The hardness of Intervalizing Four Colored Caterpillars *

C. Àlvarez[†] J. Díaz[†] M. Serna[†]

Abstract

The problem of Intervalizing Colored Graphs (ICG) has received a lot of attention due to their use as a model for DNA physical mapping with ambiguous data. If k is the number of colors, the problem is known to be NP-Complete for general graphs for $k \geq 4$ and has polynomial time algorithms for $k = 2$ and $k = 3$. In this paper we show that the ICG problem is NP-complete when the graph is a caterpillar tree, colored with $k \geq 4$ colors, strengthen the cases for which the problem remains difficult.

1 Introduction

Interval graphs have become very fashionable because they model the overlaps of DNA clones (see Section 5.2 in [SM97]). Recall that in an *interval graph* one can assign to each vertex in the graph an interval of the real line, in such a way that two vertices are adjacent if and only if their intervals have nonempty intersection [Gol80].

Given a graph with colors assigned to its vertices the problem of *Intervalizing Colored Graphs* (ICG) consists in deciding if there is a properly colored supergraph which is an interval graph. The ICG problem is a special case of the *Interval Sandwich Problem* [GS93]. It has received a lot of attention recently because it models (in a simplified way) the problem of reconstructing the original chromosome ordering from DNA physical maps [FHW93].

*This research was partially supported by the ESPRIT Long Term Research Project no. 20244 – ALCOM-IT, CICYT project TIC97-1475-CE, CIRIT project 1997SGR-00366 and DGICYT project DGES PB95-0787 KOALA.

[†]Departament de Llenguatges i Sistemes Informatics, Universitat Politècnica Catalunya, Edifici C6 Campus Nord, Jordi Girona Salgado 1-3, 08034-Barcelona {alvarez,diaz,mjserna}@lsi.upc.es

In this last work they prove that the general ICG problem is NP-Complete (see also [GKS94]). If the problem is given parameterized on the number of colors, hardness results were obtained in [BFH95]. Later it was shown that the problem is NP-Complete for the case of 4 or more colors, while if we only have 2 and 3 colors the problem is solvable in linear and quadratic time respectively [BdF95]. A different approach to study the problem is to fix the number of colors and look at the complexity of the problem considering the degree of vertices. For fixed number of colors k and degree bounded by a constant d there is a $O(n^{k-1})$ algorithm for the ICG [KS96].

Another line to get more information about the hardness of the problem is to look into restricted cases of graphs. One of the most simple graphs is a *caterpillar* tree. A caterpillar consists of a simple chain (called the *backbone*) with an arbitrary number of chains (*hairs*) attached by coalescing an endpoint of the hair with a vertex in the backbone. A caterpillar has hairs of *length* at most h if all the simple chains attached to the body have at most h edges. In [ADS98] we show that the ICG problem is NP-complete in the case the graphs are colored caterpillars with $h \leq 2$ and unbounded number of colors.

In this paper we focus on the case of colored caterpillars with a bounded number of colors. Notice that the reduction given in [BdF95], for 4 colored graphs produces a graph that is not a tree. On the other hand the reduction in [ADS98] produces a caterpillar colored with $n+4$ colors, where n is part of the instance. We strengthen the NP-completeness of the ICG problem to this case. Actually the constructed graph is a four colored caterpillar with unbounded hair length.

To obtain the results, we give a reduction from the multiprocessor scheduling problem to the *Colored Vertex Separation* problem, a layout formulation of the ICG problem [Din96].

2 Definitions and basic results

A *k-coloring* of a graph $G = (V, E)$ is a mapping $\kappa : V \longrightarrow \{1, \ldots, k\}$. For any vertex subset $V' \subseteq V$, let $\kappa(V') = \{\kappa(v) \mid v \in V'\}$. A *proper coloring* of G is a coloring such that no two adjacent vertices have the same color. A *k-colored* graph is a graph together with a k-coloring.

