AUSTRALASIAN JOURNAL OF COMBINATORICS
Volume **39** (2007), Pages 115–127

Factoring logic functions using star-colorings

Tatjana Gerzen

Lehrstuhl II für Mathematik
RWTH Aachen
D-52062 Aachen
Germany
gerzen@math2.rwth-aachen.de

Bert Randerath

Institut für Informatik
Universität zu Köln
D-50969 Köln
Germany
randerath@informatik.uni-koeln.de

Abstract

In this paper we study the readnumber of light Boolean functions f by means of the star arboricity of a corresponding auxiliary graph ϕ_f. This approach enables us to establish the upper bound $k+1$ for the readnumber of light Boolean functions f with ϕ_f being a k-tree, and 5 if ϕ_f is planar. Here, the readnumber of a Boolean function f is the smallest value for the maximum number of occurrences of a literal in a factored form logically equivalent to f. Moreover, we observe that the star arboricity of a k-tree is at least k and give a characterization of the family of k-trees with star arboricity equal to k for $k = 2, 3$. Finally, the decision problem whether a graph has star arboricity 2 is shown to be solvable in polynomial time for the family of 2-trees, a subclass of 2-degenerate graphs for which this problem is known to be NP-complete.

1 Introduction

Minimizing Boolean circuits and functions is a fundamental problem in computer science. For applications like circuit layout, normal forms for Boolean functions are less important than logically equivalent factored forms with minimized variable dependency. The corresponding variable occurrence parameter for Boolean functions is called the readnumber. For the class of read-once Boolean functions, a special family of unate Boolean functions, there is a characterization via the cluster-clique graph due to Gurvich [7] and a combinatorial characterization due to Karchmer,

Linial, Newman, Saks and Wigderson [10]. In [5] Golumbic and Mintz present a fast algorithm for recognizing and factoring read-once functions. For more information regarding read-once functions the reader is referred also to [4], [2], [8] and [6]. A related problem is whether an arbitrary CNF formula can be expressed as in read-once or read-twice CNF. Both are coNP-hard, as shown by Naidu and Chandru [12]. They also design a polynomial time algorithm for the problem of recognizing read-twice DNFs. Our work is motivated by the following conjecture of Golumbic[1]:

The readnumber of each unate Boolean function with a k-tree as corresponding cluster-clique graph is at most $2k-1$.

In this paper we obtain a partial solution to this problem by incorporating a connection between the readnumber of a light Boolean function and the star arboricity of the corresponding cluster-clique graph, which allows us to prove that the upper bound of the readnumber of each light Boolean function with a k-tree as cluster-clique graph is $k+1$ and 5 if the cluster-clique graph is planar.
It is known [9] that the star arboricity of a k-tree is at most $k+1$. We show that the star arboricity of a k-tree is at least k and give a characterization of the family of 2-trees with star arboricity equal to 2. We also give a characterization of 3-trees with star arboricity equal to 3. In [9] Hakimi, Mitchem and Schmeichel show that the decision problem whether a graph has star arboricity 2 is NP-complete, even for 2-degenerate graphs. We show that this problem is solvable in polynomial time for the family of 2-trees, a subclass of 2-degenerate graphs.

2 Preliminaries and Terminology

Standard forms for representing a Boolean function are the sum of products (*DNF*) and the product of sums (*CNF*). For simplicity we only consider Boolean functions without duplicate clauses. We say that a DNF of a Boolean function is simplified (*SDNF*), if the DNF is the sum of prime implicants of the function. A *prime implicant* is a minimal product of literals whose truth implies the truth of the function and whose removal from the formula would change the function. We call the prime implicants of a SDNF *clusters*. The *base* of a Boolean function f is the set of variables of its SDNF F, it is written $[F]$. A subcluster of a cluster C is a product of variables of $[C]$. For example the formula $F = abc+abh+adc+adh+cb+cd+cbh+cdh$ is in DNF. The CNF of F is $F = (a+c)(b+d)(c+h)$ and a SDNF of F is $F = cb+abh+cd+adh$. $[F] = \{a, b, c, d, h\}$ is the base of F, the clusters of F are cb, abh, cd, adh are and ab, ah, bh are the subclusters of the cluster abh.
A *unate* function is a Boolean function represented by a formula such that each variable appears either in the positive or in the negative form throughout the formula. E.g. read-once Boolean function are unate. By renaming of variables of a unate function which appear in the negative form throughout the formula, we can easily obtain a formula only containing variables. Hence, we will consider in the following only *positive* functions. Moreover, the SDNF of a positive Boolean function has a

[1] Stated on the 7^{th} Intern. Symp. on Artifical Intelligence and Math. (Fort Lauderdale, 2002)

unique representation.

