ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 5, No. 3, 2010, *pp.* 211-223

An Optimal Algorithm to Find a Maximum Weight 2-Coloured Set on Cactus Graphs

Kalyani Das[1] and Madhumangal Pal [2, +]

[1]Department of Mathematics, Ramnagar College, Ramnagar, Purba Medinipur--721481, India.
[2]Department of Applied Mathematics with Oceanology and Computer Programming, Vidyasagar University, Midnapore -- 721102, India.

(*Received August 2, 2010, accepted August 8, 2010*)

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 weight 2-coloured 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, 2-colour set, cactus graph.

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 *cut-vertex* 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.

The Graph Colouring Problem (GCP) plays a central role in graph theory and it has direct applications in real life problems [2, 4, 22], and is related to many other problems such as timetabling [11, 23, 27], frequency assignment [14] etc. A K-colouring (assignment) of an undirected graph $G=(V,E)$, where V is the set of $|V|=n$ vertices and $E\subseteq V\times V$ the set of edges, is a mapping $\alpha: V\rightarrow\{1,2,\ldots,K\}$ that assigns a positive integer from $\{1,2,\ldots,K\}$ (representing the colours) to each vertex. We say that a colouring is feasible if the end nodes of every edge in E have assigned different colours, *i.e.*, for all $(u,v)\in E$, $\alpha(u)\neq\alpha(v)$. We call conflict the situation when two nodes between which an edge exists have the same colour associated to them. We say that a colouring is infeasible if at least one conflict occurs. Alternatively to the formulation as an assignment problem, the GCP can also be represented as a partitioning problem, in which a feasible K-colouring corresponds to a partition of the set of nodes into K sets $C_1, C_2,\ldots, C_K$ such that no edge exists between two nodes from the same colour class.

The graph colouring problem is NP-complete. Hence, we need to use approximate algorithmic methods to obtain solutions close to the absolute minimum in a reasonable execution time.

The Maximum Weight k-Colourable Subgraph (MWKC) problem is related to the following problem. The input to this problem consist of an integer number k and an undirected graph $G=(V,E)$, where each vertex v has a non-negative weight w_v. The goal is to pick a subset $V'\subseteq V$, such that there exists a colouring c of $G[V']$ with k colours, and among all such subsets, the value $\sum_{v\in V'} w_v$, w_v is maximum. This problem is NP-hard for general graph even for split graph [16].

The maximum weight k-colouring problem is same as the maximum weight k-independent set (MWKIS)

[+] *E-mail address*: mmpalvu@gmail.com

problem. The maximum k-independent set 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 MWKIS problem is NP-complete for general graphs [16].

Many work on Colouring Problem has been done previously. Local search in large neighbour and iterated local search for GCP are described in [10, 1]. In [3] there are different new methods to colour the vertices of the graph. Vertex colouring by multistage branch and bound method is described in [7] and by branch and cut algorithm is described in [12]. Greedy graph colouring is used in [5]. Genetic and hybrid algorithm for graph colouring is applied in [13, 15]. In [22] parallel graph colouring is applied. Graph colouring algorithm for assignment problem in radio network is done in [8] and ant algorithm for GCP is described in [9].

The maximum weight 2-colouring problem or the maximum weight 2-independent set (MW2IS) problem, which is a special case of the (MWKIS) problem, is also NP-complete for general graphs [26] and it applications have been studied in the last decade [17, 18, 21, 26]. In [17], Hsiao et.al. have solved the two-track assignment problem by solving the M2IS problem on circular arc graph. In [21], Lou et. al. have solved the maximum 2-chain problem on a given point set, which is the same as the MW2IS problem on permutation graph.

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 [19] and in material handling network when automated guided vehicles are used in [20]. Moreover ring 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 a weighted cactus graph of Figure 1. The numbers in the bracket of each vertices represent the weights of that vertices.

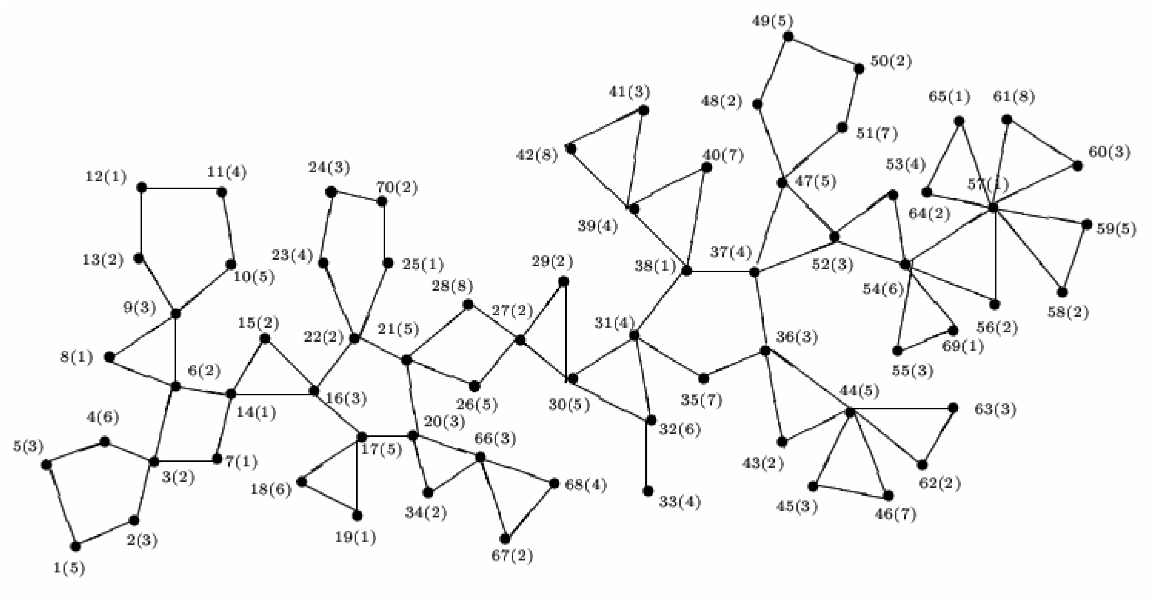


Figure 1: A cactus graph G.

In this paper maximum weight 2-colouring problem is considered on a weighted cactus.

