] Kariv, O. and Hakimi, S. L., An algorithmic approach to network location Problems, Part 1: The p-center, *SIAM J. Appl. Math.*, 37 (1979) 513-537.

[9] Koontz, W. L. G., Economic evaluation of loop feeder relief alternatives, *Bell System Technical J.,* 59 (1980) 277-281.

[10] Lou, R. D., Sarrafgadeh and Lee, D. T, An optimal algorithm for the maximum two-chain problem, *SIAM J, Discrete Math.*, 5 (1992) 285-304.

[11] Pal, M., and Bhattacharjee, G. P., An optimal parallel algorithm for all-pairs shortest paths on unweighted interval graphs, *Nordic Journal of Computing,* 4 (1997) 342-356.

[12] Reingold, E. M., Nivergent, J and Deo, N., Combinatorial Algorithms : Theory and Practice, (Prentice Hall, Inc., Englewood Chiffs, New Jersy, 1977).

[13] Saha, A., Pal, M. and Pal, T. K., An efficient pram algorithm for maximum-weight independent set on permutation graphs, *Journal of Applied Math.and Computing*, 19 (2005) 77-92.

[14] Saha, A. and Pal, M., Maximum weight k-independent set on permutation graphs, *Intern. J. Computer Mathematics*, 80 (12) (2003), 1477-1487.

[15] Sarrafgadeh, M. and Lee, D. T., A new approach to topological via minimization,*IEEE Trans. Computer Aided Design,* 8 (1989) 890-900.