Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

Threshold effect in algorithmic complexity: NP-hard and linear variants of hypergraph partitioning

Laurent Lyaudet

28th September 2005

Research Report N° 2005-46

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

CENTRE NATIONAL
DE LA RECHERCHE
SCIENTIFIQUE

INRIA

Threshold effect in algorithmic complexity: NP-hard and linear variants of hypergraph partitioning

Laurent Lyaudet

28th September 2005

Abstract

This report present an infinite family of combinatorial problems that shows extremely abrupt changes of complexity between neighbours problems. I will show that the problem P_l^k that is a purely constraint-driven variant of hypergraph partitioning is for fixed parameters l and k NP-complete when $k = 1$ and is linear when $k \neq 1$ on the class of hypergraphs with disjoint hyperedges.

Keywords: Hypergraph partitioning, NP-Completeness, Threshold effect

Résumé

Ce rapport présente une famille infinie de problèmes combinatoires qui montre des changements de complexité extrêmement brusques entre problèmes voisins. Je vais montrer que le problème P_l^k, qui est une variante du partitionnement d'hypergraphes uniquement basée sur la satisfaction de contraintes, est, pour des paramètres l et k fixés, NP-complet quand $k = 1$ et linéaire quand $k \neq 1$ sur la classe des hypergraphes avec des hyperarêtes disjointes.

Mots-clés: Partitionnement d'hypergraphes, NP-Complétude, Effet de seuil

0.1 Introduction

In this article, I present an infinite family of combinatorial problems that shows extremely abrupt changes of complexity between neighbours problems. These problems are variants of the hypergraph partitioning problem. The hypergraph partitioning problem consist of finding a partition of the vertex set of the hypergraph such that a number of constraints are satisfied and/or an objective function is minimized. In general, this problem is NP-complete. A notable exception is the MinCut Bipartition problem that is polynomial (see, e.g. [4]). Variants of hypergraph partitioning are used in many areas, like VLSI design [3], efficient storage of large databases on disks [12], operations on sparse matrices [1, 13], information retrieval [8], data mining [11, 6], and study of semi-Markov processes [2]. Practical applications need the partition classes to be of almost equal size. Many heuristic algorithms have been developped over the last thirty years (see, e.g. [7]) but, to my knowledge, no approximation algorithm (except [10]) or PTAS is known for variants of this problem even on hypergraphs subclasses.

I have studied the complexity of a purely constraint-driven variant of hypergraph partitioning on subclasses of the interval hypergraphs. This variant is appearing in the study of branchwidth of circle graphs [9]. Let $X(t)$ be a totally ordered set of size t (an interval of size t). Let I be a set of n sub-intervals of $X(t)$ such that $\forall x \in X, |\{i \in I / x \in i\}| \leq e$, where e is the thickness of the problem (in hypergraph terms, e is the maximum degree of the hypergraph). Does a coloring (partition) of X in l subsets of size at most $\lceil \frac{t}{l} \rceil$ exist, such that every element of I is colored with at most k colors, where $0 \leq k \leq l$? We will note this decision problem P_l^k.

This problem can be seen as the following scheduling problem. Consider a set of n programs, each program being associated to a natural number that represent the number of unitary tasks that communicate together to achieve the program. Let t be the sum of those natural numbers. Consider also a set of l fixed processors that are, in a first time, supposed to be of same power. The goal is to dispatch the tasks that compose the programs on the processors such that all the processors finish at approximately the same time and that each program is dispatched on a minimal number of processors to minimize the communications between processors. Hence, we will associate to each processor a color of size floor or ceiling of t/l. We obtain the following decision problem: "Does a coloring under the constraint of size of the l colors such that each program has his tasks colored by at most k distinct colors exist?"

In this paper, I will show that the problem P_l^k is NP-complete when $k = 1$ and is linear when $k \neq 1$ on the class of hypergraphs with disjoint hyperedges. In Section 0.2, I will introduce the notations and make some basic observations. In Section 0.3, I will show that the problem P_l^k is NP-complete when $k = 1$ and make simple remarks about this result. Section 0.4 contains the first results of linearity and is a soft approach to the problem. In Section 0.5, I generalize the problem to colors of arbitrary size and adapt the notions introduced in Section 0.4. In Section 0.6, I prove structural results about the possible solutions that enables me to give a linear algorithm for the generalized problem when $k \neq 1$.

0.2 Preliminaries

In the rest of this article, $t(i)$ will be the size of the sub-interval i, $t(c)$ the size of the color c and $c(i) = i(c) = |c \cap i|$ the number of units of the color c in i. We will say that the interval i

"sees" the color c or reciprocally that the color c sees the interval i if $c(i) = i(c) > 0$. We will note C the set of colors, $C(i)$ the set of colors that are seen by i and $I(c)$ the set of intervals that are seen by c.