2. Computation of blocks and cutvertices of G

As described in [25] 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, $c_3, \ldots, c_R$ 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 $B_1 = (1,2,3,4,5)$, $B_2 = (3,6,7,14)$, $B_3 = (6,8,9)$, $B_4 = (9,10,11,12,13)$, $B_5 = (14,15,16)$, $B_6 = (16,17,21,20,22)$,

$B_7 = (21,22,23,24)$, $B_8 = (20,66,34)$, $B_9 = (66,67,68)$, $B_{10} = (18,17,19)$, $B_{11} = (21,26,27,28)$, $B_{12} = (27,29,30)$, $B_{13} = (30,31,32)$, $B_{14} = (31,35,36,38,37)$, $B_{15} = (38,39,40)$, $B_{16} = (39,41,42)$, $B_{17} = (36,43,44)$, $B_{18} = (44,45,46)$, $B_{19} = (37,47,52)$, $B_{20} = (47,48,49,50,51,)$, $B_{21} = (52,53,54)$, $B_{22} = (54,56,57)$, $B_{23} = (59,57,58)$, $B_{24} = (57,61,60)$, $B_{25} = (32,33)$, $B_{26} = (54,55,69)$ $B_{27} = (57,64,65)$ $B_{28} = (44,62,63)$ and $c_1 = 3$, $c_2 = 6$, $c_3 = 9$, $c_4 = 14$, $c_5 = 16$, $c_6 = 22$, $c_7 = 20$, $c_8 = 66$, $c_9 = 17$, $c_{10} = 21$, $c_{11} = 27$, $c_{12} = 30$, $c_{13} = 31$, $c_{14} = 38$, $c_{15} = 39$, $c_{16} = 36$, $c_{17} = 44$, $c_{18} = 37$, $c_{19} = 47$, $c_{20} = 52$, $c_{21} = 54$, $c_{22} = 57$, $c_{23} = 32$.

Two blocks are said to be *adjacent* if they have at least one common vertex of the graph G. It may be noted that the number of vertices in a block is either odd or even. If the number of vertices is even(odd) then we call this block as even(odd) block.

To colour a cactus graph we construct a graph in terms of blocks and cutvertics and we refer the graph as block-cutvertex graph. The representation is described in the next section.

3. Representation of block-cutvertex graph

In this section, we determine the number of vertices in each block. Let BO and BE be the sets of all odd and even blocks respectively.

Using blocks of the graph G we define an intermediate graph $G' = (V', E')$ where $V' = \{B_1, B_2, \ldots, B_N\}$ and $E' = \{(B_i, B_j) : i \neq j, i, j = 1,2,\ldots,N$, B_i and B_j are adjacent blocks $\}$. From G' we construct another graph G'' where

$G'' = (V'', E'')$ where $V'' = BO$ and $E'' = \{(B_i, B_j) : i \neq j, B_i, B_j \in BO,$ B_i and B_j are adjacent blocks $\}$.

The graph G' for the graph G of Figure 1 is shown in Figure 2 and G'' is shown in Figure 3.

Here it is evident that edges between two adjacent blocks of the graph G' or G'' are nothing but the common cutvertex between them. The graph G'' is obtained from G' by deleting the even blocks and edges incident on those blocks.

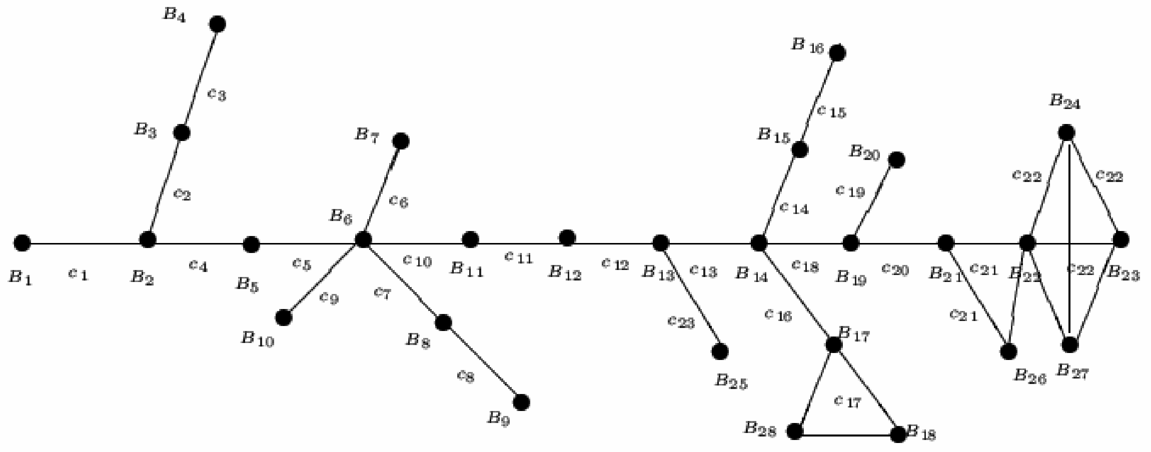


Figure 2: The intermediate graph G' obtained from the graph of Figure-1.

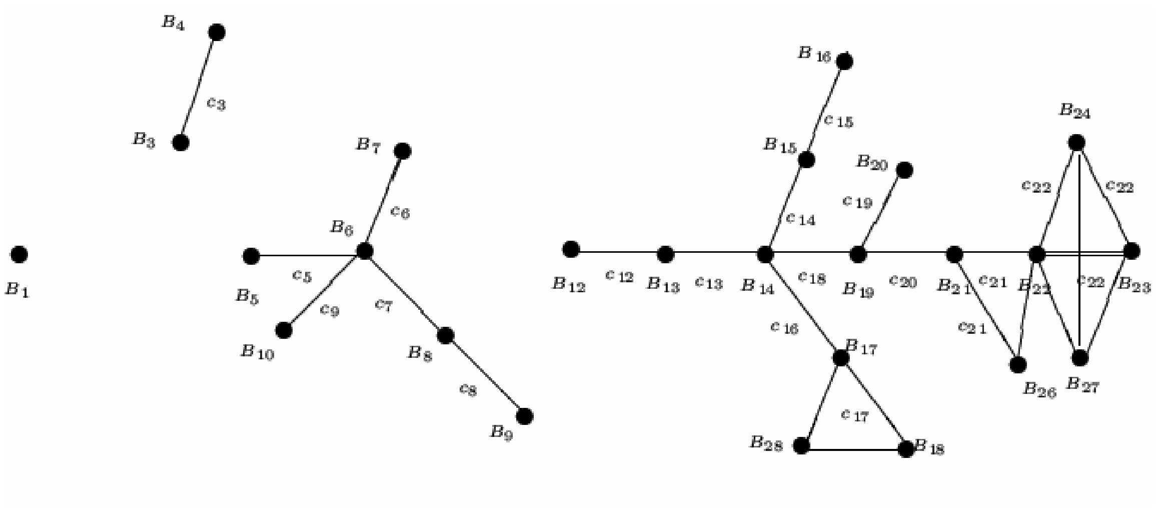


Figure 3: The graph G'' obtained from the graph G' of Figure-2.

Properties of G' and G'':

The properties of the graph G' are described below.

Lemma 1 In G' and G'' there exists no cycle of length more than three.