Let f be a positive Boolean function and F its SDNF with base $[F] = \{a_1, ..., a_r\}$. We now associate a graph ϕ_F to f by putting $\phi_F = ([F], E)$ with $(a_{i_1}, a_{i_2}) \in E$ if and only if a_{i_1} and a_{i_2} are variables of the same cluster of F. We call this graph the *cluster-clique graph* of f. Vice versa for every graph $G = (V, E)$ with vertex set $V = \{a_1, a_2, ..., a_n\}$ and edge set E we can associate a *canonical* (positive) 2-DNF Boolean function f with variable set V and f is the sum of products $a_i a_j$, where $(a_i, a_j) \in E$.

A *heavy cluster* of a Boolean function is a cluster with more than two variables. We call a positive Boolean function f *light*, if every heavy cluster C in the SDNF F of f has the property that for all $a, b \in [C]$ the product ab is not a subcluster of a cluster of $F - C$, where $F - C$ is the sum of all clusters of F except C. For example the functions $f = abc + cdr$ and $h = ab + abc$ are light but the function $g = abc + bcdr$ is not a light Boolean function. Denote by $\mathcal{B}$ the set of all Boolean functions. For an integer k we define *Read-k* as a subset of $\mathcal{B}$ such that each variable of an $f \in$ *Read-k* appears in f at most k times, that is *Read-k* $= \{f \in \mathcal{B} |$ each variable of f appears in f at most k times$\}$.

The *readnumber* $Readn(f)$ of a positive Boolean function f is the smallest integer k such that f has a factored form in which each variable appears at most k times, that is

$$Readn(f) = \min\{k | \exists_{g \approx f} : g \in \text{Read-}k\}.$$

For all notations from propositional logic not defined here the reader is referred to [2], [11] and [14].

For an integer k a graph G is *k-degenerate* if there exists an ordering $v_1, v_2, ..., v_n$ of the vertices of G such that for every i the vertex v_i has a degree at most k in the graph $< v_1, ..., v_i >$. Here $< v_1, ..., v_i >$ denotes the subgraph of G induced by vertices $v_1, ..., v_i$. Furthermore, a k-degenerate graph G with vertex ordering $v_1, v_2, ..., v_n$ is a *k-tree* if the vertices $v_1, ..., v_k$ induce a clique and for every $i > k$ the vertex v_i has the degree k in $< v_1, ..., v_i >$ and the graph induced by $N(v_i) \cap \{v_1, ..., v_{i-1}\}$ is complete. We also give a recursive definition of k-trees: A clique with k vertices is a k-tree. If $T = (V, E)$ is a k-tree and C is a clique of T with k vertices and $x \notin V$, then $T' = (V \cup \{x\}, E \cup \{cx | c \in C\})$ is a k-tree.

A *star* is a tree with at most one vertex with degree larger than 1. A *star forest* is a forest whose components are stars. The *star arboricity* $st(G)$ of a graph G is the minimum number of edge-disjoint star forests in G whose union covers all edges of G. We call G k-star-colorable if $st(G) \leq k$. For terminology not defined we refer to [13].

Let f be a positive Boolean function such that ϕ_F is a 1-tree, i.e. a tree. In Section 3 we will prove that $Readn(f) \leq 2$ by using well known results about star-coloring of k-trees. Here we present a concise proof of this claim without using the concept of star-colorings.