Given a graph $G = (V, E)$ with $|V| = n$ a *layout* φ of a G is a one to one mapping $\varphi : V \longrightarrow \{1, \ldots, n\}$. For any layout φ of G, we can define the sets $V_i = \{v \mid \varphi(v) \leqslant i \text{ and } \exists u \; \varphi(u) > i \; (u, v) \in E\}$ one for each $1 \leqslant i \leqslant n$. Recall that the *vertex separation* of a graph G with respect to a given layout

φ is defined as $\mathrm{vs}(G,\varphi) = \max_{1 \leqslant i \leqslant n}\{|V_i|\}$. The *vertex separation* of a graph G, denoted as $\mathrm{vs}(G)$ is the minimum $\mathrm{vs}(G,\varphi)$ over all layouts of G.

For a given $k \in \mathbb{N}$, a *colored layout* of a k-colored graph $G = (V, E)$ is a layout φ such that for all $u \in V$ with $\varphi(u) > 1$, $\kappa(u) \notin \kappa(V_{\varphi(u)-1})$.

From the definition of colored layout, for any three vertices $u, v, w \in V$ with $(u, w) \in E$, if there is a colored layout φ such that $\varphi(u) < \varphi(v) \leq \varphi(w)$ then $\kappa(u) \neq \kappa(v)$. In this case we say that the color of vu is *blocked* until w appears (in fact until all the neighbors of u appear).

Let us consider the following layout problem: Given a k-colored graph $G = (V, E, \kappa)$, the **colored vertex separation problem** (**CVS**) is to decide if there is a colored layout φ of G.

It is known that for a given parameter $k \in \mathbb{N}$, the CVS problem is identical to the ICG problem [Din96]. Therefore all the P and NP-Completeness results described above for the ICG problem also apply to the CVS problem. Notice that if a k-colored graph has a colored layout then its vertex separation is at most $k - 1$, and viceversa any graph with vertex separation at most $k - 1$ can be colored with k colors in such a way that the graph has a colored layout for such coloring.

A caterpillar with hairs of length at most h is formed by a chain (the backbone), each node in the chain can be connected only to paths with at most h edges.

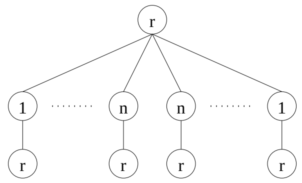

Figure 1: A barrier $B(\Delta, r)$, with $\Delta = \{1, \ldots, n\}$.

A *barrier* is a particular class of subgraph of a colored caterpillar. The subgraph is formed by a star, with length two hairs. The center of the star will be in the backbone of the caterpillar. All nodes at distance 2 (that have degree one) are colored as the barrier's center. There are exactly two paths having the same color in the node in the middle point. Let $B(\Delta, r)$ denote a barrier with color set Δ and *center* colored r with $r \notin \Delta$ (see figure 1, colors are written inside the circles representing the nodes). The main properties of colored layouts for barriers given in [ADS98] are the following.

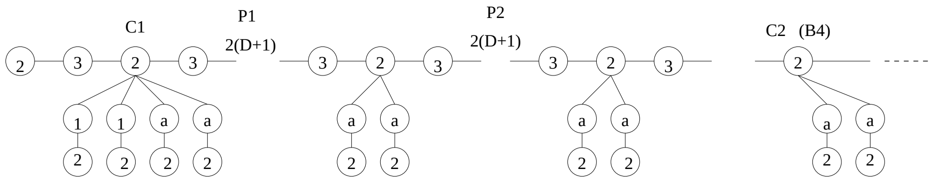

Figure 2: The graph R_m associated to the processors.