Proof: We prove it by contradiction. Suppose there exist a cycle of length 4 in G' or G'' as shown in Figure- 4(a). Here the vertices B_1 and B_3 are adjacent to B_2. Again vertex B_4 is adjacent to both the vertices B_1 and B_3. The representation of these blocks in original graph is G shown in Figure 4(b) which shows that the properties of cactus graph is not satisfied. Hence our assumption that there exist a cycle of length 4 is not true. Similarly, it can be shown that cycles of length more than four in G or G'' also do not exist. But, for the cycle of length 3 as in 4(c) in G' or G'', the representation of this in G satisfies the property of cactus graph as shown in 4(d). Hence the proof. □

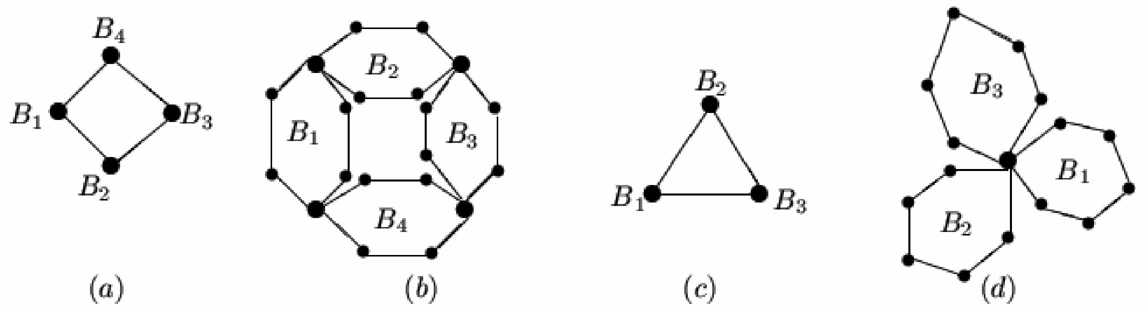


Figure 4: Illustration of lemma-1.

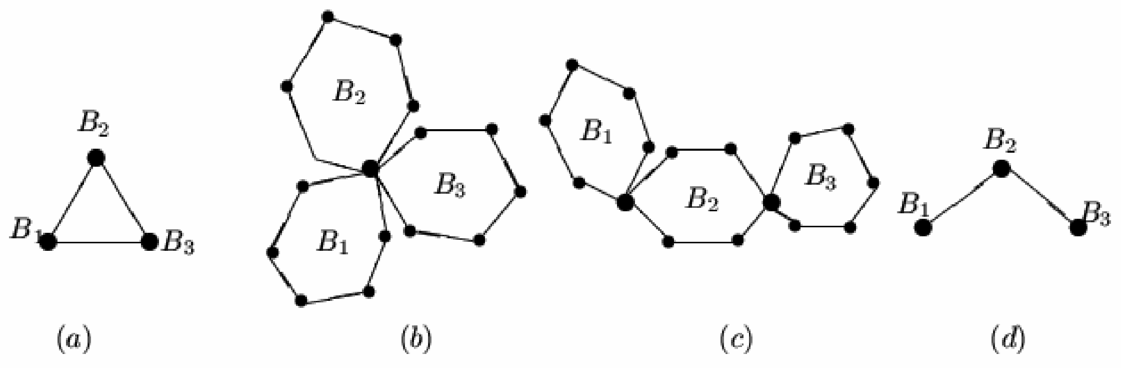


Figure 5: Illustration of lemma-2.

Lemma 2 Let there be a cycle of length three in G' and G''. The vertices corresponding to the edges of this cycle represent the same cutvertex in G.

Proof: Let there be a cycle of length three in G' or G'' as in Figure-4(a). By definition of G' and G'' the corresponding subgraph in G is drawn which shows in Figure-5(b). Obviously, this is a cactus graph. Again suppose three blocks of G do not have same cutvertex in common as in Figure-5(c) and its representation in G' and G'' is shown in Figure-5(d) which shows that they do not form a cycle. □

Lemma 3 Three or more vertices of G' and G'' form a clique provided the blocks corresponding to the vertices have a common cutvertex in G.

Proof: It is obvious from the property of cactus graph that two blocks have only one vertex in common. Hence there is only one edge between two vertices of G' and G'' as we define the edge of G' and G'' is the cutverx of G. Again when there exist three or more blocks with same common cutvertex in G then the vertices corresponding to these blocks in G' and G'' are connected by an edge with each other. As these vertices are connnected each other by an edge they form a clique. Hence the proof. □

Definition:

Clique: A clique Q in G' or G'' is defined as the subgraph of G' and G'' whose all vertices are connected by an edge with each other.

Adjacent Clique: Two cliques are said to be adjacent cliques if they have one common vertex in G' or G''.

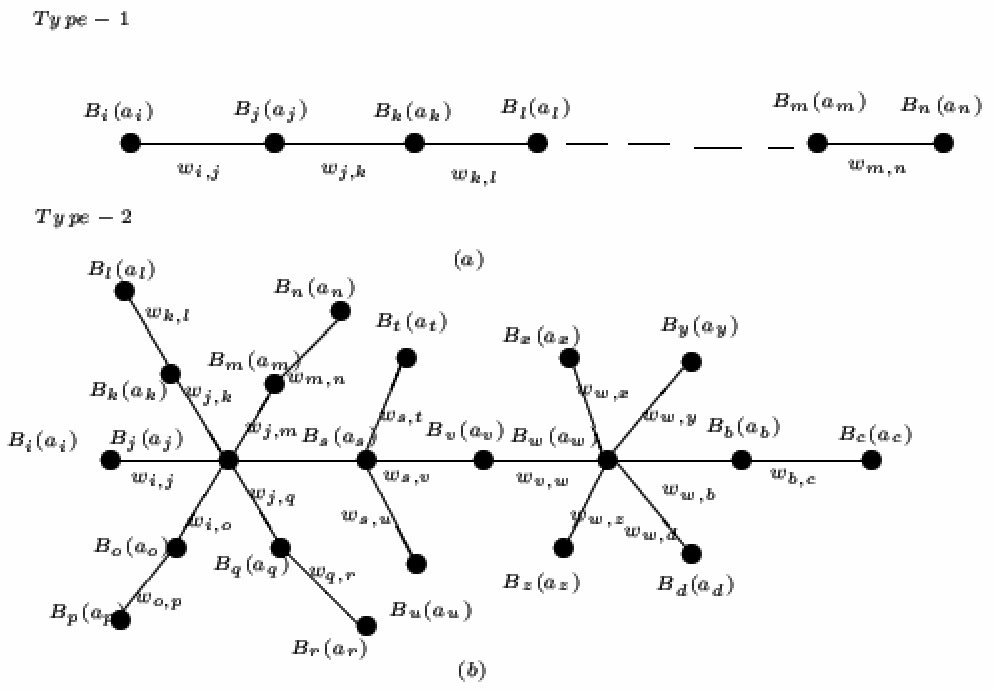
Lemma 4 Intersection of two adjacent cliques produce one vertex in G' or G'' but the intersection of blocks corresponding to the vertices of different cliques do not produce any vertex in G.