Lemma 1 *Let F be the SDNF of a positive Boolean function f such that ϕ_F is a tree. Then $Readn(f) \leq 2$.*

PROOF: Let $p = k_0...k_s$ be a longest path in ϕ_F. Obviously, k_0 and k_s are leaves of ϕ_F. Let $s \geq 3$ (in case of $s \leq 2$ the result is obviously true). Then there is only one vertex $v \in N(k_1) = \{k_0, k_2, v_1, ..., v_{n+2}\}$ such that v is not a leaf. (Else we can easily find a path in ϕ_F which is longer than p.) Thus, we have $v = k_2$. We now consider the Boolean function F^1 defined by

$$f = k_1(k_0 + v_1 + ... + v_{n+2}) + F^{(1)}. \qquad (*)$$

The cluster-clique graph $T^1 = \phi_F \setminus \{k_0, v_1, ..., v_{n+2}\}$ of F^1 is a tree in which the vertex k_1 is a leaf. Therefore, k_1 appears in F^1 only once and, consequently, in the presentation $(*)$ of f exactly twice. Now we apply this procedure recursively until we obtain finally a tree T^m being a star. Observe now that in the corresponding representation of f each variable occurs at most twice. So, $Readn(f) \leq 2$. ■

3 Star-Colorings and Boolean Functions

In this section we establish a relation between the readnumber of a light Boolean function f and the star arboricity of the clique-cluster graph of f.

Theorem 1 *Let F be the SDNF of a light Boolean function f. Then*

$$Readn(f) \leq st(\phi_F).$$

The proof of Theorem 1 follows from Proposition 1 and Proposition 2.

Proposition 1 *Let f be a positive Boolean function such that its SDNF F is a 2-DNF. Then $Readn(f) \leq st(\phi_F)$.*

PROOF: Let $st(\phi_F) = k$. Denote the star forests of a k-star-coloring of ϕ_F by $S_1, \ldots, S_k$. For each i denote by $S_{i1}, \ldots, S_{in_i}$ the stars of the star forest S_i, let a_{ij} be the center of S_{ij}.
Note that there is a 1-1 correspondence between the edge set $E(\phi_F)$ and the cluster set of F (because F has only 2-clusters). Hence

$$f = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \left(a_{ij} \sum_{b \in [S_{ij}] \setminus \{a_{ij}\}} b \right). \qquad (*)$$

The sum $\sum_{j=1}^{n_i}(a_{ij} \sum_{b \in [S_{ij}] \setminus \{a_{ij}\}} b)$ represents the star forest S_i. Thus, each variable of the presentation $(*)$ of f occurs at most k times. Therefore, $Readn(f) \leq k = st(\phi_F)$. ■

We are now able to present an alternative proof of Lemma 1: In [1] Algor and Alon proved that star arboricity of a tree is at most 2. Now let F be the SDNF of a positive Boolean function f such that ϕ_F is a 1-tree. Since f is obviously a light Boolean function, we can apply Theorem 1. Thus, we have $Readn(f) \leq st(\phi_F) \leq 2$.

Proposition 2 *Let F be the SDNF of a light Boolean function f and let f' be the canonical 2-DNF Boolean function of $G = \phi_F$. Then $Readn(f) \leq Readn(f')$.*

PROOF: We proof this result inductively on the number of heavy clusters of a light Boolean function F in SDNF. Since every light Boolean function f without heavy clusters forms a 2-DNF, we deduce $f = f'$. Thus $Readn(f) = Readn(f')$.
Let $n+1$ be the number of heavy clusters of F and C be a heavy cluster of F. Define

$$\tilde{F} = F - C + \sum_{\substack{a,b \in [C] \\ a \neq b}} ab. \qquad (*)$$

Observe that by the formula $(*)$ function $\tilde{F}$ is given in SDNF since F is given in SDNF of a light Boolean function f. Obviously, $\phi_{\tilde{F}} = \phi_F$. Therefore, $\tilde{f}'$, the canonical 2-DNF of $\phi_{\tilde{F}}$, is identical to f', the canonical 2-DNF of ϕ_F. Hence $Readn(\tilde{f}') = Readn(f')$ and by inspection $Readn(\tilde{f}) \geq Readn(f)$, where $\tilde{F}$ is the SDNF of a light Boolean function $\tilde{f}$ with n heavy clusters. By induction we also obtain $Readn(\tilde{f}') \geq Readn(\tilde{f})$. Finally, we thus derive $Readn(f') \geq Readn(f)$. ■

Remark: The following example demonstrates that the inequality in Proposition 2 (and Theorem 1) can not be replaced by an equality. Let f be the light Boolean function with SDNF $F = abcd$. Then the canonical 2-DNF f' of $G = \phi_F$ is given by $f' = ab+ac+ad+bc+cd+bd = (a+d)(b+c)+ad+bc$. The graph $G = \phi_F$ is a 4-clique. It is easy to see that $Readn(f) = 1$, $Readn(f') = 2$, and $st(\phi_F) = st(G) = 3$. Hence f' is moreover an example of a positive 2-DNF showing that likewise in Proposition 1 the inequality can not be replaced by an equality.

