ELSEVIER Discrete Mathematics 165/166 (1997) 403–419

DISCRETE MATHEMATICS

Splitting trees

Pierre Hansen[a], Alain Hertz[b,*], Nicolas Quinodoz[b]

[a] *GERAD and Ecole des Hautes Etudes Commerciales, Montréal, Canada*

[b] *Département de Mathématiques, Ecole Polytechnique Féderale de Lausanne, Switzerland*

Abstract

Splitting a tree is defined as removing all edges of a chain and disconnecting one from the other edges incident with that chain. Splitting a forest is simultaneously splitting each of its non-trivial trees. The splitting number $\sigma(T)$ of a tree T is the minimum number of successive forest splittings which lead to deletion of all of T's edges. An $O(N)$ algorithm is proposed to get an upper bound $\sigma'(t)$, the connected splitting number, on the splitting number $\sigma(t)$ of a tree T and an $O(N \log N)$ algorithm to compute this last number, where N is the number of vertices of the tree.

Subject to a mild condition, these numbers lead to find a 'black-and-white coloring' of a tree T. In such a coloring a large part of T's vertices are colored in black or white and no two adjacent vertices receive a different color.

Keywords: Tree; Chain; Splitting; Polynomial algorithm; Coloring

Résumé

La *fente* d'un arbre est l'action qui consiste à ôter toutes les arêtes d'une de ses chaînes et à déconnecter entre elles toutes les arêtes incidentes à cette chaîne. La fente d'une forêt est la fente simultanée de chacun de ses arbres. La *sapinicité* $\sigma(T)$ d'un arbre T est le nombre minimum de fentes successives de forêts nécessaires pour ôter toutes les arêtes de T.

Nous proposons un algorithme en $O(N)$ pour obtenir une borne supérieure $\sigma'(T)$ sur $\sigma(T)$, et un algorithme en $O(N \log N)$ pour calculer $\sigma(T)$, où N est le nombre de sommets de T.

Sous certaines conditions peu restrictives, les nombres $\sigma(T)$ et $\sigma'(T)$ permettent de déterminer une "coloration en noir et blanc" d'un arbre T. Dans de telles colorations, la plupart des sommets de T sont colorés en noir ou en blanc et aucun sommet noir n'est adjacent à un sommet blanc.

Mots clés: arbre; chaîne; fente; algorithme polynomial; coloration

* Corresponding author. E-mail: hertz@dma.epfl.ch.

1. Introduction

We follow the terminology of Berge [1], to which we refer for undefined terms. Berge [2] studies the problem of finding how many black queens and how many white queens can be placed on an $N \times N$ chessboard without any black queen being in a position where it threatens a white queen (or vice versa). A related graph $G = (V, E)$ is obtained by associating vertices with positions and edges with pairs of vertices on the same horizontal, vertical or diagonal line. The problem, in decision form, is then to determine if x vertices of G may be colored black and y vertices white in such a way that no edge has a black and a white vertex. Both x and y are assumed to be strictly positive. Note that some vertices remain uncolored. The complexity of this problem on chessboard graphs, which we call 'black-and-white coloring', is unknown (the authors will provide a bottle of Black and White$^{\text{TM}}$ whisky for the first proof or disproof of NP-completeness).

Considering general graphs $G = (V, E)$ leads to an NP-complete problem. Indeed, the black-and-white coloring problem on the complementary graph $\bar{G} = (V, \bar{E})$ of G amounts to finding a partial complete bipartite graph $G' = (V_1, V_2, E')$ of $\bar{G}$ with $|V_1| = x$ and $|V_2| = y$. This problem is known to be NP-complete for $x = y$ [3]. The black-and-white coloring problem is however easy for some classes of graphs. If G is a chain, there is clearly a black-and-white coloring if and only if $x + y < N = |V|$. For trees, a dynamic programming algorithm, presented in the next section, gives a solution in $O(N^3)$ time. Moreover, we define a new parameter for trees T, the *splitting number* denoted $\sigma(T)$. This parameter yields a sufficient condition for existence of a black-and-white coloring of T, i.e., $x + y \leqslant N - \sigma(T)$. If this condition holds, which can be checked in $O(N \log N)$ time, a black-and-white coloring of $x + y$ vertices can be found in $O(N)$ time. Moreover, a precise upper bound on $\sigma(T)$, i.e., the *connected splitting number* $\sigma'(T)$, can be obtained in $O(N)$ time, which if $x + y \leqslant N - \sigma'(T)$ again lead to a black-and-white coloring of T in $O(N)$ time.

Given a tree T a *splitting* operation is defined as follows:

(i) choose a chain C of T with vertex set V_C and edge set E_C;

(ii) delete from T all edges of E_C, thus obtaining a forest $F_1 = T \backslash C$;

(iii) disconnect one from the other all edges of F_1 incident with V_C, i.e., make as many copies $v'_k, v''_k \ldots$ of each vertex v_k of V_C as there are edges $\{v_k, v_l\}, \{v_k, v_m\} \ldots$ incident with v_k in F_1 and replace these edges by $\{v'_k, v_l\}, \{v''_k, v_m\} \ldots$ thus obtaining a forest F_2. These three steps are illustrated in Fig. 1.

Splitting a forest is defined as splitting simultaneously each of its non-trivial trees (i.e., trees with edges). Let us call a *complete splitting* of a tree T the application of successive forest splittings until all edges of T are removed. A complete splitting of T defines a partition $P = (S_1, S_2, \ldots, S_z)$ of its edges where S_i denotes the edges removed in the ith forest splitting. The *level* of an edge is the index of the forest splitting in which it is removed. It follows that an edge has level i if and only if it belongs to S_i.