Proof: From the definition of adjacent clique the first part is proved.

For the second part, suppose intersection of two cliques produce a vertex in G. This imply that common vertex of G' and G'' i.e, corresponding block in G of that vertex contain only one vertex of G. But from the property of cactus graph it is clear that every block is either an edge or a cycle.

Again, if two cliques have a common vertex in G, all the blocks corresponding to the vertices of these two cliques contain that vertex i.e, all vertices of these two cliques belongs to same clique by definition. Thus our assumption is wrong. Hence the proof. □

Lemma 5 The graph G'' may be connected or disconnected graph.



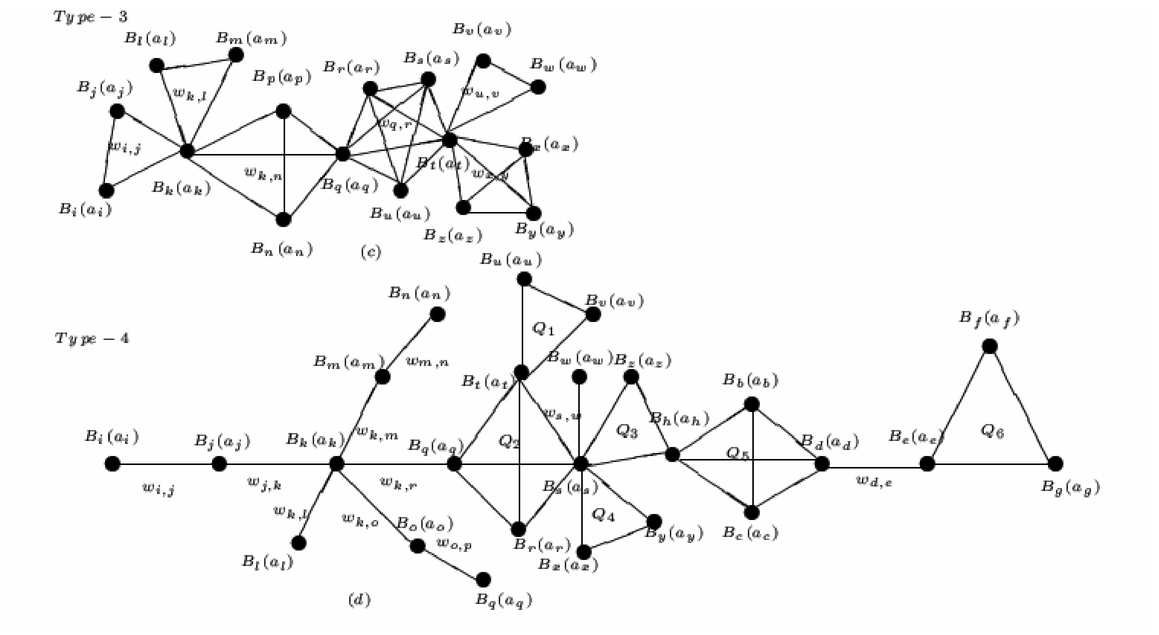


Figure 6: Different form of the graph G''.

4. Representation of G'' in terms of vertex weight and edge weight

Let $v_i \in B_i$ be the vertex whose weight is minimum and let $a_i = wt(v_i)$ and the weight of edge between two vertices B_i and B_j in G' and G'' is $w_{i,j}$. We sometimes refer a_i is the weight of the block B_i.

We denote the cutvertex of B_i and B_j as v_j^i which is a member of G.

The graph G'' or the connected components of G'' can be classified in the following ways.

Type-1: Some vertices in G'' occurs as isolated vertices or paths. These are shown in the Figure-6(a).

Type-2: Some vertices in G'' are of degree more than two and they do not form a cycles or cliques. This situation occurs when there exist more than two cutvertices in the blocks of G. In this case, G'' looks like a tree shown in Figure-6(b).

Type-3: Some vertices in G'' form cycles or cliques. This case occurs when there exist one cutvertex between three or more blocks of G. This is shown in the Figure-6(c).

Type-4: All the above types occur in this form. This is shown in Figure-6(d).

Lemma 6 *If B_k is an even block containing k vertices, then all the vertices of B_k can be coloured with two colours.*

Proof: In an even block, the number of vertices are even. Let two colours be denoted by `0' and `1'. If we colour alternate vertices with colour `0' and rest vertices with colour `1' it is seen that the set of vertices with colour `0' and set of vertices of colour `1' are disjoint and the union of these two sets gives all the vertices of B_k. Hence the proof. □

Lemma 7 If B_k is an odd block containing k vertices, the vertices of B_k can be coloured with three colours.

Proof: At first we remove one vertex, say v from the cycle B_k. Then $B_{k'} = B_k - \{v\}$ becomes even block and hence we can colour all the vertices of B_k by `0' and `1'. Lastly we colour v by using the colour `2'. Hence the proof. □

From these lemmas it is evident that to colour all the vertices of an odd block with two colours, one vertex remain colourless.

5. Formation of a tree T_{BQ} with the vertices of G''

There are three different types of subgraph which aries in G''. We consider the general case which arises in type-4 of G''.

In this subgraph we denote the cliques as $Q_1, Q_2, Q_3, \ldots$ and divide the vertices of a cliques in two ways:

(a) **Free Vertices:** Vertices which belongs to one clique are called *free vertices*.

(b) **Non-free Vertices:** Vertices which belongs to more than one cliques are called *non-free vertices*

Let $W_{Q_k} = wt(v_j^i) = w_{i,j}$ where v_j^i is the cutvertex joining to the blocks B_i and B_j and $B_i, B_j \in Q_k$.

Introduce a function F so that $F(a_i) = v_i, F(a_j) = v_j, \ldots$ where $v_i \in B_i$ and $wt\{F(a_i)$ *i.e,* $wt(v_i) = a_i$. Also $F(w_{ij}) = v_j^i$, $v_j^i = B_i \bigcap B_j \in G$ and $F(a_i + a_j) = \{F(a_i) \bigcup F(a_j)\}$ *i.e,* $\{v_i, v_j\}$.

For each clique we also define $W_{Q_k}^R = \sum_j a_j$, a_j is the weight of $B_j \in Q_k$, B_j's are the free vertices of Q_k and $F(a_j)$ are unique. If all $F(a_j)$ are not unique, *i.e,* if they represent same vetex in G then $\sum_j a_j = \sum_{j, j \neq i} a_j + a_l$, where i is the suffixes for which $F(a_i)$ represent the same vertex and l is one of i.

Now we construct a tree T_{BQ} whose nodes are the vertices of G'' (not forming cliques) and the cliques if they exists. First take any vertex B_1 (say), of G'' as root of the tree T_{BQ}. Find the cliques which contain B_0 and adjacent vertices which does not include in any clique containing B_0. Place them in level one and mark them. Thereafter find the vertices which are adjacent to the vertices of level one or contained in the cliques of level one. Place them in level two and marked. Applying same procedure we get the tree.