Remark: The constraint on a Boolean function f to be light in Proposition 2 cannot be dropped. Consider for example the Boolean function f with SDNF F given by

$$\begin{aligned} F = {} & abc + abd + abe + abf + acd + ace + acf + ade + adf + aef \\ & + bcd + bce + bcf + bde + bdf + bef + cde + cdf + cef + def. \end{aligned}$$

The canonical 2-DNF f' of $G = \phi_F$ is given by
$f' = ab + ad + ae + ag + ac + bc + bd + bc + be + bg + cd + cg + ce + de + dg + eg$. It is $f = (e+g)(d(a+b+c)) + eg(a+b+c+d) + (d+e+g)(c(a+b)) + ab(c+d+e+g)$ and $f' = (a+b+c+d)(e+g) + eg + (a+b)(c+d) + cd + ab$, so it is easy to see that the readnumber of f is equal to 4 and the readnumber of f' is 3.

We will now establish an upper bound for the readnumber of certain Boolean functions such that their corresponding cluster-clique graph is a k-tree. In [9] S.L. Hakim, J. Mitchem and E. Schmeichel developed a method that allows to estimate the star arboricity of many graph classes. In order to present their results we have to clarify additional terminology.

Let $C = (E_1, E_2, ..., E_k)$ be a k-star-coloring of a graph G, where E_j denotes the set of edges with color j. A vertex v is said to be a *good vertex*, respectively *center*, under C if there is at most one, respectively exactly one, index j such that E_j contains two or

more edges incident to v. In the latter case j is called the color of the center v under C. A star-coloring C of G is called *strict* if each edge joining two centers has the color of one of the two centers under C. A vertex-coloring D of G is called *unicyclic* if every pair of color classes under D induces a graph such that each component has at most one cycle.

Theorem 2 *[9] Let k be an integer. If G has an unicyclic k-vertex-coloring, then G has a strict k-star-coloring in which each vertex is good.*

And as corollary of Theorem 2

Corollary 1 *[9] Let G be a k-tree for an integer k. Then $st(G) \leq k+1$.*

Theorem 1 and Corollary 1 easily imply the next theorem. Note that this result forms a partial solution of Golumbic's conjecture.

Theorem 3 *Let f be a light Boolean function such that its corresponding cluster-clique graph is a k-tree. Then*

$$Readn(f) \leq k+1.$$

In [9] Hakim, Mitchem and Schmeichel showed that every planar graph has a 5-star-coloring. Analogously now Theorem 1 ensures that the readnumber of a light Boolean function with a planar graph as cluster-clique graph is at most 5.

4 Star-Colorings and k-Trees

Each k-tree G with more than k vertices contains a $(k+1)$-clique and, obviously, exact k colors (induction on k) are needed for a star-coloring of the $(k+1)$-clique. Hence we can refine Corollary 1.

Corollary 2 *Let G be a k-tree with more than k vertices. Then $k \leq st(G) \leq k+1$.*

Now we show the sharpness of the upper bound in the Corollary 2 for $k = 2$ and $k = 3$. Moreover, we give a characterization for 2-trees to be 2-star-colorable and for 3-trees to be 3-star-colorable.

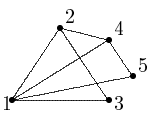


Figure 1: A 2-tree T with $st(T) = 3$.

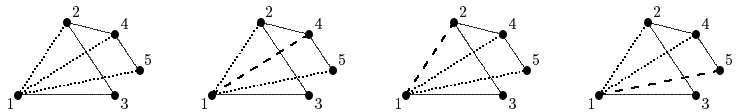


Figure 2:

Example 1 *Let T be the graph shown in Figure 1. Then we have $st(T) = 3$. Since, otherwise we try to color T with two colors*

red, represented by .....

and blue, represented by - - - - - - -

The degree of vertex 1 is equal to four, thus we have to color two of the edges (1,2), (1,4) and (1,5) with the same color. Therefore, there are four possible cases as depicted in the Figure 2. We have to color blue the edges (2,4) and (4,5) in the cases (a) and (b); the edge (4,5) in the case (c) and the edge (2,4) in the case (d) (Figure 3). Now it is easy to see that it is not possible to star-color T with two colors.