Lemma 1. *Given a barrier $B = B(\Delta, r)$, in any colored layout of B the children's center must be partitioned into two sets, one in each side of the center in such a way that in each set there is no repeated color.*

Lemma 2. *Given a barrier $B = B(\Delta, r)$ and an edge (u, v) such that $\kappa(\{u, v\}) \subseteq \Delta \cup \{r\}$ and u and v do not form part of the barrier. In any colored layout of the barrier and the edge, the two endpoints of the edge must lay both on the same side (before or after) of the barrier's center.*

3 The reduction

We will show that the CVS problem is NP-complete for caterpillars, more precisely our reduction will produce a caterpillar with hairs of unbounded length, colored with four colors. We give a reduction from the *Multiprocessor Scheduling problem*. That is, given a set of n tasks, having duration t_i for $1 \leqslant i \leqslant n$, a deadline D and a set of m processors determine whether the tasks can be assigned to processors so that all processors finish their work before the deadline D. Recall that the problem is NP-complete in the strong sense, so we can assume that the duration of each task is polynomially bounded.

Our reduction will produce a colored caterpillar such that it has a colored layout if and only if the original Multiprocessor Scheduling instance has solution. The reduction construct three gadgets, one for the processors, one for the tasks and an additional one called the *turning point*. Informally the processors part consists of m slots, formed by a line with alternated colors, separated by barriers. Tasks correspond also to lines with alternated colors. The turning point forces the overlap of tasks and processors in the (possible) colored layout.

The graph R_m, associated to the m processors, (the same used in [ADS98]) is the caterpillar given in figure 2. In this figure the backbone corresponds to the horizontal line. The graph is formed by a series of barriers whose centers

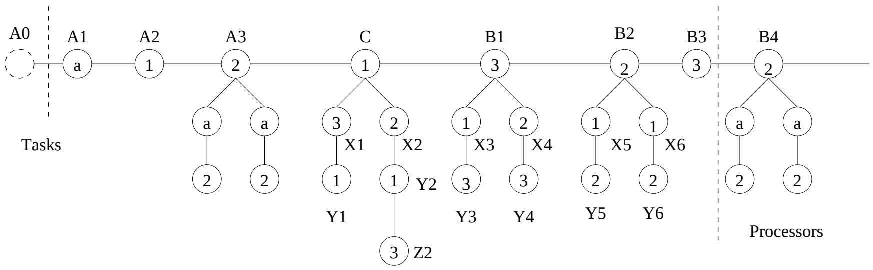

Figure 3: The turning point T.

are connected by a path with $2(D+1)$ edges, the nodes in the backbone alternate colors between 2 and 3. It has two big barriers, involving all four colors, one at the beginning and the other at the end. All internal barriers are small ones, involving only three colors. One of the big barriers forms also part of the turning point gadget.

Lemma 3 ([ADS98]). *In any colored layout of the graph R_m the backbone must be layed out as it is, in between the nodes c_1 and c_2.*

The second gadget is the colored caterpillar T given in figure 3, the backbone follows the horizontal line, and the dashed lines represents the connection to the other two gadgets. Some of the nodes are labeled, this labels will be used to refer to them. We will refer to such caterpillar as *the turning point* with center c. The gadget here is different to the one used in [ADS98], it just uses 4 colors instead of the $n+4$ used in the other reduction, but its role is the same, force that the colored layout (if any) must overlap the other two gadgets.

Theorem 1. *In any colored layout of the turning point T, the two end points, a_1 and b_4 lay on the same side of c.*

Proof. First notice that the turning point contains three barriers, centered at c, b_1 and b_2 respectively.

The first barrier, induced by the nodes $\{a_2, a_3, x_1, x_2, y_1, y_2, b_1, x_3, c\}$ has center c and colors $\{2,3\}$. By lemma 1 in any colored layout of T, the nodes x_1 and b_1 cannot lay on the same side of c, and the same happens with the nodes x_2 and a_3.

The second one is formed by the nodes $\{x_1, x_3, x_4, c, b_1, b_2, b_3, y_3, y_4\}$ with center b_1 and colors $\{1,2\}$. By lemma 2 the edges (x_2, y_2) and (Y_2, z_2) cannot cross the center of a barrier with colors $\{1,2\}$, therefore the nodes x_2, y_2 and z_2 must be placed in the same side, with respect to b_1.