The weight of the edge between two nodes B_i and B_j is $w_{i,j}$ and no weight is given to the edges between a clique Q_k and its children nodes but weight $W_{Q_k}^R$ is given to the edge between the clique Q_k and its parent node. The weight of every node are either a_j for B_j and W_{Q_k} for Q_k.

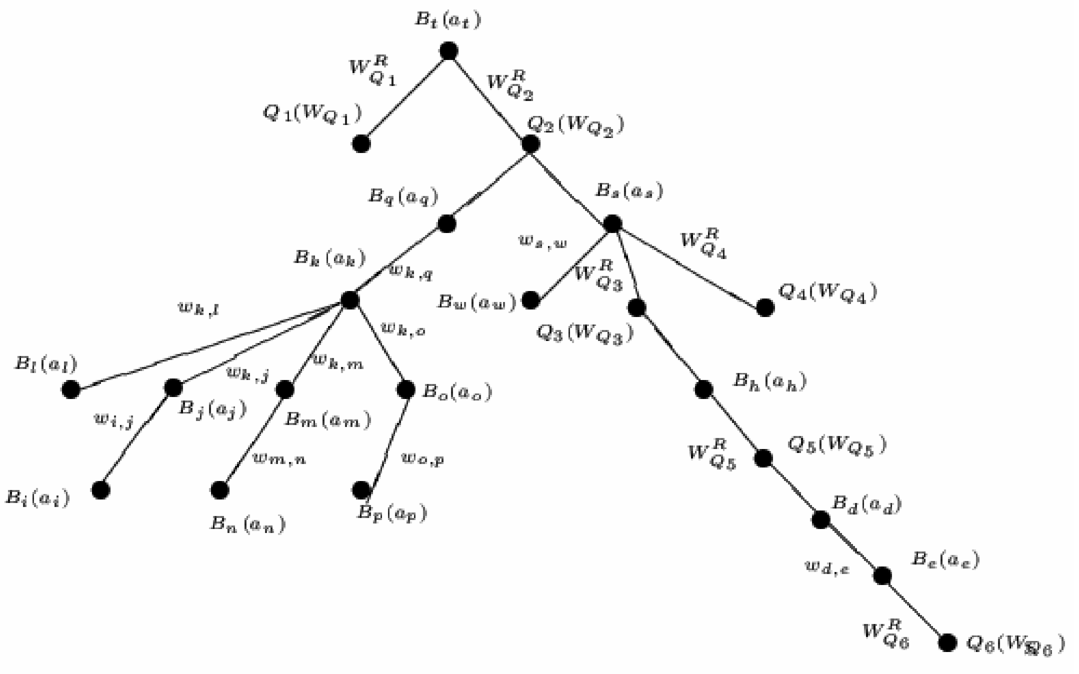


Figure 7: The Tree T_{BQ} of G''

The tree T_{BQ} for the graph G'' in Figure-6(d) is shown in Figure-7.

In the tree T_{BQ} the parent of a node B_i is denoted by $Pr(B_i)$ and child of B_i is denoted by $Ch(B_i)$.

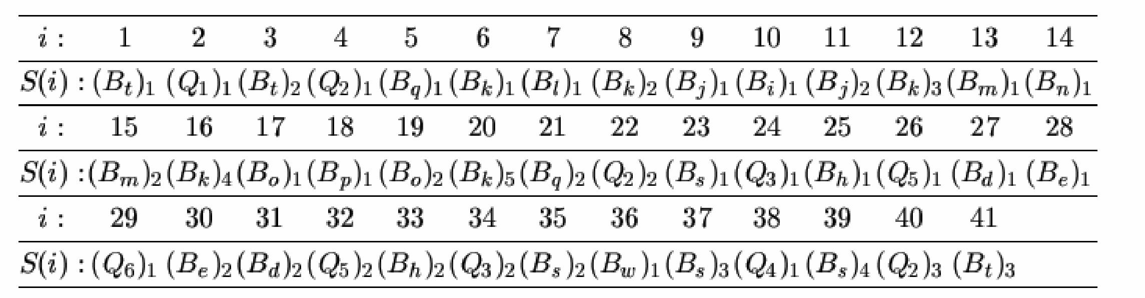
Now we find a sequence of nodes B_i so that before considering a node its children nodes are considered. This sequence of nodes are obtained from the following section.

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. [6] 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 $f(j)$ and $f(k)$ which stores the total number of occurrence of the block B_j and Q_k, $j = i, j, k, l, \ldots$ and $k = 1, 2, \ldots$, so that $|j + k| = N$ in the array $S(i), i = 1, 2, 3, \ldots, 2N-1$. Thus $f(j)$ and $f(k)$ represents the number of visits of the block B_j and Q_k in the Euler tour, *i.e.*, $f(j)$ and $f(k)$ is the maximum subscript of B_j and Q_k in the array $S(i)$.

The array S for the graph of Figure 7 is shown in Table 1.

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



i:	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$S(i)$:	$(B_t)_1$	$(Q_1)_1$	$(B_t)_2$	$(Q_2)_1$	$(B_q)_1$	$(B_k)_1$	$(B_l)_1$	$(B_k)_2$	$(B_j)_1$	$(B_i)_1$	$(B_j)_2$	$(B_k)_3$	$(B_m)_1$	$(B_n)_1$
i:	15	16	17	18	19	20	21	22	23	24	25	26	27	28
$S(i)$:	$(B_m)_2$	$(B_k)_4$	$(B_o)_1$	$(B_p)_1$	$(B_o)_2$	$(B_k)_5$	$(B_q)_2$	$(Q_2)_2$	$(B_s)_1$	$(Q_3)_1$	$(B_h)_1$	$(Q_5)_1$	$(B_d)_1$	$(B_e)_1$
i:	29	30	31	32	33	34	35	36	37	38	39	40	41	
$S(i)$:	$(Q_6)_1$	$(B_e)_2$	$(B_d)_2$	$(Q_5)_2$	$(B_h)_2$	$(Q_3)_2$	$(B_s)_2$	$(B_w)_1$	$(B_s)_3$	$(Q_4)_1$	$(B_s)_4$	$(Q_2)_3$	$(B_t)_3$	

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

The sequence of nodes obtained from the Euler Tour is $Q_1, B_l, B_i, B_j, B_n, B_m, B_p, B_o, B_k, B_q, Q_6, B_e,$

$B_d, Q_5, B_h, Q_3, B_w, Q_4, B_s, Q_2, B_t$.

The order of B_j's in this sequence is

$B_l, B_i, B_j, B_n, B_m, B_p, B_o, B_k, B_q, B_e, B_d, B_h, B_w, B_s, B_t$.