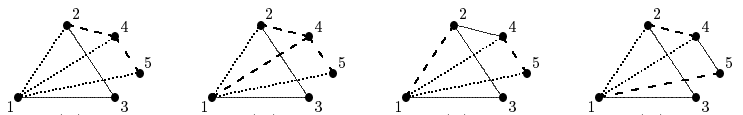


Figure 3:

Example 2 *Let G be the 3-tree as depicted in the Figure 4. Then the star arboricity of G is equal to 4.*

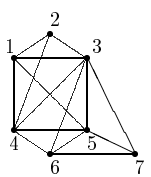


Figure 4: A 3-tree G with $st(G) = 4$.

Definition 1 *For an integer r we call a graph T_r an r-fence-graph if there are $v_1, v_2 \in V(T_r)$ such that $V(T_r) = \{v_1, ..., v_{r+2}\}$ and*

$$E(T_r) = \{(v_1, v_2), (v_1, v_i), (v_2, v_i) \mid i = 3, ..., r + 2\}.$$

We call the vertices v_1 and v_2 the special vertices of T_r. Figure 5 provides an example of a 5-fence-graph.

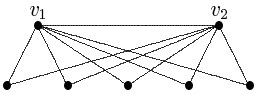


Figure 5: A 5-fence-graph.

The following theorem provides a characterization of 2-trees with star arboricity 2.

Theorem 4 *Let T be a 2-tree with more than 2 vertices. Then $st(T) = 2$ if T is isomorphic to a fence-graph, else $st(T) = 3$. (For a 2-tree with only two vertices the star arboricity is equal to 1.)*

PROOF: Clearly, $st(T) = 2$ and T is a fence-graph if $|V(T)| = 3$ or $|V(T)| = 4$.
Let T be isomorphic to a fence-graph and v_1 and v_2 be the special vertices of T. Color all with the vertex v_1 incident edges of T with color A and all with v_2 incident edges (except (v_1, v_2)) with color B. Thus we have $st(T) = 2$.
Let T be a 2-tree with $|V(T)| \geq 5$ which is not isomorphic to a fence-graph. Since T is a 2-tree with more than 3 vertices, T contains a 2-fence-graph T_2 as a subgraph. (Let $(v_1v_2v_3)$ be the start clique of T. Two of the vertices v_1, v_2 and v_3 must be adjacent to the fourth vertex v_4. Let v_1 and v_2 be these vertices. Then v_1 and v_2 are the special vertices of $T_2 = \{E = \{v_1, v_2, v_3, v_4\}, V = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_1, v_4), (v_2, v_4)\}\}$ and $T_2 \subset T$.)
Since T is not isomorphic to a fence-graph and T is a 2-tree, T must have at least one edge $e \notin T_2$ incident with one of the vertices v_3 and v_4. But the A-B-coloring of $T_2 \subset T$ considered above is, obviously, the only possible 2-star-coloring of T_2. Therefore, we cannot color the edge e with A or B. Thus, $st(T) \geq 3$ and the claim follows with Corollary 1. ■

Another characterization is the following corollary of Theorem 4.

Corollary 3 *Let T be a 2-tree with more than 2 vertices. Then $st(T) = 2$ if and only if T contains no (not necessary induced) path of length ≥ 5 and T is not isomorphic to the graph shown in the Figure 6.*

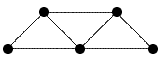


Figure 6:

Hakim, Mitchem and Schmeichel studied in [9] the complexity of the 2-star-colorability problem:

INSTANCE: graph G.
QUESTION: $st(G) \leq 2$?

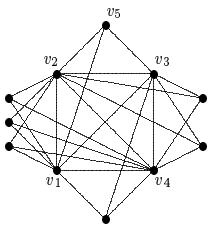
They showed that this decision problem is NP-complete, even for the class of 2-degenerate graphs. For the family of 2-trees, a subclass of 2-degenerate graphs, we have the following theorem, which can easily be derived from our characterization (Theorem 4).