Let us remark at first that if $k = l$ then the problem is true for every instance. If $k = 0$, the problem is false for every non-empty instance.

If $t \leq l$ then all the colors are of size at most 1 and then it is sufficient to test if a sub-interval of size more than k exists. If an interval is of size less than or equal to k then it accepts every coloring and we can remove it from the instance.

With thickness 1, I is a set of n disjoints sub-intervals. We can notice that the problem has a dual nature if we consider things from the viewpoint of hypergraphs. Indeed, we can see I as a family of hyperedges on X satisfying the following properties:

- $\forall i, j \in I, i \cap j = \emptyset$ (Intersection property) ;
- there exists a total order on X such that $\forall i \in I, \forall x, y \in i$, either $x \leq y$ and $\forall z$ such that $x \leq z \leq y, z \in I$, or $y \leq x$ and $\forall z$ such that $y \leq z \leq x, z \in I$ ("Intervality" property).

Whereas C is a second family of hyperedges on X that satisfies the following properties:

- $\forall c, d \in C, c \cap d = \emptyset$ (Intersection property) ;
- $\forall x \in X, \exists c \in C$ such that $x \in c$;
- $\forall c \in C, |c| \leq \lceil \frac{t}{l} \rceil$.

The "intervality" property brings nothing to the problem with thickness 1; we can always find an order such that a family that satisfies the intersection property satisfies also the "intervality" property. The decision problem consist to ask if, being given two families of hyperedges I and C that are disjoints and whose we know the size, there exist a positioning of thoses hyperedges such that every hyperedge of I is adjacent to at most k hyperedges of C.

In the rest of the paper we assume that the thickness of the problem is 1. Nevertheless, as we shall indicate, some results hold for arbitrary thickness. We will look at the complexity of P_l^k for fixed parameters l and k. We will now show by induction that the problem P_l^k is NP-complete when $k = 1$.

0.3 The NP-complete case

Theorem 1 *The problem P_l^1 is NP-complete, $\forall l \geq 2$*

Proof :

Initialisation: the problem P_2^1 consist to find a partition of X in two colors of size variating of at most one, such that each element of I is in only one color. We can trivialy code an instance of the problem of 2-partition of size $n \log(t)$ of a set of n natural numbers $e_1, \ldots, e_n$ whose sum is t even in an instance of size $O(n \log(t))$ of P_2^1 by associating to each natural number e_i a sub-interval of size e_i of an interval X of size t. If t is odd, the instance is trivialy false, then P_2^1 is NP-complete.

Induction : Suppose that P_l^1 is NP-complete, let us show that P_{l+1}^1 is too. Indeed, consider an instance of P_l^1 with n sub-intervals of $X(t)$, it is of size $O(n \log(t))$. This instance satisfies the problem P_l^1 if and only if the following instance of P_{l+1}^1 is satisfiable. We take the n sub-intervals which we add a sub-interval of size $\lceil \frac{t}{l} \rceil$. We obtain an instance of size $O((n+1)\log(t+\lceil \frac{t}{l} \rceil)) = O((n+1)\log((1+\frac{1}{l})t)) = O((n+1)(\log(t)+\log(1+\frac{1}{l}))) = O(n\log(t)+\log(t)) = O(n\log(t))$. Hence, P_{l+1}^1 is NP-complete. ■

Let us remark here that the problem P_l^k with thickness $e+1$ obviously contains the problem P_l^k with thickness e, so P_l^1 is NP-complete whatever the thickness is. But the interesting point is especially that the problem P_l^1 in superior thickness is exactly identical to the problem with thickness 1. Indeed, if two sub-intervals have a non empty intersection, since each sub-interval must see a single color, they must see the same color and we can remplace them by the union of both. So we end up with a set of disjoints sub-intervals. This shows us that there is a collapse of the complexity on the thickness near the border of the problem's family. In fact, for all considered hypergraphs we come down to a hypergraph with disjoint hyperedges. It is the second reason why it is natural to consider interval hypergraphs for studying this partition problem.

0.4 The linear case

Now that we know that a border of the family is NP-complete, we will be interested in the members of the other border. Let us start by showing that a half of the problems has a linear complexity. In the rest of the paper, we define $r = t \mod l$. This quantity counts the number of "big" colors (those of size $\lceil \frac{t}{l} \rceil$) whereas $l - r$ count the number of "small" colors (those of size $\lfloor \frac{t}{l} \rfloor$). We define also

$$L = \min(k, r)\left\lceil \frac{t}{l} \right\rceil + \max(0, k-r)\left\lfloor \frac{t}{l} \right\rfloor$$

the limit size. This quantity counts the maximal size that can be covered by k colors. We will lean on a particular type of coloring for reasoning:

Definition 1 *Being given a total order on X such that $H = (X, I)$ is an interval hypergraph for this order, we will say that a coloring of X is* continuous *relatively to this order if all the colors are intervals except maybe one that is the union of an interval containing the smallest element of the order and an interval containing the biggest element of the order. A coloring is said to be* continuous *if there is such an order such that it is continuous relatively to this order.*

We will next study two cases and prove that if there is a solution, then a continuous coloring is among the solutions. The study of the third case with the same approach will show its limitations and guide us to the next section.