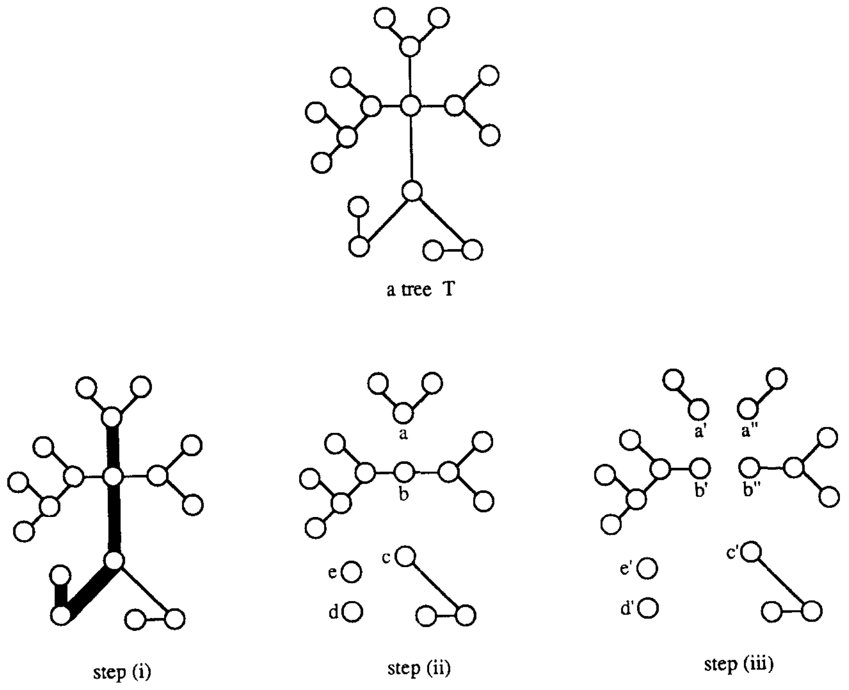

Fig. 1. A splitting operation.

The *splitting number* $\sigma(T)$ of a tree is defined as the smallest number of successive forest splittings in a complete splitting of T. Observe that a chain deleted at level i $(2 \leqslant i \leqslant z)$ does not necessarily share a common vertex with the chains deleted at a previous level. If it is assumed that each maximal chain in S_i $(2 \leqslant i \leqslant z)$ intersects an edge in at least one $S_j (j < i)$, another definition obtains, i.e., that of *connected splitting* and correlatively of *complete connected splitting* and *connected splitting number* $\sigma'(T)$.

Clearly, one could define a splitting operation and splitting number for a graph G in a similar way as for trees. The splitting number may be of independent interest. It corresponds to a process on a graph performed in parallel. While many parallel algorithms have been proposed for graph-theoretical problems, few parallel processes on graphs, and related parameters, appear to have been studied. A notable exception is broadcasting (or gossip propagation), see, e.g., the recent special number of Discrete Applied Mathematics [4].

2. Black-and-white coloring of trees

In this section we present an exact dynamic programming algorithm for the black-and-white coloring problem of trees, expressed in the following decision form.

BLACK-AND-WHITE COLORING

Input: Tree T, positive integers x and y.

Question: Does T admit a coloring of x of its vertices in black and y in white such that no edge has one black and one white vertex.

Vertices of T will be labelled B (black), W (white) or U (uncolored). The algorithm will in fact determine all non-dominated pairs of numbers (x, y) such that T admits a black-and-white coloring, i.e., pairs (x, y) such that there is no pair (x', y') satisfying this condition and such that $x' \geqslant x$, $y' \geqslant y$ and $(x, y) \neq (x', y')$.

For this purpose, T is oriented towards a leaf chosen as root. The vertices of T are then renumbered such that for any oriented edge (v_k, v_l) of T, $k < l$ holds. For any vertex v_i let v_j be its unique successor (if $i < n$) and $v_{i_1}, \ldots, v_{i_k}$ be its immediate predecessors (if v_i is not a leaf). We use the following notations:

— G_i is the subgraph of T induced by all vertices of index $\leqslant i$;

— V_i denote the set of vertices of G_i in the same connected component as v_i;

— T_i denote the subtree induced by $V_i \cup \{v_j\}$;

— $T_{i,r}$ denote the subtree induced by $V_{i_1} \cup \ldots \cup V_{i_r} \cup \{v_i\}$ $(1 \leqslant r \leqslant k)$.

For a subtree T' of T, three lists $B_{T'}$, $W_{T'}$ and $U_{T'}$ will be considered, one for each possible label of the root of T'. List $B_{T'}$ (respectively $W_{T'}$ and $U_{T'}$) contains all non-dominated pairs of numbers for T', assuming that the root of T' has label B (respectively W and U). Two operations will be performed, i.e.,

— *exstension* of a subtree T' which consists in determining the lists for the subtree obtained from T' by adding the unique arc out of its root;

— *fusion* of two subtrees T' and T'' having the same root and no other vertex in common, which consists in determining the lists for the subtree containing all vertices of T' and T''.

Algorithm LIST(T) described below generates all non-dominated pairs (x, y) for T. It uses procedures EXTENSION and FUSION for performing the above-mentioned operations.

```
Algorithm LIST(T)
— Orient the edges of T towards a leaf chosen as root and renumber the vertices of
  T so that $k < l$ holds for any oriented edge $(v_k, v_l)$ of T.
— For each leaf $v_i$ of T, define the lists for $T_i$ as follows: $B_{T_i} = \{(2,0)\}$, $W_{T_i} = \{(0,2)\}$
  and $U_{T_i} = \{(1,0), (0,1)\}$.
— Consider all vertices of T which are not a leaf in order of increasing indices. Let
  $\{v_{i_1}, \ldots, v_{i_k}\}$ be the set of immediate predecessors of $v_i$. Lists for $T_i$ are determined as
  follows (notice that lists for $T_{i,1} = T_{i_1}$ have already been determined):
    — call FUSION ($T_{i,j}$, $T_{i_{j+1}}$) for getting the lists of $T_{i,j+1}$, where j varies from 1 to $k-1$.
    — call EXTENSION ($T_{i,k}$) for getting the lists of $T_i$.
— Merge $B_T$ and $W_T$ and delete dominated pairs (notice that $T = T_{N-1}$ and that the
  third list $U_T$ does contain no pair which is not dominated or equivalent to one of
  the first two lists as the label of $v_{N-1}$ may always be shifted to the root $v_N$ if the
  latter is labelled U).
```

Procedure EXTENSION(T)

Lists for the subtree T' obtained from T by adding the unique arc out of the root of T are determined as follows:

— set $B_{T'} = \{(a+1, b)/(a, b) \in B_T \cup U_T\}$
$W_{T'} = \{(a, b+1)/(a, b) \in W_T \cup U_T\}$
$U_{T'} = B_T \cup W_T \cup U_T$.

— Remove pairs of a list which are dominated by a pair of the same list.

Procedure FUSION(T, T')

Lists for the subtree T'' containing all vertices of T and T' are determined as follows:

— set $B_{T''} = \{(a + a' - 1, b + b')/(a, b) \in B_T \text{ and } (a', b') \in B_{T'}\}$
$W_{T''} = \{(a + a', b + b' - 1)/(a, b) \in W_T \text{ and } (a', b') \in W_{T'}\}$
$U_{T''} = \{(a + a', b + b')/(a, b) \in U_T \text{ and } (a', b') \in U_{T'}\}$.

