A sufficient condition for the bicolorability of a hypergraph

David Défossez

Laboratoire Leibniz-Imag, 46 avenue Félix Viallet, 38031 Grenoble Cédex, France

Received 23 July 2004; received in revised form 1 December 2005; accepted 27 September 2006
Available online 6 May 2007

Abstract

In this note we prove a long-standing conjecture of Sterboul [P. Duchet, Hypergraphs, in: R. Graham, M. Grötschel, L. Lovász (Eds.), Handbook of Combinatorics, 1995, pp. 381–432 (Chapter 7)], which states that a hypergraph is bicolorable provided it does not contain a specific kind of odd cycle. This is currently the strongest result of its kind, improving on results by Berge [Graphs and Hypergraphs, North-Holland, American Elsevier, Amsterdam, 1973] and Fournier and Las Vergnas [Une classe d’hypergraphes bichromatiques II, Discrete Math. 7 (1974) 99–106; A class of bichromatic hypergraphs, Ann. Discrete Math. 21, in: C. Berge, V. Chvátal (Eds.), Topics on Perfect Graphs, 1984, pp. 21–27].

A hypergraph is a pair, \( H = (V, \mathcal{E}) \), where the elements of \( V \) are the vertices, and the elements of \( \mathcal{E} \) are subsets of \( V \) and are called the edges. A function \( c: V \to \{1, 2\} \) is called a bipartition of \( V \), and for \( x \in V \), we call \( c(x) \) the color of \( x \). If there exists \( e \in \mathcal{E} \) with \( |e| \geq 2 \) in which only one color occurs, then \( e \) is said to be monochromatic. If \( c \) is such that there exists no monochromatic edge, then \( c \) is called a bicoloration of \( H \). If a hypergraph admits a bicoloration then we say that it is bicolorable.

The question of when a hypergraph admits a bicoloration (also known as possessing “Property B”) is a classical one in combinatorics, with research spanning extremal problems [1], algorithmic recognizability [7], and structural obstructions [4]. Here we prove a structural result conjectured by Sterboul [4] that subsumes the work of previous authors [2,5,6].

A sequence \( (x_1, e_1, x_2, \ldots, e_k, x_1) \), where the \( e_i \)'s are distinct edges, the \( x_i \)'s are distinct vertices, and \( k \geq 3 \), is said to be a cycle if \( x_i \in e_{i-1} \cap e_i \) for \( i = 2, \ldots, k \) and \( x_1 \in e_1 \cap e_k \). A cycle is said to be odd if it has an odd number of edges.

An odd cycle \( (x_1, e_1, x_2, \ldots, e_k, x_1) \) such that two non-consecutive edges are disjoint and \( |e_i \cap e_{i+1}| = 1 \) for \( i = 1, 2, \ldots, k - 1 \), is called a anti-Sterboul cycle. If a hypergraph \( H \) has no anti-Sterboul cycle, it is said to be a Sterboul hypergraph.

Then we can word Sterboul’s conjecture as follows:

**Theorem.** If \( H \) is a Sterboul hypergraph, then \( H \) is bicolorable.

E-mail addresses: david.defossez@imag.fr, d_defossez@yahoo.fr.
Observe that as it was noticed in [5,6], we have to allow for the possibility that \(|e_k \cap e_1| > 1\) in the definition of an anti-Sterboul cycle. Indeed the complete \(r\)-uniform hypergraph on \(2r - 1\) vertices has no anti-Sterboul cycle in which \(|e_k \cap e_1| < r - 1\), but is not bicolorable.

**Proof of Theorem.** The proof works by induction on the number of edges.

When the hypergraph has no edge, the theorem clearly holds.

The general step assumes that we have a hypergraph \(H = (V, \mathcal{E})\) that is Sterboul, and \(e_0 \in \mathcal{E}\) such that \(H \setminus e_0 = (V, \mathcal{E} \setminus e_0)\) has a bicoloration \(c: V \to \{1, 2\}\).

We can assume that \(e_0\) has size at least 2, and that \(c\) leaves \(e_0\) monochromatic, or else we have nothing to do.

Now we use the following algorithm to transform the bipartition \(c\) into a bicoloration of \(H\). The algorithm successively switches the colors of some vertices of \(H\) that are contained in a monochromatic edge in the current bipartition. It constructs a tree \(T = (V', E)\) and a mapping \(g: V' \to \mathcal{E}\) that keep track of the running of the algorithm: the vertices of \(T\) are those of \(H\) whose colors were switched (thus \(V' \subset V\)), and \(g\) associates a vertex with the monochromatic edge that caused its color switch.

The vertices are chosen with a DFS (depth-first search) method, and to do so the algorithm uses a LIFO (last in first out) list \(\mathcal{P}\) (equivalently called a stack) that contains a subset of vertices whose colors have been switched, and which form a cover of the existing monochromatic edges at each iteration. \(\text{top}(\mathcal{P})\) returns the last vertex entered in \(\mathcal{P}\), \(\text{pop}(\mathcal{P})\) removes \(\text{top}(\mathcal{P})\) out of \(\mathcal{P}\), and \(\text{push}(x, \mathcal{P})\) enters a new vertex \(x\) in \(\mathcal{P}\).