First case: $l - 2k < 0$, it is sufficient to look if there exists or not a sub-interval of size strictly superior to the limit size L. Indeed, if there is one, it is not coloriable. If there is no such interval, we color X continuously beginning with color 1, then the color 2 when we have exhausted the color 1, ... We will say that an interval i exhausts a color c if all the elements of X c-colored are contained in i. If a sub-interval does not accept this type of coloring, then it exhausts at least $k-1$ colors and it sees two more colors. Let us give it a name :

Definition 2 *A* troublemaker *is a sub-interval that can exhaust at least $k-1$ colors and see two more colors.*

One can easily prove that we have the following property.

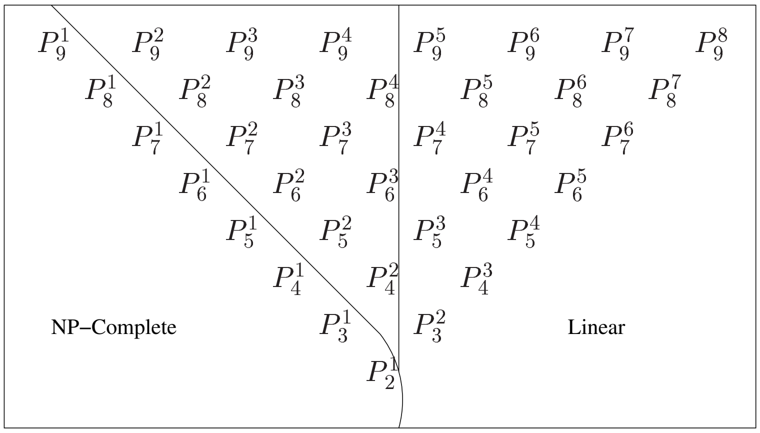
Property 1 *A sub-interval is a troublemaker if and only if it has size at least*

$$\min(k-1, l-r)\left\lfloor \frac{t}{l} \right\rfloor + \max(0, k-1-l+r)\left\lceil \frac{t}{l} \right\rceil + 2.$$

Suppose there exists a troublemaker. If the first interval to be colored is a troublemaker, since his size is less than L, he will accept the coloring. And more, we can exhaust at least $k-1 \geq \lfloor \frac{l}{2} \rfloor = l - \lceil \frac{l}{2} \rceil \geq l-k$ colors so it remain at most $l-(k-1) \leq l-(l-k) = k$ colors. Hence we can color the remaining intervals as we want.

This shows us that for answering the decision problem, we only need to test if the condition on the limit size is satisfied for all the intervals. If we want to give an acceptable coloring, we just need to color continuously starting from the border of a troublemaker if there is one. For this coloring we can see the interval as a torus. Hence, the problem is linear.

For now we have the following distribution:



We have obtained this first result of linearity thanks to the notion of troublemaker. More exactly, using the fact that we can exhaust enough colors in a single troublemaker so we do not have to bother with the others intervals. Let us consider the following family of properties: the interior of j troublemakers is sufficient to exhaust enough colors so that we do not have to bother with the others intervals. More formally the property $Prop_1(j) : l - j(k-1) \leq k$. If $k \neq 1$, we can always find j such that property $Prop_1(j)$ is satisfied. We can also always find j such that property $Prop_1(j)$ is not satisfied ($Prop_1(0)$ is always unsatisfied). We will say that the problem P_l^k has property $Prop_2(j)$ if it satisfies $Prop_1(j)$ and not $Prop_1(j-1)$. The study of the first case is then corresponding to the case where the problem has property $Prop_2(1)$.

In parallel to this notion of troublemaker, we must verify that there are enough "big" colors so we can color all the big intervals. We express this condition with the following inequality: $\sum_{i \in I} \max(0, t(i) - k \lfloor \frac{t}{l} \rfloor) \leq r$. This is the «big» condition.

We will now study the cases $Prop_2(2)$ and $Prop_2(3)$ in order to introduce the last tools we will use for solving the general case.

Case $Prop_2(2)$: We have the two following inequalities: $l - 2(k-1) \leq k$ and $l - (k-1) > k$. We start by checking if the "big" condition is true. If there is 0 or 1 troublemaker, a continuous coloring is sufficient. If there are two, we know that their interior is sufficient to exhaust enough colors so we only need to check if we can color those two troublemakers. But $l - (k-1) > k \Leftrightarrow l - 2k + 1 > 0 \Leftrightarrow l - 2k \geq 0$. This means that we can split the interval in two blocks of at least k colors and put each troublemaker in one of the two blocks. It was too easy, interesting things begin with the following case.