— Remove pairs of a list which are dominated by a pair of the same list.

Proposition 2.1. *The above algorithm provides a complete list of non-dominated pairs for T in $O(N^3)$ time.*

Proof. Correctness follows from that only dominated pairs are eliminated (Bellman's principle). Lists are of length at most N. Extension is in $O(N)$ and fusion in $O(N^2)$ using e.g. bucket sorting for elimination of dominated triples. As those operations are done N times at most, the algorithm is in $O(N^3)$. □

To find if T admits a black-and-white coloring with x black and y white vertices it suffices to consider the pair $(a = x, b)$ in the final list of non-dominated pairs; there is a black-and-white coloring if and only if $b \geqslant y$. Finding such a coloring is easy if pointers have been used in the algorithm to note which non-dominated pairs are implied in the extension and fusion operations to get new non-dominated pairs. Then a black-and-white coloring with $a = x$ black and b white vertices can be obtained. By removing the colour of $b - y$ white vertices one gets the desired coloring.

3. Complete splittings of trees

Note that, as shown in [5], a complete splitting with minimum splitting number is not necessarily obtained if, at each forest splitting, the largest chain or the chain passing by the largest number of vertices of degree $\geqslant 3$ is removed from each tree.

Given positive integers associated with all edges of T one may ask if these numbers correspond to levels of a complete splitting of T. Three edges with the same level k incident to a same vertex v will be called a *k-claw* of center v. A chain with at least 3 edges, with both extreme edges of level k and other edges of level $>k$ will be called a *k-chain.*

Theorem 3.1. *A numbering of the edges of T corresponds to levels of a complete splitting of T if and only if the following two conditions are satisfied*:

(i) *if there is a k-claw, there must be a fourth edge incident to its center with a number $<k$;*

(ii) *if there is a k-chain, there must be an edge incident to a non-extreme vertex of the chain with a number $<k$.*

Proof. *Necessity*: Consider levels associated with a complete splitting of T:

(i) The three edges of any k-claw cannot all be in the same chain, so the center v of the k-claw must belong to a chain already removed in a previous forest splitting. Hence, there must be an edge incident to v with level $<k$;

(ii) if there is a k-chain, its extreme edges cannot have been removed in the same forest splitting unless they have been disconnected by a previous forest splitting; therefore, there must be an edge incident with a non-extreme vertex of the chain with level $<k$.

Sufficiency: By induction on the levels. For level 1, condition (i) implies that the partial graph with edges of level 1 can have no vertex of degee $\geqslant 3$ and condition (ii) that it is connected; hence it is a chain. So assume that the edges of levels $1, 2, \ldots, k-1$ correspond to edges deleted by $k-1$ successive forest splittings. Let us show that the edges of level k correspond to those removed by one more forest splitting.

Consider the forest F obtained after the $k-1$ first forest splittings and let T' be any connected component of F. Subtree T' cannot contain a k-claw as otherwise the original tree T would have had an edge of level $<k$ incident with the center v of the k-claw and this claw would have been disconnected. Similarly, T' cannot contain a k-chain, as otherwise this k-chain would also have been disconnected due to an edge of level $<k$ incident to a non-extreme vertex of the k-chain. Hence, the edges of level k constitute a (possibly empty) chain in each connected component of F. $\square$

The next theorem characterizes complete connected splittings.

Theorem 3.2. *A numbering of the edges of T corresponds to levels of a complete connected splitting of T if and only if the following two conditions are satisfied*:

(i) *if there is a k-claw, there must be a fourth edge incident to its center with a number $<k$;*

(ii) *there is no chain in T having a non-extreme edge with a number larger than the numbers on the extreme edges of the chain.*

Proof. *Necessity*: Consider levels associated with a complete connected splitting of T. Condition (i) follows from Theorem 3.1. Assume there is a chain $C = (v_1, v_2, \ldots, v_p)$ in T having a non-extreme edge $\{v_i, v_{i+1}\}$ $(1 < i < p - 1)$ with level k and both extreme edges $\{v_1, v_2\}$ and $\{v_{p-1}, v_p\}$ with level $<k$. Consider the two subtrees T' and T'' obtained from T by deleting edge $\{v_i, v_{i+1}\}$. Since each maximal chain with level $l > 1$ in T must share a common vertex with a chain with level $<l$, it follows that T' and T'' must contain edges with level 1. Hence, the edges with level 1 do not induce a chain in T, a contradiction.

Sufficiency: Notice that condition (ii) implies that there is no k-chain in T. It follows from Theorem 3.1 that the numbering of the edges of T corresponds to levels of a complete splitting of T. Let e and e' be two edges in T with level $i > 1$ and 1, respectively. According to condition (ii) the numbers on the chain linking e to e' are decreasing. Hence, each maximal chain with number i intersects an edge with number $<i$. □

We next give a lower bound on the number of vertices of a tree with splitting number $\sigma(T)$.

Proposition 3.3. *The minimum number of vertices of a tree T with splitting number $\sigma(T) = k$ is at least $3^{k-1} + 1$ (where $k \geqslant 1$).*

Proof. By induction. For $k = 1$, T is an edge with 2 vertices and the result clearly holds. Assume the result holds for $k - 1$ and consider a tree with $\sigma(T) = k$. For a vertex v in T consider the forest F_v obtained from T by disconnecting one from the other all edges incident with v. Call the *score* of v the number of connected components in F_v with splitting number equal to $k - 1$. Observe first that T must contain a vertex with score $\geqslant 2$, else T would have splitting number $\leqslant k - 1$. We now prove that there is a vertex v in T with score $\geqslant 3$. Indeed suppose not and consider the set of vertices with score 2. All vertices in this set are on a chain C; otherwise there would be a vertex of score 3 at the intersection of chains joining pairwise 3 of them not on the same chain. Taking C as chain for a first splitting in T would lead to a complete splitting with $k - 1$ levels, a contradiction.

Now let $N(K)$ denote the minimum number of vertices of a tree with splitting number $\sigma(T) = k$. We have observed that $N(K)$ is at least 3 times $N(\mathrm{k} - 1)$ minus 2 and as $N(1) = 2$ one gets the result by solving the recurrence relation. □

Corollary 3.4. $\sigma(T) \leqslant \log_3(N - 1) + 1 \in O(\log N)$.