The third contains the nodes $\{x_4, b_1, y_5, x_5, y_6, x_6, b_4, b_3, b_2\}$ with center b_2 and colors $\{1,3\}$.

Let us show that the nodes b_1 and x_2 can not be on the same side of the node c. The proof considers two cases, the two possible positions of b_1 with respect to c.

In the case $\varphi(c) < \varphi(b_1)$ the nodes x_2, y_2 can not be placed in between c and b_1 because c blocks color 1 before b_1.

If $\varphi(c) < \varphi(b_1) < \varphi(x_2)$ then the problem arises when we try to lie b_2 or x_4 with color 2. Since c is in the left of b_1, the center of the second barrier, the nodes x_3 and y_3 must lay on the right side of b_1. As you can see in figure 4 the only possible layout must satisfy

$$\varphi(c) < \varphi(b_1) < \varphi(x_2) < \varphi(x_3) < \varphi(y_3) < \varphi(y_2) < \varphi(z_2).$$

But now it remains to lay one of the edges (b_2, b_3) or (x_4, y_4) to the right of b_1, and that is not possible.

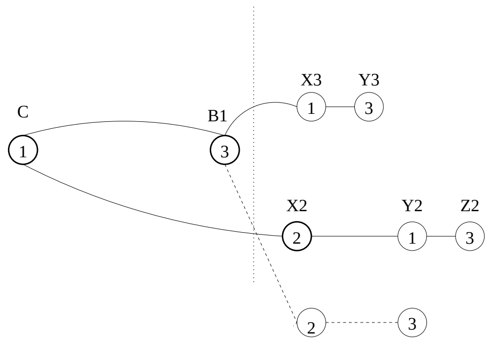

Figure 4: The case $\varphi(c) < \varphi(b_1) < \varphi(x_2)$.

In the case $\varphi(b_1) < \varphi(c)$, if the node x_2 is in between b_1 and c then the edge $(x_2, 2_2)$ can not cross b_1, the center of the second barrier, neither c, the center of the first barrier. The same happens with the edge (y_2, z_2).

If $\varphi(x_2) < \varphi(b_1) < \varphi(c)$ as b_1 is the center of the second barrier it remains to lay one of the edges (b_2, b_3) or (x_4, y_4) to the left of b_1. But then, according to figure 5), there is no layout.

But that means that nodes x_2 and b_1 cannot lay in the same side with respect to c.

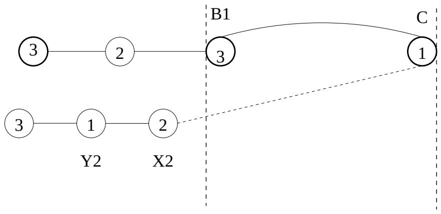

Figure 5: The case $\varphi(x_2) < \varphi(b_1) < \varphi(c)$.

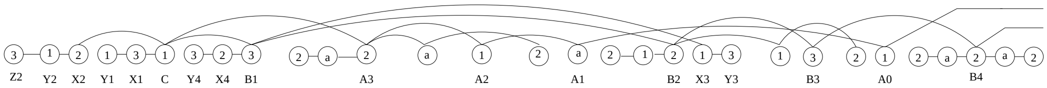

Figure 6: A layout for the turning point.

At this point we have that the pairs, x_1 and b_1, x_2 and a_3, and x_2 and b_1 can not be placed in the same side with respect to c. The only possibility is to lay a_3 and b_1 on one side of c and x_1 and x_2 on the opposite side of c.