Theorem 5 *2-star-colorability problem can be solved in polynomial time for the class of 2-trees.* ■

We also give a characterization of 3-trees with star arboricity equal to 3.

Definition 2 *For an integer n we call a graph T_n an n-one-point-graph if there are vertices $v_1, v_2, ..., v_{3+n} \in V(T)$ such that*

1. *v_1, v_2, v_3 form a clique and*
2. *$\forall_{i=1,...,n} : N_{<v_1,...,v_{n+3}>}(v_{3+i}) = \{v_1, v_2, v_3\}$ and*
3. $$\exists_{j\in\{4,...,3+n\}} \forall_{v\in T_n \setminus \{v_1,...,v_{3+n}\}} : N(v) = \begin{cases} \{v_j, v_1, v_3\} \text{ or} \\ \{v_j, v_3, v_2\} \text{ or} \\ \{v_j, v_1, v_2\}. \end{cases}$$



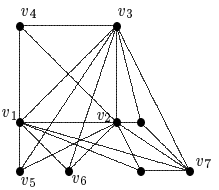


Figure 7: A 2-one-point-graph and a 4-one-point-graph.

Theorem 6 *Let T be a 3-tree with more than 3 vertices. Then $st(T) = 3$ if T is isomorphic to an one-point-graph, else $st(T) = 4$. (For a 3-tree with only 3 vertices the star arboricity is equal to 2.)*

Proof: Let T be a 3-tree with more than 3 vertices. Let $v_1, ..., v_4$ be the first four vertices of T. If $|V(T)| = 4$, then T is a 4-clique and so $st(T) = 3$. If $|V(T)| = 5$, then Figure 8 shows all possibilities to 3-star-color T. We use the colors

green, represented by .....

red, represented by .....

and blue, represented by - - - - - - - - - -

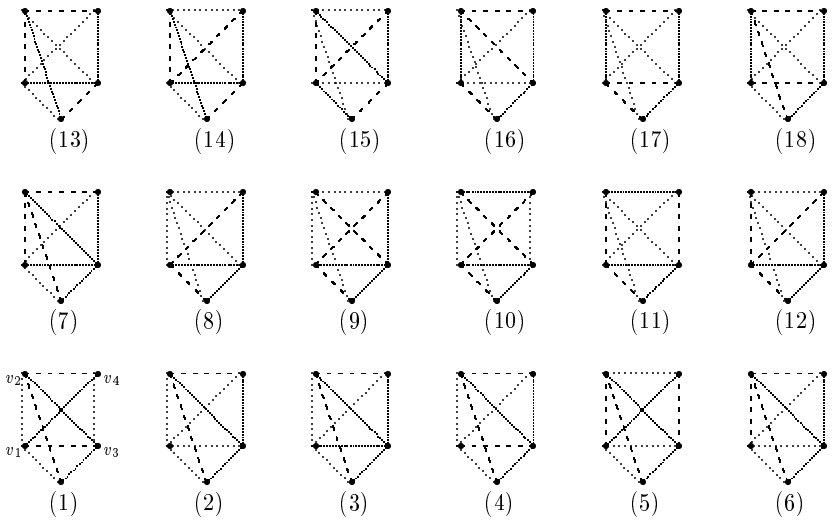


Figure 8:

If $|V(T)| > 5$, then let $N_{<v_1,\dots,v_5>}(v_5) = \{v_1, v_2, v_3\}$ without loss of generality.
If $N_{<v_1,\dots,v_5,v>}(v) = N_{<v_1,\dots,v_5>}(v_5)$ for a vertex $v \in V(T)$, we can choose arbitrarily a way from those shown in Figure 8 to color $< v_1, ..., v_5 >$. Then we can color the edges that are incident with v in the same way as those incident with v_5. Therefore we assume for the sixth vertex $v_6 \in V(T)$ that $N_{<v_1,\dots,v_6>}(v_6) \neq N_{<v_1,\dots,v_5>}(v_5)$.
For convenience we will using following notation: "$color < v_1, ..., v_5 >= figure8(j)$" for "color the graph $< v_1, ..., v_5 >$ as shown in Figure 8 part (j)" (for $j = 1, ..., 18$).
First we state three claims, which can easily be verified:

Claim 1. If $v_4 \in N(v_6)$ and
$color\langle v_1, ..., v_5\rangle \notin \{figure8(1), figure8(5), figure8(10), figure8(11), figure8(17)\}$, then it is not possible to 3-star-color $\langle v_1, ..., v_6\rangle$.