Finding a complete connected splitting of a tree T is easy and can be done by means of the following algorithm. Define a *pendant chain* as a chain such that one terminal vertex is a leaf, the other a vertex of degree at least 3 and all intermediary vertices are of degree 2.

```
Algorithm Complete-Connected-Splitting($T$)
  Initialization: Set $k = 0$ and $T' = T$.
  Recursive step
    Set $k = k + 1$. If $T'$ is a chain give level $k$ to all its edges and go to the final step. If
    $T'$ has a single vertex $v$ of degree $>2$ give level $k$ to all edges of $T'$ except one of
    them which is adjacent to $v$ and receives level $k + 1$; then go to the final step.
    Otherwise, give level $k$ to all edges of all pendant chains in $T'$; remove these edges
    from $T'$ and apply the recursive step to the resulting tree.
  Final step
    Let $L$ be the largest level used. Replace the level $l$ of each edge of $T$ by
    $l' = L - l + 1$.
```

Proposition 3.5. *The above algorithm finds a complete connected splitting of T in $O(N)$ time.*

Proof. Consider any edge $\{v, w\}$ of T. Let i be its level and let T' and T'' be the two connected subtrees obtained by removing $\{v, w\}$ from T. Observe that T' or T'' contains only edges with level $>i$. Hence, condition (ii) of Theorem 3.2 holds.

Assume that more than 2 adjacent edges have a same level $l > 1$, their common vertex is the extreme vertex of pendant chains with level l in the subtree containing all edges with level $\leqslant l$. It follows that this vertex is adjacent to an edge with level $<l$. No three adjacent edges can have level 1, as only edges of a chain receive such a level. Hence, (i) holds and it follows from Theorem 3.2 that the levels determined by the above algorithm correspond to a complete connected splitting of T.

For complexity, computing degrees, updating them and performing operations from the leaves inwards takes $O(N)$ time. □

Proposition 3.6. *The above algorithm finds a complete connected splitting with $\sigma'(T)$ levels.*

Proof. Let L be the largest level in T. It follows from Proposition 3.5 that the above algorithm provides a complete connected splitting of T with L levels. Hence, $\sigma'(T) \leqslant L$ and it remains to prove that $\sigma'(T) \geqslant L$.

Let T^i denote the subtree of T containing all edges of level $\leqslant i$. We prove by induction on the levels that $\sigma'(T^i) \geqslant i$. For $i = 1$ the result is clear. For $i = 2$, T^2 contains at least one vertex of degree $\geqslant 3$ which means that $\sigma'(T^2) \geqslant 2$. Assume the result holds for $i < k$ and consider $i = k$.

Consider a complete connected splitting $(S_1, \dots, S_{\sigma'(T^i)})$ of T^i using $\sigma'(T^i)$ levels (where S_i denotes the set of edges with level i). Let $\{v, w\}$ be any edge in $S_{\sigma'(T^i)}$ and consider the subtrees T' and T'' obtained by removing edge $\{v, w\}$ from T^i. We may assume, without loss of generality, that all edges of S_1 are in T'. According to condition (ii) of Theorem 3.2, all edges in T'' belong to $S\sigma'(T^i)$. It now follows from

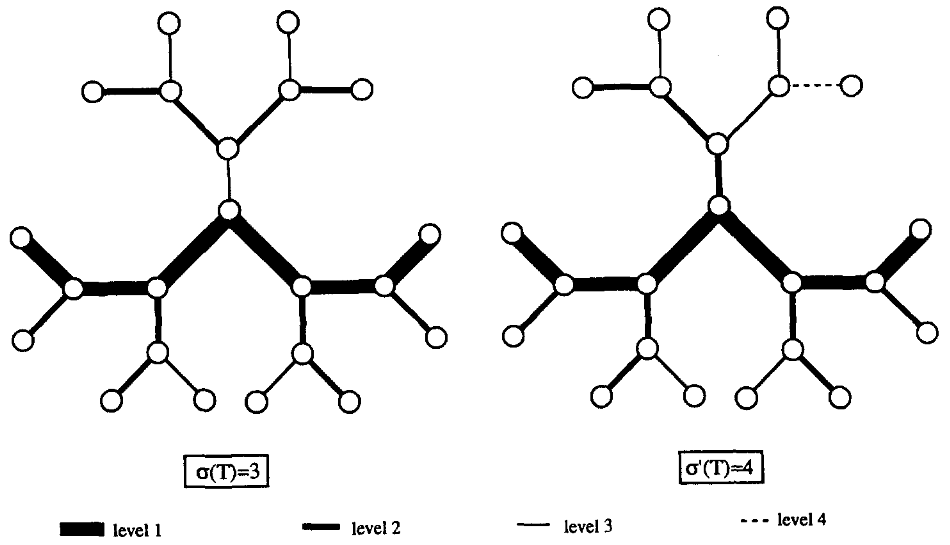

Fig. 2. The connected splitting number of a tree T may be strictly larger than the splitting number of T.

condition (i) of Theorem 3.2 that the subtree obtained by adding edge $\{v, w\}$ to T'' is a pendant chain in T^i.

Hence, all edges in $S_{\sigma'(T^i)}$ appear on pendant chains of T^i. Since T^{i-1} is obtained from T^i by deleting all pendant chains, we have $\sigma'(T^i) \geqslant \sigma'(T^{i-1}) + 1 = i$. □

Observe that the connected splitting number $\sigma'(T)$ of a tree T may be strictly larger than the splitting number $\sigma(T)$, as illustrated in the example of Fig. 2. To be able to get rapidly a black-and-white coloring of a larger proportion of trees we next consider the problem of finding efficiently the splitting number $\sigma(T)$ of a tree T.

For computing the splitting number $\sigma(T)$ of a tree T, the edges of T are first oriented towards a vertex chosen as root. We then assign to each arc (v_i, v_j) of T a level l_i and a set $\mathscr{F}_i$ of forbidden levels for the arc out of v_j. We use the following notations:

— $f(i)$ is the largest level which appears in at least two sets $\mathscr{F}_k$ and $\mathscr{F}_h$ of two arcs (v_k, v_i) and (v_h, v_i) entering v_i. If no such level exists then $f(i) = 0$;

— $r_\alpha(i)$ is the largest level appearing on at least α arcs entering v_i. If no such level exists the $r_\alpha(i) = 0$;

— P_i is the set of all levels in $\{1, \dots, N\}$ which do not appear in any set $\mathscr{F}_k$ of an arc (v_k, v_i) entering v_i.