The following important lemma is proved in [24].

Lemma 8 If $S(i).subscript = 1$ and $S(i+1).subscript \neq 1$, then $S(i).node$ is a leaf node of the tree.

Find all the $Ch(B_j)$ and $Ch(Q_k)$ for $j = 1, 2, \ldots, r$ and $k = r+1, r+2, \ldots, N$ from the tree T_{BQ}.

6. Method to find a set of vertices with minimum weight which remain colourless

In this section we find the cumulative weight of each vertex B_j. We denote them as $W^+_{B_j}$ and $W^-_{B_j}$ where

$W^+_{B_j}$ represents the weight gained from its children node including the weight of B_j and $W^-_{B_j}$ represents the weight gained from its children node excluding the weight of B_j.

6.1. Finding minimum weight vertices from a leaf node

For a leaf node B_j, $W^+_{B_j} = a_j$ and $W^-_{B_j} = 0$

Also $F(W^+_{B_i}) = F(a_j) = \{v_j\}$ and $F(W^-_{B_i}) = \varnothing$.

6.2. Finding minimum weight vertices from interior node

For an interior node B_j find the children of it. They may be cliques or vertices of G''.

Case-1: Suppose they are vertices of G'' say, $B_k, B_l, B_m, \ldots$. Therefore $W^+_{B_r}$, $W^-_{B_r}$ and also $F(W^+_{B_r})$, $F(W^-_{B_r})$ for $r = k, l, m, \ldots$ are known.

Here $W^+_{B_j} = \min_r (\min(W^+_{B_r} + a_j, W^-_{B_r} + a_j + a_r, W^-_{B_r} + w_{r,j}) + \sum_{p\neq r} W^+_{B_p})$, $p = k, l, m, \ldots$

and $W^-_{B_j} = \sum_r W^+_{B_r}$.

Also find $F(W^+_{B_j})$ and $F(W^-_{B_j})$ which makes the $W^+_{B_j}$ and $W^-_{B_j}$ minimum.

Case-2: Suppose the children of B_j are cliques of G'' say, $Q_x, Q_y, Q_z, \ldots$. Here $W_{Q_k}, Ch(Q_k), B_s \in Q_k, k = x, y, z, \ldots$ are all known. Also $W^+_{Ch(Q_k)} = \sum W^+_{B_s}, W^-_{Ch(Q_k)} = \sum W^-_{B_s}$, $B_s \in Ch(Q_k)$, are known.

Here

$$W^+_{B_j} = \min_k \{\min\{W^-_{Ch(Q_k)} + \min(W_{Q_k}, \sum_{B_s\in Q_k} a_s), W^+_{Ch(Q_k)} + \min(a_j + W^R_{Q_k}, W_{Q_k})\} + \sum_{l\neq k} \min\{W^-_{Ch(Q_l)} + \min(\sum_{s\neq j, B_s\in Q_l} a_s, W_{Q_l})$$

and $W^-_{B_j} = \min_k \{W^-_{Ch(Q_l)} + \min(\sum_{s\neq j, B_s\in Q_k} a_s, W_{Q_k}), W^+_{Ch(Q_k)} + \min(W^R_{Q_k}, W_{Q_k})\}$

Also find $F(W^+_{B_j})$ and $F(W^-_{B_j})$ which makes the $W^+_{B_j}$ and $W^-_{B_j}$ minimum.

Case-3: Suppose some of the children of B_j are cliques and some are vertices of G'' as in Case-2 and Case-1. In this case

$W^+_{B_j} = \min$ {$W^+_{B_j}$ of Case-1+ $W^-_{B_j}$ of Case-2, $W^+_{B_j}$ of Case-2+ $W^-_{B_j}$ of Case-1}

and $W^-_{B_j}$ = {$W^-_{B_j}$ of Case-1+ $W^-_{B_j}$ of Case-2}. Also find $F(W^+_{B_j})$ and $F(W^-_{B_j})$ which makes the $W^+_{B_j}$ and $W^-_{B_j}$ minimum.

Lemma 9 Maximum weighted 2-colour set from two adjacent blocks B_i and B_j is obtained from the rest vertices other than the vertices $\{v_i, v_j\}$ or v^i_j.

Proof: For two adjacent blocks B_i and B_j suppose B_i is the root and B_j is the leaf node. For the leaf node

B_j we have from Section-6.2 $W^+_{B_i} = a_j$ and $W^-_{B_i} = 0$. Also $V(W^+_{B_i}) = F(a_j) = \{v_j\}$ and $V(W^-_{B_i}) = \varnothing$. When we consider B_i its child node is B_j and $W^+_{B_i} = \min\{W^+_{B_j} + a_i, W^-_{B_j} + a_i + a_j, W^-_{B_j} + w_{ij}\}$ which is $\min\{a_i + a_j, w_{ij}\}$. Also we know that $F(a_i) = v_i, F(a_j) = v_j$ and $F(w_{ij}) = v^i_j$. Hence rejection of the vertices $\{v_i, v_j\}$ or v^i_j produces the rest vertices give maximum weight. □

Lemma 10 Maximum weighted 2-colour set from three blocks B_i, B_j and B_k lying on a path is obtained from the rest vertices other than the vertices $\{v_i, v_j, v_k\}$ or $\{v^i_j, v_k\}$ or $\{v_i, v^j_k\}$.

Proof: The proof is similar to Lemma-9.

7. Algorithm and its Complexity

In this section we present the proposed algorithm MW2COL to compute the 2-colour set on cactus graphs. The time and space complexities are also computed here. The proof of correctness of the algorithm is also presented in this section.

```
Algorithm MW2COL

Input: The cactus graph $G$.
Output: The 2-colour set S.
Step 1: Compute the blocks and cutvertices of $G$ as describe in Section-3.
Step 2: Determine the set of odd blocks $BO$ and even blocks $BE$ and represent them into
        block-cutvertex graph $G'$ and $G''$ as described in Section-3.
Step 3: Represent $G''$ in terms of edge weight and vertex weight as described in Section-4.
Step 4: For $G''$ or for each connected components of $G''$ construct a tree $T_{BQ}$ as described
        in Section-5 and find the children of each node.
Step 5: Apply Euler tour to find a sequence of vertices to be considered in the tree and note
        their corresponding $f(j), j = 1, 2, \ldots, r$, $r$ being the number of vertices occured in
        the resulting sequence.
Step 6: For each node of the resulting sequence, if
        (i) $f(j) = 1$, then find the vertices of $G$ using rule described in Section-6.1 and put
        them in set $V_1$.
        (ii) $f(j) \neq 1$, then find the vertices of $G$ using the rule described in Section-6.2 and
        put them in set $V_1$.
Step 7: Delete the set of vertices $V_1$ from the graph $G$ i.e, $V - V_1$ is determined.
Step 8: Colour the alternate vertices of the set $V - V_1$ of $G$ with colour '0' and with colour
        '1'.

end MW2COL
```

Lemma 11 The weight of 2-colour set obtained from the algorithm $MW2COL$ is maximum.