Claim 2. If $v_5 \in N(v_6)$ and
$color\langle v_1, ..., v_5\rangle \notin \{figure8(13), figure8(14), figure8(15), figure8(16)\}$, then it is not possible to 3-star-color $\langle v_1, ..., v_6\rangle$.

Claim 3. The 3-star-colorings (13), (14), (15) and (16) shown in figure 8 are equal to the star-colorings (10), (11), (1) and (17) shown in figure 8 if we replace the vertices v_4 and v_5. The coloring $figure8(5)$ is equal to the coloring $figure8(14)$ by exchanging vertex v_4 with vertex v_5 and v_1 with v_3.

Claims 1 and 2 imply that, if there are vertices $p, q \in V(T) \setminus \{v_1, ..., v_5\}$ such that $v_4 \in N(p)$, $v_5 \in N(q)$, then it is not possible to 3-star-color T. More general if there are vertices $v_n, v_m \in V(T)$ and $v, w \in V(T) \setminus \{v_1, ..., v_n, v_m\}$ such that $N_{<v_1,\dots,v_n>}(v_n) = N_{<v_1,\dots,v_m>}(v_m)$ and $v_n \in N(v)$, $v_m \in N(w)$, then it is not possible to 3-star-color T.
By Claim 3 it is sufficient to consider for the vertex v_6 only the case $v_5 \in N(v_6)$.

The following subcases are possible

a) $N_{<v_1,...,v_6>}(v_6) = \{v_1, v_2, v_5\}$

b) $N_{<v_1,...,v_6>}(v_6) = \{v_1, v_3, v_5\}$

c) $N_{<v_1,...,v_6>}(v_6) = \{v_2, v_3, v_5\}$.

Case a) We can 3-star-color the graph $< v_1, ..., v_6 >$ in the following ways:

- $color < v_1, ..., v_5 >= figure8(13)$ and color
  (v_1, v_6) green and $(v_2, v_6)((v_5, v_6))$ red (blue) or vice-versa.
- $color < v_1, ..., v_5 >= figure8(14)$ and color
  $\begin{cases}(v_2, v_6) \text{ green, } (v_1, v_6)\text{blue and } (v_5, v_6)\text{red or} \\ (v_2, v_6) \text{ red and } (v_1, v_6)((v_5, v_6)) \text{ green (blue) or vice-versa.}\end{cases}$
- $color < v_1, ..., v_5 >= figure8(15)$ and color
  (v_2, v_6) green and $(v_1, v_6)((v_5, v_6))$ red (blue) or vice-versa.
- $color < v_1, ..., v_5 >= figure8(16)$ and color
  $\begin{cases}(v_5, v_6) \text{ green and } (v_1, v_6)((v_2, v_6)) \text{ red (blue) or vice-versa or} \\ (v_5, v_6) \text{ blue and } (v_1, v_6)((v_2, v_6)) \text{ green (red) or vice-versa.}\end{cases}$

The are no other possibilities to 3-star-color $< v_1, ..., v_6 >$ in this case.

Case b) In this case we have the following possibilities:

- $color < v_1, ..., v_5 >= figure8(13)$ and color
  (v_1, v_6) green and $(v_3, v_6)((v_5, v_6))$ red (blue) or vice-versa.
- $color < v_1, ..., v_5 >= figure8(14)$ and color
  $\begin{cases}(v_1, v_6) \text{ blue, } (v_3, v_6)\text{red and } (v_5, v_6)\text{green or} \\ (v_1, v_6) \text{ green and } (v_3, v_6)((v_5, v_6)) \text{ red (blue) or vice-versa.}\end{cases}$
- $color < v_1, ..., v_5 >= figure8(15)$ and color
  $\begin{cases}(v_5, v_6) \text{ red and } (v_1, v_6)((v_3, v_6)) \text{ green (blue) or vice-versa or} \\ (v_5, v_6) \text{ blue and } (v_1, v_6)((v_3, v_6)) \text{ green (red) or vice-versa.}\end{cases}$
- $color < v_1, ..., v_5 >= figure8(16)$ and color
  (v_3, v_6) red and $(v_1, v_6)((v_5, v_6))$ green (blue) or vice-versa.