Algorithm Complete-Splitting(T)

Initialization step

Orient the edges of T towards a leaf chosen as root and renumber the vertices so that $k < l$ holds for any arc (v_k, v_l) of T.

```
Sequential step
  Scan all vertices of $T$ except $v_N$ in order of increasing indices.
  If $v_i$ is a source then set $l_i = 1$ and $\mathscr{F}_i = \emptyset$.
  If $v_i$ is not a source then
  — define $l_i$ as follows:
      (a) if $r_1(i) \notin P_i$ or $r_1(i) \leqslant \max\{f(i), r_3(i)\}$ or $r_1(i) = r_2(i) = 1$ then
          set $l_i = \min\{l \in P_i / l > \max\{f(i), r_1(i)\}\}$;
      (b) if $r_1(i) \in P_i$ and $r_1(i) > \max\{f(i), r_3(i)\}$ and $r_1(i) > r_2(i)\}$ then set $l_i = r_1(i)$;
      (c) if $r_1(i) \in P_i$ and $r_1(i) > \max\{f(i), r_3(i)\}$ and $r_1(i) = r_2(i)\} > 1$ then set $l_i = 1$;
  — put in $\mathscr{F}_i$ all levels $l$ which are not in $P_i$ and which are strictly larger than $l_i$ and
      $r_2(i)$. Add level $r_2(i)$ to $\mathscr{F}_i$ if $r_2(i) > l_i$.
  Final step
    Let $L$ be the largest level used. Replace the level $l$ of each edge of $T$ by
    $l' = L - l + 1$.
```

Before proving that the above algorithm finds a complete splitting of T with $\sigma(T)$ levels, we make simple observations.

Observation 1. Level l_i belongs to P_i for each index, i, $1 \leqslant i < N$.

Indeed, this is clearly true if l_i is defined by case (a) or (b) of the sequential step. In case (c) l_i is set equal to 1 and it is sufficient to notice that level 1 cannot belong to any set of forbidden levels.

Observation 2. Let (v_i, v_j) be an arc of T and assume $l_j < l_i = r_1(j)$. Then $l_i \in \mathscr{F}_j$.

Indeed, $l_j < r_1(j)$ means that l_j is defined by case (c) of the sequential step. Hence, $l_j = 1 < r_2(j) = r_1(j) = l_i$ which means that $r_2(j) = l_i$ belongs to $\mathscr{F}_j$.

Observation 3. Let (v_i, v_j) be an arc of T and consider any level l in $\mathscr{F}_i$. If $l > l_j$ and $l \geqslant r_1(j)$ then $l \in \mathscr{F}_j$.

Indeed, $l \in \mathscr{F}_i$ implies $l \notin P_j$ and since $l > l_j$ it is sufficient to show that $l > r_2(j)$. Suppose not; then $l \leqslant r_2(j) \leqslant r_1(i) \leqslant l$ which means that $r_1(j) = l \notin P_j$. It follows that l_j is defined by case (a) of the sequential step and $l_j > r_1(j) > l$, a contradiction.

Theorem 3.7. *The levels defined by the above algorithm correspond to levels of a complete splitting of T.*

Proof. By induction on the number N of vertices in T. For $N \leqslant 2$ the result clearly holds. So assume the algorithm finds a complete splitting for trees having at most

$N-1$ vertices. It must be proved that conditions (i) and (ii) of Theorem 3.1 are satisfied. Consider the levels defined by the algorithm before applying the final step. By induction hypothesis, it is sufficient to show that

(i) if there is a k-claw of center v_{N-1} then there is an edge of level $>k$ incident to v_{N-1};

(ii) if there is a chain containing v_{N-1} as non-extreme vertex with both extreme edges of level k and other edges of level $<k$, then there is an edge of level $>k$ incident to a non-extreme vertex of the chain.

Proof. (i) Assume there is a k-claw of centre v_{N-1}. If the arc out of v_{N-1} is in the k-claw then $k = l_{N-1}$ and there are two arcs of level l_{N-1} entering v_{N-1}. It follows that $l_{N-1} \leqslant r_2(N-1) \leqslant r_1(N-1)$ which means that l_{N-1} is defined by case (c) of the sequential step. Hence, $l_{N-1} < r_1(N-1)$ and there is an arc entering v_{N-1} of level $> l_{N-1}$.

If the arc out of v_{N-1} is not in the k-claw then $k \leqslant r_3(j)$. Now if $r_1(N-1) = r_3(N-1)$ then l_{N-1} is defined by case (a) of the sequential step which means that $l_{N-1} > r_1(N-1) \geqslant k$. Otherwise, $r_1(N-1) > r_3(N-1) \geqslant k$. In all cases there is an edge of level $>k$ incident with v_{N-1}.

(ii) Consider a chain $C = (v_{j_0}, v_{j_1}, \dots, v_{j_p})$ containing v_{N-1} as non-extreme vertex with both extreme edges of level k and other edges of level $<k$. Let v_{j_q} be the root of the subtree of T induced by C and assume $q > 1$ without loss of generality. It follows that j_q equals $N-1$ or N and that $k = l_{j_0}$. We prove that there is at least one edge of level $> l_{j_0}$ incident with a non-extreme vertex of C. Suppose not; then $l_{j_0} \geqslant r_1(j_s)$, $1 \leqslant s \leqslant p-1$. Since l_{j_0} cannot be larger than $r_1(j_1)$ it follows that $l_{j_1} < l_{j_0} = r_1(j_1)$ which means, by Observation 2, that $l_{j_0} \in \mathscr{F}_{j_1}$. It then follows from Observation 3 that $l_{j_0} \in \mathscr{F}_{j_s}$, $1 \leqslant s < \min\{q, p-1\}$. Moreover, if $q < p-1$ then the same arguments imply $l_{j_p} = l_{j_0} \in \mathscr{F}_{j_s}$, $q < s \leqslant p-1$. As a consequence, the root v_{j_q} of C cannot be equal to v_N else $l_{j_0} = l_{N-1} \in \mathscr{F}_{j_{q-2}}$ which means that $l_{N-1} \notin P_{j_{q-1}} = P_{N-1}$, contradicting Observation 1. So $v_{j_q} = v_{N-1}$ and it follows that $l_{j_0} \in \mathscr{F}_{j_{q-1}}$ which means that $l_{j_0} \notin P_{j_q}$. Now,

— if $l_{j_0} = r_1(j_q)$ the l_{j_q} is defined by case (a) of the sequential step and $l_{j_q} > r_1(j_q) = l_{j_0}$;

— if $l_{j_0} > r_1(j_q)$ then $q > p-1$ which means that $l_{j_0} \in \mathscr{F}_{j_{q+1}}$. Hence, $(v_{j_{q-1}}, v_{j_q})$ and $(v_{j_{q+1}}, v_{j_q})$ are two arcs entering v_{j_q} and containing l_{j_0} in their set of forbidden levels. It follows that $f(j_q) \geqslant l_{j_0} > r_1(j_q)$. Level l_{j_q} is therefore defined by case (a) of the sequential step and we have $l_{j_q} > f(j_q) \geqslant l_{j_0}$.