Proof: Here the odd blocks are considered only to delete some vertices from these blocks because of Lemma-6 and Lemma-7.The algorithm $MW2COL$ describe a method for finding the set of those vertices with minimum total weight so that deletion of that set from the original graph give the maximum weight in rest vertices. Therefore the 2-colour set obtained from these vertices is maximum weighted 2-colour set. □

Theorem 1 The 2-colour set obtained from the algorithm $MW2COL$ is computed in $O(n)$ time.

Proof: The blocks and cutvertices of any graph can be computed in $O(m+n)$ time [25]. For the cactus graph $m = O(n)$, hence Step 1 of Algorithm $MW2COL$ takes $O(n)$ time. Also formation of G' and G''

take $O(n)$ time. Hence Step 2 can be computed in $O(n)$ time. In Step 3, the representation of G'' in term of edge and vertex weight obviosly takes $O(n)$ time. The construcction of tree T_{BQ} and finding children of each node and finding sequence of nodes using Euler Tour take $O(n)$ time. Hence Step 4 and Step 5 take $O(n)$ time. Step 6 can be perform by comparing $f(j)$ with 1 for $j=1,2,\ldots,r,r<n$, so this step takes only $O(n)$ time. Deletion of the set of vertices V_1 from G and giving colour to the rest vertices obviously take $O(n)$ time. Hence the complexity of the algorithm $MW2COL$ can be computed in $O(n)$. □

8. Illustration

The components of G'' is presented as tree in the Figure-7.

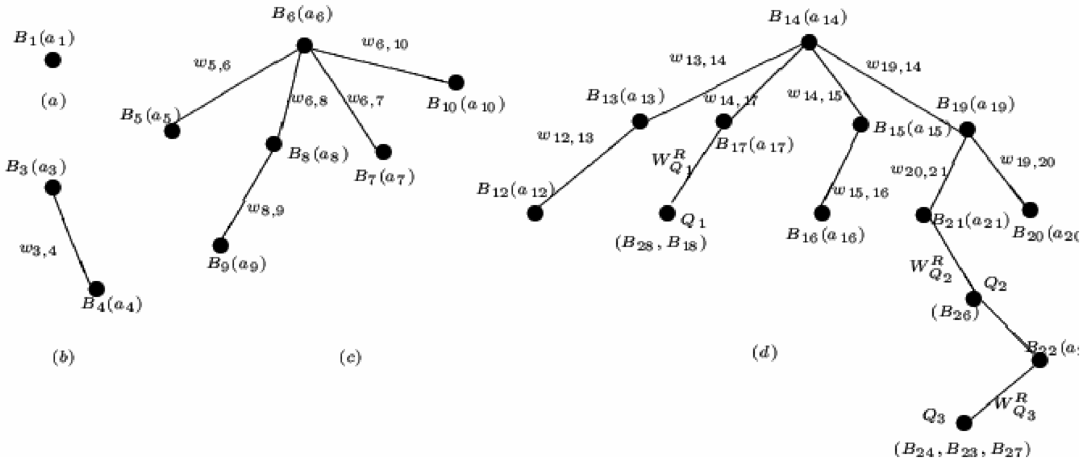


Figure 8: Verification of the method on the components of G''.

Here $a_1=2$, $a_3=1$, $a_4=1$, $a_5=1$, $a_6=2$, $a_7=1$, $a_8=2$, $a_9=2$, $a_{10}=1$, $a_{12}=2$, $a_{13}=4$, $a_{14}=1$, $a_{15}=4$, $a_{16}=3$, $a_{17}=2$, $a_{18}=3$, $a_{19}=3$, $a_{20}=2$, $a_{21}=3$, $a_{22}=1$, $a_{23}=1$, $a_{24}=1$, $a_{26}=1$, $a_{27}=1$, $a_{28}=2$.

and $v_1=3$, $v_3=8$, $v_4=12$, $v_5=14$, $v_6=22$, $v_7=25$, $v_8=34$, $v_9=67$, $v_{10}=19$, $v_{12}=27$, $v_{13}=31$, $v_{14}=38$, $v_{15}=38$, $v_{16}=41$, $v_{17}=43$, $v_{18}=45$, $v_{19}=52$, $v_{20}=48$, $v_{21}=52$, $v_{22}=57$, $v_{23}=57$, $v_{24}=57$, $v_{26}=69$, $v_{27}=57$, $v_{28}=62$.

The cycles $Q_1=(B_{17},B_{18},B_{28})$, $Q_2=(B_{26},B_{21},B_{22})$, $Q_3=(B_{22},B_{23},B_{24},B_{27})$. Free vertices of Q_1 are B_{18},B_{28}, Q_2 are B_{26} and Q_3 are B_{23},B_{24},B_{27}, so that $W^R_{Q_1}=a_{18}+a_{28}=5$, $W^R_{Q_2}=a_{26}=1$, $W^R_{Q_3}=a_{27}$ or a_{23} or $a_{24}=1$ as v_{27},v_{23},v_{24} represent the same vertex in G.

Also $w_{3,4}=w(c_3)=3$, $w_{5,6}=w(c_5)=3$, $w_{6,7}=w(c_6)=2$, $w_{6,10}=w(c_9)=5$, $w_{6,8}=w(c_7)=3$, $w_{8,9}=w(c_8)=3$, $w_{12,13}=w(c_12)=5$, $w_{13,14}=w(c_{13})=4$, $w_{14,15}=w(c_{14})=1$, $w_{15,16}=w(c_{15})=4$, $w_{14,17}=w(c_{16})=3$, $w_{14,19}=w(c_{18})=4$, $w_{19,20}=w(c_{19})=5$, $w_{19,21}=w(c_{20})=3$, $W_{Q_1}=w(c_{17})=5$, $W_{Q_2}=w(c_{21})=6$, $W_{Q_3}=w(c_{22})=1$.

and $v^3_4=c_3=9, v^5_6=c_5=16, v^7_6=c_6=22, v^{10}_6=c_9=17, v^8_6=c_7=20, v^8_9=c_8=66,$ $v^{12}_{13}=c_{12}=30, v^{13}_{14}=c_{13}=31, v^{14}_{15}=c_{14}=38, v^{15}_{16}=c_{15}=39, v^{14}_{17}=c_{16}=36, v^{17}_{18}=c_{17}=44, v^{14}_{19}=c_{18}=37,$ $v^{19}_{20}=c_{19}=47, v^{19}_{21}=c_{20}=52, v^{21}_{22}=c_{21}=54, v^{22}_{23}=c_{22}=57.$

Here the sequence of nodes obtained from Euler tour from Figure-7(a) is B_1, from Figure-7(b) is B_4,B_3, from Figure-7(c) is $B_5, B_9, B_8,\ B_7,\ B_{10}, B_6$, and from Figure-7(d) is $B_{12}, B_{13}, B_{17}, B_{16},\ B_{15}, B_{22}, B_{21}, B_{20}, B_{19}, B_{14}$.