When a_3 and b_1 are on the right of c, as b_1, b_2, b_3, b_4 form a chain with alternating colors we have $\varphi(c) < \varphi(b_1) < \varphi(b_2) < \varphi(b_3) < \varphi(b_4)$. As $\varphi(c) < \varphi(a_3)$, and $(a_3, a_2) \in E$ and $\kappa(a_2) = \kappa(c)$ we must have $\varphi(c) < \varphi(a_3) < \varphi(a_2)$. Finally a_1 cannot be on the left of c because the edge (a_1, a_2) contains the colors of the barrier with center a_3. Therefore b_4 and a_1 must be placed on the rigth of c. A similar argument applies when a_3 and b_1 are on the left of c. $\square$

We have that the only possible colored layout of the turning point T, when a_1 and b_4 lay to the right of c, is the one given in figure 6 modulo some permutation of the nodes in the barriers. Similarly it can be obtained a layout for the other case.

Figure 7: The graph T_i associated to task i.

Finally we use the colored caterpillar T_i of figure 7 to represent task i. We will join all the graphs T_i with the node colored 1 that will be connected to the node a_0, an end point of the turning point (see figure 8).

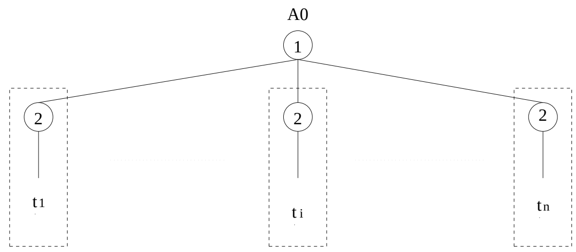

Figure 8: The graph associated to the tasks.

For sake of simplicity let us denote by L_d a two colored line of length $2d$ such that the end points are colored with color 2 and the chain alternates colors between 3 and 2. This is just a part of the backbone of the caterpillar for the processors. In [ADS98] we show under which conditions the lines corresponding to tasks can be layed together with the piece of backbone corresponding to a processor.

Lemma 4. *Consider the graphs L_d, T_i, T_j . In any colored layout in which the first and the last node are the end points of L_d, T_i and T_j can be placed together with L_d if $t_i + t_j < d$.*

Given an instance $I = (t_1, \ldots, t_n, m, D)$ of the Multiprocessor Scheduling problem let $G'(I)$ be the caterpillar obtained from the different gadgets (see figure 9). Then we have

Theorem 2. *The instance $I = (t_1, \ldots, t_n, m, D)$ of multiprocessor Scheduling has a solution if and only if the colored caterpillar $G'(I)$ has a colored layout.*

Proof. From the previous lemmas the only possible colored layout for the caterpillar must maintain as it is the ground line corresponding to the processors, between the center of the two big barriers, the turning point must be *turned* at its center and the two endpoints on the same side (by theorem 1). The piece corresponding to the tasks must be placed between the center of the two barriers of the processors part. The part corresponding to just one task, must go between the center of two barriers. And from lemma 4 the total duration of the tasks placed in the line corresponding to a processor must be at most D. Finally notice that the additional node with different color allow us to place different tasks in any relative position provided we