In both cases level l_{j_q} is strictly larger than l_{j_0} which means that $\{v_{N-1}, v_N\}$ is an edge of level $> l_{j_0}$ incident to a non-extreme edge of C. $\square$

For a tree T, let E_T denote the set of edges e such that there exists a complete splitting $(S_1, \dots, S_{\sigma(T)})$ of T using $\sigma(T)$ levels with $e \in S_1$. In other words, E_T is the set of all edges which can possibly belong to the first chain deleted in a complete splitting of T made of $\sigma(T)$ forest splittings.

Moreover, consider the numbering of the vertices defined in the initialization step of the above algorithm and let T_i $(1 \leqslant i < N)$ denote the subtree containing v_i obtained from T by deleting the arc out of v_i.

Theorem 3.8. *Consider the levels l_i defined by the above algorithm before applying the final step and let L denote the largest level used. Then*
(1) *If $L \in \mathscr{F}_{N-1}$ then edge $\{v_{N-1}, v_N\}$ does not belong to E_T;*
(2) $\sigma(T) = L$.

Proof. By induction of the number N of vertices in T. For $N \leqslant 2$ the result clearly holds. So assume it holds for trees having at most $N-1$ vertices. Notice that it follows from Theorem 3.7 that $\sigma(T) \leqslant L$.

Proof. (1) Assume $L \in \mathscr{F}_{N-1}$. According to the sequential step of the above algorithm, either $L \notin P_{N-1}$ and $L > \max\{l_{N-1}, r_2(N-1)\}$ or else $l_{N-1} < r_2(N-1) = L$.
— If $L \notin P_{N-1}$ and $L > \max\{l_{N-1}, r_2(N-1)\}$, consider any arc (v_i, v_{N-1}) entering v_{N-1} with $L \in \mathscr{F}_i$. It follows that level L appears on at least one edge of T_i. By induction hypothesis we have $\{v_i, v_{N-1}\} \notin E_{T_i}$ and $\sigma(T_i) = L$. Hence, $\{v_i, v_{N-1}\} \notin E_T$ else the edges of T_i cannot be deleted with $\sigma(T)$ forest splittings. Since at least one edge of T_i must belong to the first chain deleted in a complete splitting of T using $\sigma(T)$ levels, it follows that $\{v_{N-1}, v_N\} \notin E_T$.
— If $l_{N-1} < r_2(N-1) = L$ there are two arcs (v_i, v_{N-1}) and (v_j, v_{N-1}) of level L and entering v_{N-1}. By induction hypothesis, we have $\sigma(T_i) = \sigma(T_j) = L$ which implies that $\sigma(T) = L$. Since the first chain deleted in a complete splitting of T using $\sigma(T)$ levels must contain at least one edge of T_i and one of T_j, it follows that $\{v_{N-1}, v_N\} \notin E_T$.

(2) It is sufficient to prove that $\sigma(T) \geqslant L$. If there is an arc $(v_i, v_j) \neq (v_{N-1}, \mathrm{v}_N)$ of level L then, by induction hypothesis, we have $\sigma(T_i) = L$ and, as $\sigma(T_i) \leqslant \sigma(T)$, we are done. So assume that $\{v_{N-1}, v_N\}$ is the unique edge in T of level L. Then $r_1(N-1) < L = l_{N-1}$ and it follows that l_{N-1} is defined by case (a) of the sequential step. We now examine the cases where $r_1(N-1) = L-1$ and $r_1(N-1) < L-1$ and prove that $\sigma(T) \geqslant L$ is both cases.

Case 2.1: $r_1(N-1) = L-1$. Consider any arc (v_i, v_{N-1}) of level $L-1$ entering v_{N-1}. It follows from the induction hypothesis that $\sigma(T_i) = L-1$.
— If $r_1(N-1) \notin P_{N-1}$ then there is an arc (v_j, v_{N-1}) entering v_{N-1} with $L-1 = l_j \in \mathscr{F}_j$. Hence, T_j contains an edge of level $L-1$. Now $\sigma(T_j) = L-1$ and $\{v_j, v_{N-1}\} \notin E_{T_j}$ by induction hypothesis. It follows that $\sigma(T) > L-1$, else the first chain deleted in a complete splitting of T using $L-1$ levels contains at least one edge of T_i and one of T_j different from $\{v_j, v_{N-1}\}$, which is clearly not possible.
— If $r_1(N-1) \leqslant \max\{f(N-1), r_3(N-1)\}$ then we may assume $r_1(N-1) > f(N-1)$, else $r_1(N-1) \notin P_{N-1}$ and this case has been studied above. Hence, $r_1(N-1) = r_3(N-1)$ and there are three arcs (v_i, v_{N-1}), (v_j, v_{N-1}) and

(v_k, v_{N-1}) of level $L-1$ entering v_{N-1}. By induction hypothesis, we have $\sigma(T_i) = \sigma(T_j) = \sigma(T_k) = L-1$. It follows that $\sigma(T) > L-1$, else the first chain deleted in a complete splitting of T using $L-1$ levels contains at least one edge of T_i, one of T_j and one of T_k, which is clearly not possible.

— If $r_1(N-1) = r_2(N-1) = 1$ then $L = 2$ and since T contains at least three edges incident to v_{N-1} it follows that $\sigma(T) \geqslant 2 = L$.

Case 2.2: $r_1(N-1) < L-1$. Since $l_{N-1} > L-1 > r_1(N-1)$ it follows from case (a) of the sequential step that $L-1 \notin P_{N-1}$. Consider any arc (v_i, v_{N-1}) entering v_{N-1} with $L-1 \in \mathscr{F}_i$. By induction hypothesis, we have $\sigma(T_i) = L-1$ and $\{v_i, v_{N-1}\} \notin E_{T_i}$.