**Algorithm:**

**INPUT:** A hypergraph \(H = (V, \mathcal{E})\) and a bipartition \(c\) such that \(e_0 \in \mathcal{E}\) is the only monochromatic edge.

**OUTPUT:** A bicoloration of \(H\) or an Error message only if \(H\) is not a Sterboul hypergraph.

1. let \(x_0 \in e_0\);
2. \(V' := \{x_0\};\ E := \emptyset;\)
3. \(g(x_0) := e_0;\)
4. switch \(c(x_0);\)
5. \(\text{push}(x_0, \mathcal{P});\)
6. While \(\mathcal{P} \neq \emptyset\) do
   1. let \(v := \text{top}(\mathcal{P});\)
   2. If \(\exists e \in \mathcal{E}, |e| \geq 2,\monochromatic\ such\ that\ v \in e\ then\)
      1. If \(e \subset V'\ then\)
         1. return Error;
      2. else
         1. let \(w \in e \setminus V';\)
         2. \(V' := V' \cup \{w\};\ E := E \cup \{(vw)\};\)
         3. \(g(w) := e;\)
         4. switch \(c(w);\)
         5. \(\text{push}(w, \mathcal{P});\)
         6. end If;
   3. else
      1. \(\text{pop}(\mathcal{P});\)
   4. end If;
7. end While;

The remainder of the proof will be devoted to proving the validity of the algorithm.

Since the algorithm consists in a While loop containing an if/then/else statement, we call an iteration of the algorithm one completion of that statement.

First we remark that \(T\) is indeed a tree since one endpoint of each new edge of \(T\) is a new vertex. Then for a given \(x \in V'\) there is a unique path in \(T\) from \(x_0\) to \(x\). Moreover when \(x\) is at the top of \(\mathcal{P}\), then the stack contains exactly the vertices of that path (because \(\mathcal{P}\) is a LIFO stack). In addition, at each iteration the existing monochromatic edges are covered by the vertices of \(\mathcal{P}\).
We also remark that if the algorithm does not return Error, then at each iteration either a new vertex is pushed into $\mathcal{P}$, or a vertex is popped from $\mathcal{P}$. Since a vertex appears at most once in $T$ and thus can be pushed at most once in $\mathcal{P}$, we have at most $2|V|$ iterations, and the algorithm ends.

We denote by $\mathcal{P}^{(i)}$, $T^{(i)} = (V^{(i)}, E^{(i)})$, $c^{(i)}$ the values of $\mathcal{P}$, $T = (V', E)$, $c$ (respectively) at the beginning of the $i$th iteration. We also denote by $c^{(0)}$ the original bipartition (which is different from $c^{(1)}$ because of the switch of $c(x_0)$).

Now we prove the following lemma:

**Lemma.** Suppose that $H$ is a Sterboul hypergraph. Consider the beginning of the $i$th iteration. Let $\mathcal{P}^{(i)} = (x_k \ldots x_0)$, and $e_j = g(x_j)$ for $j = 0, \ldots, k$. Then we have:

(a) For each $j = 0, \ldots, k - 1$ we have $e_j \cap e_{j+1} = \{x_j\}$.

(b) Two non-consecutive edges are disjoint.

(c) For each $j = 0, \ldots, k$, $x_j$ is the only vertex of its color in $e_j$.

**Proof of Lemma.** The proof works by induction on $i$.

For $i = 1$ the lemma clearly holds since $\mathcal{P}^{(1)} = (x_0)$.

We now consider $i \geq 1$ and we suppose the lemma holds at iteration $i$. We are going to prove that it also holds at iteration $i + 1$.

First, if during the $i$th iteration Error was returned, there would be no iteration $i + 1$, so assume that Error was not returned. If during the $i$th iteration the algorithm popped $x_k$ from $\mathcal{P}$ (that is, $\mathcal{P}^{(i+1)} = (x_k \ldots x_0)$), the lemma clearly holds at iteration $i + 1$. Thus we assume that the algorithm found $e_{k+1} \in \mathcal{E}$ with $x_k \in e_{k+1}$ that is monochromatic for $c^{(i)}$, and $x_{k+1} \in e_{k+1} \setminus V^{(i)}$ so that $\mathcal{P}^{(i+1)} = (x_{k+1} x_k \ldots x_0)$.

Since (c) holds at iteration $i$, we know that if $w \in e_k \setminus \{x_k\}$ then $c^{(i)}(w) \neq c^{(i)}(x_k)$. As $x_k \in e_{k+1}$ and $e_{k+1}$ is monochromatic for $c^{(i)}$, then $e_k \cap e_{k+1} = \{x_k\}$ and (a) holds at iteration $i + 1$.