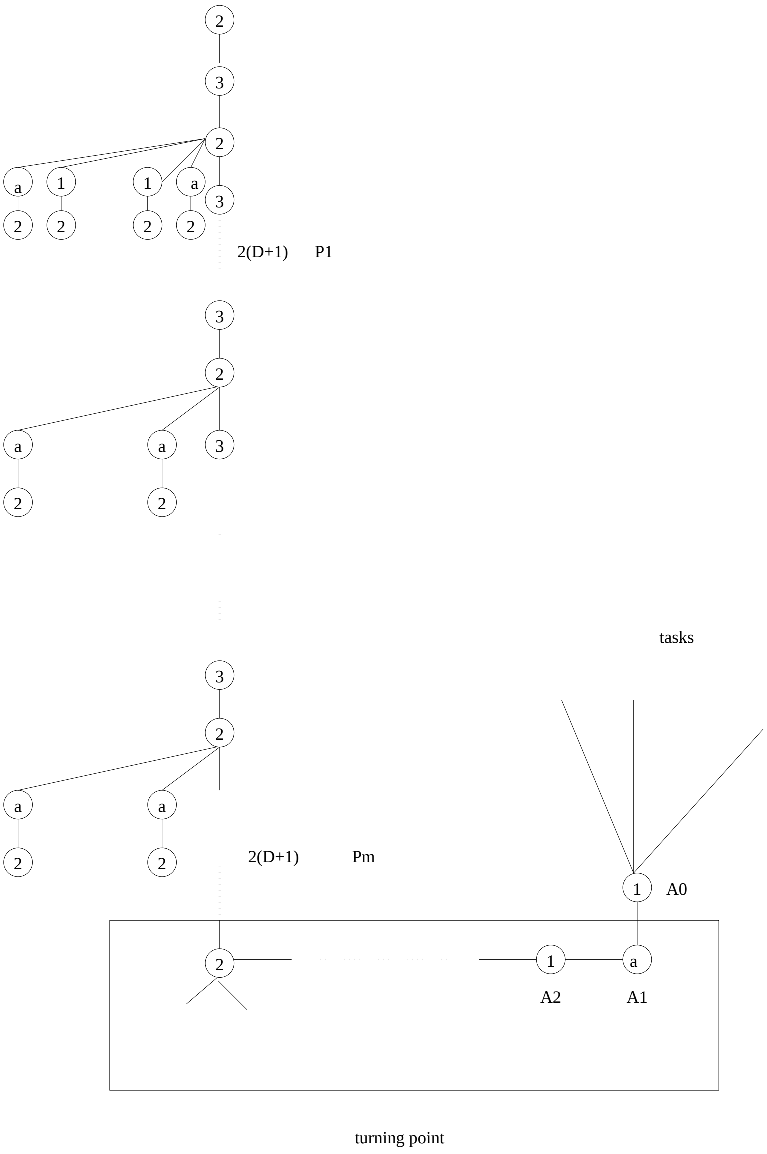

Figure 9: A sketch of the caterpillar $G'(I)$.

place this node in the first position that can be placed. Once it is layed the remaining hairs can be placed in any position provided they do not overlap, but the last situation is not possible. So each scheduling meeting the deadline requirement gives a colored layout and every colored layout gives a correct scheduling. □

References

[ADS98] C. Àlvarez, J. Díaz, and M. Serna. Intervalizing colored graphs is NP-complete for caterpillars with hair length 2. Technical Report LSI 98-9-R, Universitat Politècnica de Catalunya, 1998.

[BdF95] H. Bodlaender and B. de Fluiter. Intervalizing k-colored graphs. In Z. Fulop and F. Gecseg, editors, *International Colloquium on Automata Languages and Programming*, volume 944, pages 87–98. Springer-Verlag, Lecture Notes in Computer Science, 1995.

[BFH95] H. Bodlaender, M.R. Fellows, and M.T. Hallet. Beyond NP-completeness for problems of bounded width: hardness for the W-hierarchy. In *ACM Symposium on Theory of Computing*, pages 449–458, 1995.

[Din96] M.J. Dinneen. VLSI Layouts and DNA physical mappings. Technical report, Los Alamos National Laboratory, 1996.

[FHW93] M.R. Fellows, M.T. Halett, and T. Wareham. DNA physical mapping: Three ways difficult. In T. Lengauer, editor, *1st European Symposium on Algorithms*, volume 726 of *Lecture Notes in Computer Science*, pages 157–168. Springer-Verlag, Berlin, 1993.

[GKS94] M.C. Golumbic, H. Kaplan, and R. Shamir. On the complexity of DNA physical mapping. *Advances in Applied Mathematics*, 15:203–215, 1994.

[Gol80] M.C. Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, New York, 1980.

[GS93] M.C. Golumbic and R. Shamir. Complexity and algorithms for reasoning about time: A graph theoretical approach. *Journal of the ACM*, 40:1108–1113, 1993.

[KS96] H. Kaplan and R. Shamir. Physical maps and interval sandwich problems: Bounded degrees helps. Technical report, Shackler Faculty of Exact Sciences, Tel Aviv, 1996.

[SM97] J.C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology.* PWS Publishing Company, Boston, 1997.