— If there is a second arc (v_j, v_{N-1}) entering v_{N-1} with $L-1 \in \mathscr{F}_j$ then $\{v_i, v_{N-1}\} \notin E_{T_i}$ and $\sigma(T_j) = L-1$ by induction hypothesis. It follows that $\sigma(T) > L-1$, else the first chain deleted in a complete splitting of T using $L-1$ levels contains at least one edge of T_i different from $\{v_i, v_{N-1}\}$ and one edge of T_j different from $\{v_j, v_{N-1}\}$, which is clearly not possible.

— If (v_i, v_{N-1}) is the unique arc entering v_{N-1} with $L-1 \in \mathscr{F}_i$, then consider the vertex v_j in T_i with smallest index such that $L-1 \in \mathscr{F}_j$. Since the edges with level $L-1$ form a chain in T, all such edges are in T_j. No arc (v_k, v_j) entering v_j has $L-1 \in \mathscr{F}_k$ and it follows from the definitions given in the sequential step that $\mathscr{F}_j = \{L-1\}$ and $r_2(j) = L-1 > l_j = 1$. Now by induction hypothesis, we have $\sigma(T_j) = L-1$ and the arc out of v_j does not belong to E_{T_j}.

Consider the subtree T' containing v_j and obtained from T be deleting all arcs entering v_j. By applying algorithm Complete-Splitting to T', choosing v_N as root in the initialization step, one gets for each arc (v_s, v_t) in T' $(1 \leqslant s < N-1)$ exactly the same levels and sets of forbidden levels as for T, except that $L-1$ is no longer forbidden on the chain linking v_j to v_{N-1}. Hence, edge $\{v_{N-1}, v_N\}$ receives level $L-1$ instead of L and it follows from the induction hypothesis that $\sigma(T') = L-1$. Finally, $\sigma(T) > L-1$, else the first chain deleted in a complete splitting of T using $L-1$ levels contains at least one edge of T' and one edge of T_j but does not contain the arc out of v_j, which is clearly not possible. □

Proposition 3.9. *The above algorithm finds a complete splitting of T with $\sigma(T)$ levels in* $O(N \log N)$ *time.*

Proof. Observe first that computing a level l_i $(1 \leqslant i < N)$ for the arc out of v_i requires only information related to all arcs (v_j, v_i) entering v_i, i.e., labels L_j and sets of forbidden levels $\mathscr{F}_j$. These sets have length at most $\sigma(T)$. It follows from Corollary 3.4 that $\sigma(T) \in O(\log N)$. Thus, each level l_i and each set of forbidden levels $\mathscr{F}_i$ may be computed in $O(\log N)$ time multiplied by the number of predecessors of v_i. Moreover, computing $r_\alpha(i)$ and $f(i)$ may be done by listing the number of times a level is given to an arc entering v_i or forbidden on an arc entering v_i (in lists of length $O(\log N)$) and scanning these lists for computing $r_1(i)$, $r_2(i)$, $r_3(i)$ and $f(i)$. Again the time required is $O(\log N)$ times the number of predecessors of v_i.

Then considering the total computing work, and noting that the sum of the numbers of predecessors of all vertices is equal to $N - 1$, one finds that $O(N \log N)$ operations suffice. □

4. A sufficient condition

We state in this section a sufficient condition for the existence of a black-and-white coloring of T with x black and y white vertices.

Theorem 4.1. *Let $(S_1, \ldots, S_z)$ be a complete splitting of a tree T. If $x + y \leqslant N - z$ there exists a black-and-white coloring of T with x black and y white vertices.*

Proof. By induction on z. If $z = 1$ then T is a chain. The x vertices closest to one end of the chain may be colored in black and the y vertices closest to the other end in white.

Assume the result holds for $z < k$ and consider $z = k$. Let $C = (v_1, v_2, \ldots, v_p)$ be the chain induced by the edges of level 1. For each v_i $(1 \leqslant i < p)$ in C define v_i as the set of vertices in the subtree containing v_i and obtained from T by deleting edge $\{v_i, v_{i+1}\}$. Let V_p be the set of vertices in T and consider the smallest index j such that $|V_j| > x$ (which exists as $|V_p| = N > x$).

If $|V_j| = x + 1$, color in black all vertices of V_j except v_j and choose any y vertices in $V_p \setminus V_j$ to be colored in white. Otherwise, let $W_1, \ldots, W_q$ denote the sets of vertices in the subtrees containing no vertex of C and obtained from T by removing v_j. Consider the smallest index r such that $|V_{j-1}| + \sum_{i=1}^{r} |W_i| > x$ (where $|V_{j-1}| = 0$ if $j = 1$) which exists as $|V_{j-1}| + \sum_{i=1}^{q} |W_i| = |V_j| - 1 > x$.

If $|V_p| - (|V_{j-1}| + \sum_{i=1}^{r} |W_i|) > y$, choose x vertices in $V_{j-1} \cup W_1 \cup \cdots \cup W_r$ to be colored black and y vertices in $(V_p \setminus V_j) \cup W_{r+1} \cup \cdots \cup W_q$ to be colored white. Otherwise, color in black all vertices in $V_{j-1} \cup W_1 \cup \cdots \cup W_{r-1}$ and color in white all vertices of $(V_p \setminus V_j) \cup W_{r+1} \cup \cdots \cup W_q$. Let x' denote the total number of vertices colored black and y' the total number of vertices colored white. Since all vertices except v_j and those in W_r are now colored, we have $x' + y' = N - |W_r| - 1$. We use induction on the subtree T' induced by W_r with $\tilde{x} = x - x'$ and $\tilde{y} = y - y'$. The total number of vertices to be colored in T' is $\tilde{x} + \tilde{y} = (x + y) - (x' + y') \leqslant (N - z) - (N - |W_r| - 1) = |W_r| - (z - 1)$. Since the edges of T' can be deleted in $z - 1$ successive forest splittings, it follows that T' can be colored in black and white with $\tilde{x}$ black and $\tilde{y}$ white vertices. □

Observe that from Corollary 3.4 and Theorem 4.1 there is a black-and-white-coloring of T whenever $x + y \leqslant N - \log_3(N - 1) + 1$. This condition is a mild one.