Suppose by way of contradiction that (b) fails; since (b) holds at the previous iteration, this means that we can select a maximal $j \leq k - 1$ such that $e_j \cap e_{k+1} \neq \emptyset$ and choose $y \in e_j \cap e_{k+1}$. Note that $y \neq x_j$, since then we would have found $y \in e_{j+1} \cap e_{k+1}$ contradicting the maximality of $j$. If $k - j$ is odd, then $(y, e_j, x_j, \ldots, e_{k+1}, y)$ is an anti-Sterboul cycle because (b) holds at iteration $i$ and (a) holds at iteration $i + 1$, so $k - j$ is even.

The colors $c^{(i)}(y)$ and $c^{(i)}(x_j)$ must differ since $y \in e_j \setminus \{x_j\}$ and (c) holds at iteration $i$. The colors $c^{(i)}(x_j)$, $c^{(i)}(x_{j+1})$, ..., $c^{(i)}(x_{k+1})$ must alternate since (c) holds at iteration $i$ and (a) holds at iteration $i + 1$. Assuming that $k - j$ is even, this implies that $c^{(i)}(y) \neq c^{(i)}(x_{k+1})$ but the colors $c^{(i)}(y)$ and $c^{(i)}(x_{k+1})$ must coincide since $y$ and $x_{k+1}$ are both contained in the $c^{(i)}$-monochromatic edge $e_{k+1}$. Hence $k - j$ cannot be even. Since it cannot be odd either we have a contradiction and thus (b) holds at iteration $i + 1$.

Thus $x_{k+1} \notin e_j$ for all $j = 0, \ldots, k$. Since the only color switch done during the $i$th iteration concerns $x_{k+1}$, (c) holds at iteration $i + 1$.

This completes the inductive step and proves the lemma. \(\square\)

Observe that we claim nothing about the intersection of $g(u)$ and $g(v)$ for $u, v \in V'$ that are never in the stack at the same time. Nevertheless, the validity of the algorithm now follows easily from the above lemma.

Suppose that $H$ is a Sterboul hypergraph. Consider an iteration $i$, and let $\mathcal{P}^{(i)} = (x_k \ldots x_0)$. If $k = 1$ then from (a) of the lemma we have $x_1 \notin e_0$. If $k \geq 2$ then from (b) of the lemma we also have $x_k \notin e_0$. This proves that we always have $e_0 \cap V' = \{x_0\}$.

If Error is returned, it means that at a given iteration $i$, the algorithm found an edge $e$ monochromatic for $c$ such that $e \in V^{(i)}$. Since $e \subseteq V^{(i)}$, this means that each vertex of $e$ had its color switched exactly once from $c^{(0)}$ at the time iteration $i$ occurs. Therefore, $e$ was monochromatic under $c^{(0)}$ with the opposite coloration it receives under $c^{(i)}$. However, $e_0$ is the only edge monochromatic under $c^{(0)}$, so $e = e_0$, a contradiction to $e \subseteq V^{(i)}$ because we have just seen that $e_0 \cap V' = \{x_0\}$. Thus if $H$ is a Sterboul hypergraph, Error cannot be returned.

Finally, since at each iteration the vertices of $\mathcal{P}$ form a cover of the existing monochromatic edges, then if the algorithm terminates without Error, $\mathcal{P} = \emptyset$ and no monochromatic edge can remain.

So the algorithm is correct, the inductive step is completed, and the theorem is proved. \(\square\)
We can slightly modify the algorithm so that it gives an anti-Sterboul cycle instead of just returning Error when the hypergraph is not Sterboul (in order to have a certificate that the hypergraph is not Sterboul).

To do so we just have to check at each iteration that the properties of the lemma still hold. If not it means that the monochromatic edge considered intersects the path induced by the stack, and an anti-Sterboul cycle can be easily found.

While studying the properties of a class of hypergraphs, it is also interesting to study the complexity of recognizing hypergraphs from that class. For example, a hypergraph in which each odd cycle has an edge containing at least three vertices of the cycle is called a balanced hypergraph. These hypergraphs can be bicolored in polynomial time [2], and the polynomial-time recognizability of a balanced hypergraph was resolved affirmatively by Conforti et al. [3].

For Sterboul hypergraphs, our algorithm finds a bicoloration in polynomial time. However it cannot be used to recognize bicolorable hypergraphs (since a non-Sterboul hypergraph may be bicolorable) and neither to recognize Sterboul hypergraphs (since it may happen that it gives a bicoloration for a hypergraph that is not Sterboul).

The problem of recognizing bicolorable hypergraphs is well known to be NP-complete [7]. But we leave the following question open:

**Question.** What is the complexity of recognizing a Sterboul hypergraph?

**Acknowledgement**

I am very grateful to all the people who helped to improve the readability of this paper, and particularly to an anonymous referee.

**References**