Here we find $W^+_{B_1} + W^+_{B_4} + W^+_{B_6} + W^+_{B_{14}}$ and coreesponding vertices.

For the first component $W^+_{B_1} = a_1 = 2$ and $F(W^+_{B_1}) = \{v_1\} = \{3\}$

For the second component $W^+_{B_3} = a_3 = 1$, $W^-_{B_3} = 0$ and $F(W^+_{B_3}) = \{v_3\}$, $F(W^-_{B_3}) = \{\varnothing\}$. Now the child of B_4 is B_3. Thus $W^+_{B_4} = \min(W^+_{B_3} + a_4, W^-_{B_3} + a_3 + a_4, W^-_{B_3} + w_{3,4})$

i.e, $\min(1+1, 0+1+1, 0+3) = 2$. Here $F(W^+_{B_4}) = \{v_3, v_4\} = \{8,12\}$

Similarly from the third component $W^+_{B_6} = 7$ and $F(W^+_{B_6}) = \{v^7_6, v_{10}, v^8_9, v_5\} = \{22,19,66,14\}$

and fourth component $W^+_{B_{14}} = 21$ and $F(W^+_{B_{14}}) = \{v^{12}_{13}, c_{17}, v_{16}, v^{14}_{15}, c_{22}, v_{20}, v_{26}, v^{19}_{21}\} = \{30,44,41,38, 57,48,69,52\}$.

Thus $W^+_{B_1} + W^+_{B_4} + W^+_{B_6} + W^+_{B_{14}} = 2+1+7+21 = 31$

and $F(W^+_{B_1} + W^+_{B_3} + W^+_{B_6} + W^+_{B_{14}}) = \{3,8,12,22,19,66,14,30,44,41,38,48,57,69,52\}$

9. Minimum weight feedback vertex set for cactus graphs

For a finite graph $G = G(V, E)$, a set S is *feedback vertex set* if and only if the graph $G(V - S)$ has no cycles. The minimum weight feedback vertex set problem is to find a feedback vertex set such that the sum of weights of the vertices in this set is minimum among all such sets.

To find the minimum weight feedback vertex set S we have to select either the minimum weight vertex of each cycle or the cutvertex for two or more cycles as in 2-colour set problem. The only difference is in previous one we consider only odd blocks where as in this problem we consider both even and odd blocks except the edge blocks.

Thus we obtain the graph G'' from G' by deleting only vertices corresponding to the edge blocks and edges incident on that vertices. All properties of G' and G'' thus remain unchanged.

Using the same process we form a tree from G'', and applying the Euler tour on that tree and we get a sequence of nodes to consider one by one. Also applying the same procedure for each node we find the minimum weight as well as the set of vertices whose deletion from the graph G makes the graph cycle free. Thus we obtain the minimum feedback vertex set S.

10. References

[1] Ahuja, R.K., Magnanti, T.L. and Sharma, D. Very large-scale neighbourhood search. *International Transactions in Operational Research.* 2000, **7**: 295-302.

[2] Allen, M., Kumaran, G. and Liu, T. A combined algorithm for graph-colouring in register allocation. *Proceeding of the Computational Symposium on Graph Colouring and its Generalizations.* New York: Ithaca, 2002.

[3] Br´elaz, D. New methods to color the vertices of a graph. *Communications of the ACM.* 1979, **22**(4): 251-256.

[4] Barnier, N. and Brisset, P. Graph coloring for air traffic flow management. *In CPAIOR'02: Foruth International Workshop on Integration of AI and OR Techniques in Constraints Programming for Combinatorial Optimization Problems.* France: Le Croisic. 2002, pp. 133-147.

[5] Culberson, J.C. Iterated greedy graph coloring and the difficulty landscape. Technical Report 92- 07, Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada, June 1992.

[6] 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.* 1991, **41**: 49-55.

[7] Caramia, M. and Dell'Olmo, P. Vertex coloring by multistage branch and bound. *In D. S. Johnson, A. Mehrotra, and M. Trick, editors, Proceedings of the Computational Symposium on Graph Coloring and its Generalizations.* Ithaca, New York, USA, 2002.

[8] Comellas, F. and Oz_on, J. Graph coloring algorithms for assignment problems in radio networks. *Applications of Neural Networks to Telecommunications* (2. Edit). J. Alspector, R. Goodman y T.X. Brown, Lawrence Erlbaum Ass.. 1995, pp. 49-56. http://www-mat.upc.es/ comellas/radio/radio.html

[9] Costa, D. and Hertz, A. Ants can colour graphs. *Journal of the Operational Research Society*. 1998, **48**: 295-305.

[10] Chiarandini, M.and St¨utzle, T. An application of iterated local search to graph coloring. *In D. S. Johnson, A. Mehrotra, and M. Trick, editors, Proceedings of the Computational Symposium on Graph Coloring and its Generalizations.* Ithaca, New York, USA. 2002, pp.112-125.

[11] de Werra, D. An introduction to timetabling. *European Journal of Operation Research.* 1985, **19**: 151-162.

[12] Diaz, I.M. and Zabala, P. A branch-and-cut algorithm for graph coloring. *In D. S. Johnson, A. Mehrotra, and M. Trick, editors, Proceedings of the Computational Symposium on Graph Coloring and its Generalizations.* Ithaca, New York, USA. 2002.

[13] Fleurent, C. and Ferland, J. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research.* 1996, **63**: 437-464.

[14] Gamst, A.. Some lower bounds for a class of frequency assignment problems. *IEEE Transactions of Vehicular Technology.* 1986, **35**(1): 8-14.

[15] Galinier, P. and Hao, J. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization.* 1999, **3**(4): 379-397.

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

[17] 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*. 1992, **43**: 229-235.

[18] Hsu, W. L. and Tsai, K. H. A linear time algorithm or the two-track assignment problem. *Proceedings of 27th Allerton Conf. on Communication, Control and Computing,* 1989, pp. 91-300. going guide: *J.Algorithms*. 1985, **6**: 434-451.

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

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

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

[22] Lewandowski, G. and Condon, A. Experiments with parallel graph coloring heuristics and applications of graph coloring. In David S. Johnson and Michael A. Trick, editors. *Cliques, coloring and Satisfiability: Second DIMACS Implementation Challenge,* 1993. American Mathematical Society. 1996, **26**: 309-334.

[23] Leighton, F. T. A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards.* 1979, **85**: 489-506.

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

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

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

[27] Schaerf, A. A survey of automated timetabling. *Artificial Intelligence Review.* 1999, **13**: 87-127.