Given a complete connected splitting $(S_1, \ldots, S_z)$ of a tree T, Theorem 4.1 leads to the following $O(N)$ algorithm for finding a black-and-white coloring of T, when condition $x + y \leqslant N - z$ holds.

```
Algorithm BW1($T$)
Step 1: Orient the edges of $T$ towards an extreme vertex of the chain induced by $S_1$.
        For each $v_i$ in $T$, let $V_i$ denote the set of vertices of the subtree containing $v_i$
        and obtained from $T$ by removing the arc going out of $v_i$. Using depth-first
        search compute the labels $n_i = |V_i|$.
        Set $T' = T$ and $l = 1$.
Step 2: Set $v$ equal to the root in $T'$. Set $b = 0$.
Step 3: As long as $b < x$ perform the following operations:
        (i) for each arc $(v_i, v)$ of level $> l$, in order of increasing indices, add $n_i$ to $b$;
        (ii) Let $(w, v)$ be the arc of level $l$ entering $v$. Set $v = w$ and add 1 to $b$.
Step 4: If the last increase of $b$ was made in (ii) (in which case $b = x$), consider the
        immediate successor $w$ of $v$. Color black all vertices of the subtree containing
        $w$ and obtained from $T'$ by deleting the arc $(v, w)$. Leave $v$ uncolored. Color
        white any $y$ remaining vertices and stop.
Step 5: If the last increase of $b$ was made in (i), consider the predecessor $v_i$ of $v$ at the
        last increase of $b$. It follows that $b \geqslant x$ with $b - n_i < x$. Let $F$ be the forest
        obtained by deleting vertex $v$. Color black all vertices of the connected
        components of $F$ containing the immediate successor of $v$ or an immediate
        predecessor $v_j$ of $v$ such that $j < i$ and $(v_j, v)$ is of level $> l$. Leave $v$ uncolored.
        • If $b = x$ color black all vertices of $V_i$, color white any $y$ remaining vertices
          and stop.
        • If $b > x$ color white at most $y$ and possibly all vertices of the connected
          components of $F$ containing an immediate predecessor $v_j$ of $v$ such that
          $j > i$ and $(v_j, v)$ is of level $> l$. If $y$ vertices have been colored white, color
          black any $x - (b - n_i)$ vertices of $V_i$ and stop. Otherwise substract $(b - n_i)$
          from $x$ and $N - (b + 1)$ from $y$. Set $T'$ equal to the subtree induced by $V_i$
          and go to Step 2.
```

Proposition 4.2. *Consider any complete connected splitting* $(S_1, \ldots, S_z)$ *of a tree* T. *If* $x + y \leqslant N - z$ *then* $BW1(T)$ *provides a black-and-white coloring of* T *in* $O(N)$ *time.*

Proof. Correctness follows from Theorem 4.1. For complexity, observe first that Step 1 is in $O(N)$ as it uses depth-first search. Then Steps 2–5 use the same vertex numbering and cardinalities n_i of subtrees induced by V_i as for T. Moreover, each n_i is considered at most once to be added to b. Hence, only $O(N)$ computations are made. □

Corollary 4.3. *If* $x + y \leqslant N - \sigma'(T)$ *a black-and-white coloring of* T *can be found in* $O(N)$ *time.*

Proof. From Propositions 3.5 and 3.6, a labelling corresponding to a complete connected splitting of T with $\sigma'(T)$ levels can be found in $O(N)$ time. Given this splitting, a black-and-white coloring can be found also in O(N) time from Theorem 4.1 □

Given a non-connected complete splitting $(S_1, \dots, S_z)$ of T, a slightly more complex algorithm named BW2 can be applied for finding a black-and-white coloring of T, when condition $x + y \leqslant N - z$ holds. It differs from BW1 in the first two steps.

Algorithm BW2(T)

```
Step 1: Set $T' = T$ and $l = 1$.
Step 2: Orient the edges of $T'$ towards an extreme vertex of the chain induced by $S_l$.
        For each vertex $v_i$ not in $S_l$ which precedes a vertex $v$ on $S_l$, denote $V_i$ the set of
        vertices of the subtree containing $v_i$ and obtained from $T'$ by removing the arc
        $(v_i, v)$. Using depth-first search compute the labels $n_i = |V_i|$.
        Set $v$ equal to the root in $T'$. Set $b = 0$.
Steps 3–5: As in BW1($T$).
```

Proposition 4.4. *Consider any complete splitting $(S_1 \dots, S_z)$ of a tree T.*

If $x + y \leqslant N - z$ then BW2 (T) provides a black-and-white coloring of T in O(Nz) *time.*

Proof. Correctness follows from Theorem 4.1. For complexity, observe that Steps 2–5 are in O(N). Since each step is applied at most z times, only O(Nz) computations are made. □

Corollary 4.5. *If $x + y \leqslant N - \sigma(T)$, a black-and-white coloring of T can be found in* O($N \log N$) *time.*

Proof. From Proposition 3.9, a labelling corresponding to a complete splitting of T with $\sigma(T)$ levels can be found in O($N \log N$) time. Given this splitting, a black-and-white coloring can be found in O($N\sigma(T)$) time from Theorem 4.1. But $\sigma(T) \in$ O(log N) from Corollary 3.4 and the result follows. □

To conclude, in order to find a black-and-white coloring of a tree T, the following steps (of increasing complexity) may be taken:

(i) Compute the connected splitting number $\sigma'(T)$ of T in O(N) time. If $x + y \leqslant N - \sigma'(T)$ compute a black-and-white coloring of T also in O(N) time and stop.

(ii) Compute the splitting number $\sigma(T)$ of T in O($N \log N$) time. If $x + y \leqslant N - \sigma(T)$ compute a black-and-white coloring of T in O($N \log N$) time and stop.

(iii) Use the dynamic programming algorithm to determine in O(N^3) time whether T admits a black-and-white coloring and obtain one if it is the case.

Acknowledgements

Work of the first author was done during a visit to École Polytechnique Fédérale de Lausanne, whose hospitality is gratefully acknowledged. This work was also supported by NSERC grant GP0105574 and FCAR grant 92EQ1048.

References

[1] C. Berge, Graphs and Hypergraphs (North-Holland, Amsterdam, 1973).
[2] C. Berge, private communication.
[3] M.R. Garey and D.S. Johnson, Computers and Intractibility: A Guide to the Theory of NP-Completeness (Freeman, New York, 1979).
[4] A.L. Liesman and D. Richards, Eds., Proc. Internat. Workshop on Broadcasting and Gossiping, 1990, Discrete Appl. Math. 53 (1994) 1–338.
[5] N. Quinodoz, Colorations partielles sous contraintes, Diploma Project, Swiss Federal Institute of Technology, Lausanne, 1993.