Case 1. c) In this case we have the following possibilities:

- $color < v_1, ..., v_5 >= figure8(13)$ and color
  $\begin{cases}(v_5, v_6) \text{ red and } (v_2, v_6)((v_3, v_6)) \text{ green (blue) or vice-versa or} \\ (v_5, v_6) \text{ blue and } (v_2, v_6)((v_3, v_6)) \text{ green (red) or vice-versa.}\end{cases}$

- $color < v_1, ..., v_5 >= figure8(14)$ and color
  $\begin{cases} (v_5, v_6) \text{ green, } (v_2, v_6) \text{red and } (v_3, v_6) \text{blue or} \\ (v_2, v_6) \text{ green and } (v_3, v_6)((v_5, v_6)) \text{ red (blue) or vice-versa.} \end{cases}$

- $color < v_1, ..., v_5 >= figure8(15)$ and color
  (v_2, v_6) green and $(v_3, v_6)((v_5, v_6))$ red (blue) or vice-versa.

- $color < v_1, ..., v_5 >= figure8(16)$ and color
  (v_3, v_6) red and $(v_2, v_6)((v_5, v_6))$ green (blue) or vice-versa.

Claim 4: In all of the above cases, we have to use all three colors to color the edges incident to v_6 and each of this three stars has a center different to v_6. Therefore, it is not possible to 3-star-color a 3-tree T if T has a vertex $v_r \notin \{v_1, ..., v_6\}$ such that $(v_6, v_r) \in E(T)$.
It is noteworthy that a 3-tree with at least one vertex of type a), b) or c) is 3-star-colorable (Figure 9). This finally completes our proof. ■

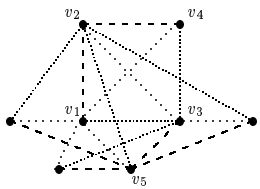


Figure 9: A 3-star-coloring of a 3-tree with one vertex of type a), b), and c).

It would be interesting to give a characterization of k-trees with star arboricity equal to k for an arbitrary k.

References

[1] I. Algor and N. Alon, The star arboricity of graphs, *Discrete Math.* 75 (1989), 11–22.

[2] Y. Crama and P.L. Hammer, Boolean functions: Theory, algorithms, and applications, Springer, 2006.

[3] T. Eiter. Exact transversal hypergraphs and application to Boolean μ-functions, *J. Symbolic Comput.* 17 (1994), 215–225.

[4] T. Eiter, Generating Boolean μ-expressions, *Acta Informatica* 32 (1995), 171–187.

[5] M.C. Golumbic, A. Mintz and U. Rotics, Factoring and Recognition of Read-Once-Functions using Cographs and Normality, DAC 2001, 109–114.

[6] M.C. Golumbic, A. Mintz and U. Rotics, Factoring and recognition of read-once-functions using cographs and normality and the readability of functions associated with partial k-trees, *Discrete Appl. Math.* 154 (2006), 1465–1477.

[7] V.A. Gurvich. On repetition-free Boolean functions, *Uspechi Mat. Nauk* 32 (1977), no. 1 (193), 183–184 (Russian).

[8] V.A. Gurvich, Criteria for repetition-freeness of functions in algebra of logic, *Sovjet. Math. Dokl.* 43 (1991), no. 3, 721–726.

[9] S.L. Hakim, J. Mitchem and E. Schmeichel, Star arboricity of graphs, *Discrete Math.* 149 (1996), 93–98.

[10] M. Karchmer, N. Linial, I. Newman, M. Saks and A. Wigderson, Combinatorial characterization of read-once formulae, *Discrete Math.* 114 (1993), no. 1-3, 275–282.

[11] H. Kleine Bning and T. Lettmann, Aussagenlogik: Deduktion und Algorithmen, B.G. Teubner, Stuttgart, 1994.

[12] S.R. Naidu and V. Chandru. On recognition of read-once and read-twice formulae, Manuscript 2000.

[13] D.B. West, *Introduction to graph theory*, Prentice Hall Inc., NJ, 1996.

[14] I. Wegener, The Complexity of Boolean Functions, Wiley, 2000. Available via `http://ls2-www.cs.uni-dortmund.de/~wegener`.

(Received 4 Aug 2006)