Case $Prop_2(3)$: We have the two following inequalities: $l-3(k-1) \leq k$ and $l-2(k-1) > k$. As for the case $Prop_2(2)$, we start by checking if the "big" condition is true. If there is 0 or 1 troublemaker, a continuous coloring is sufficient.

If there are two, we put them into two blocks of k colors. As $l - 2(k-1) > k \Leftrightarrow l - 2k > k-2 \Leftrightarrow l-2k \geq k-1$, it remain at least $k-1$ colors totally free (unseen by the troublemakers) as well as at most two partially used. Here comes the first difficulty, we can no more consider balanced colors whose size differ of at most one. By eliminating the two troublemakers, we have created some "very small" colors that mess everything up.

If there are three troublemakers, we would like to put them in three blocks as in the previous case. But $l - 2(k-1) > k \Leftrightarrow l - 3k + 2 > 0 \Leftrightarrow l - 3k + 1 \geq 0$. So sometimes, like with P_5^2, we must put two troublemakers in the same block. It is the second difficulty, we must consider the interactions between troublemakers.

How to solve the difficulties that arise during the study of the case $Prop_2(3)$?

- Firstly by generalising the problem. We will consider colors that are partitioning X without giving conditions on their size. We can see this in two different ways. Either to add to the instance the size of the colors or to define P_l^k problems in the non-balanced case, by associating to each of the l colors a fraction representing the proportion of units of X in this color. In fact the second way is more a method to supplement the instance and obtain the size of the colors in function of t.

- Secondly by looking at the structure of the solutions in detail. For this purpose, we will look at a graph representing a partition and see in detail the properties of such a graph when the partition is a solution.

0.5 Generalisation of the problem

What happens when the colors are no more balanced ? First of all, the NP-complete problems remain NP-complete. In particular the problem P_2^1 that was containing 2-PARTITION contains now the SUBSET SUM problem [5].

Let us look now at the non NP-complete problems starting with $Prop_2(1)$. We need to replace the Property 1 by the following property in order to understand the notion of troublemaker in the general case.

Property 2 *A* troublemaker *is a sub-interval of size at least*

$$\min_{C' \in \mathcal{P}_{k-1}(C)} (\sum_{c \in C'} t(c)) + 2.$$

This property is equivalent to the formula that we had in the balanced case.

If there is no troublemaker a continuous coloring is sufficient. If there is one, we must check if its size is less than

$$L = \max_{C' \in \mathcal{P}_k(C)} (\sum_{c \in C'} t(c)).$$

Here again the new formula which defines L is equivalent to the previous. The result that would allow us to conclude would be the certainty that we can exhaust $k-1$ colors in the interior of a troublemaker.

Proposition 1 *Let g be a troublemaker whose size does not exceed the limit size L. Then we can color g with k colors and exhaust $k-1$ of this colors.*

Proof :

We start with $k-1$ colors that are realising the minimum in the definition of troublemaker and we add the biggest color. If this k colors are not sufficient for covering g, we change the smallest used color for the biggest unused color. We repeat those changes until the k used colors are of size greater than those of g. As $t(g) \leq L$ the stop condition always happens. Once we have found the k colors we know that the $k-1$ small colors are smallest than g so we can exhaust them.

■

Fortunately, the problem is still linear in the case $Prop_2(1)$. If we want to solve in the same way the case $Prop_2(2)$, we must define an analoguous condition to the "big" condition. Moreover, it is possible that we cannot put two troublemakers in two blocks of k disjoint colors because it is necessary for the second troublemaker to use the big rests of the other rather than some little colors. Here is an example of such an obligation : P_7^3, $t = 54 = 10 + 10 + 10 + 10 + 10 + 2 + 2$, $t(i_1) = t(i_2) = 25$, $t(i_3)$=4. Faced with those difficulties we must start the next section.

0.6 Structure of the solutions

All our approach comes down to exhausting colors until there are at most k left. If we prove that every satisfiable instance has a solution in which an interval exhaust at least one color then the problem is weakly polynomial.

Why polynomial? Because it is sufficient to test for all intervals all the partial colorings that are exhausting one color in this interval and to check if the resulting instance of $P_{l'}^k$ is satisfiable ($l' < l$). We obtain an algorithm with an approximate complexity of $(C_l^k)^{l-k} * n^{l-k}$ so in $O(n^{l-k})$.

Why weakly? If we take into account the size t and all the sizes of the colors and of the intervals that are of the same order of magnitude, we remark that we encode them in $\log(t)$-space. But if $k \geq 3$ the interval i that we have choosed can exhaust one color and see two others without exhausting them. This means that the excess of this two colors compared to the size of i can be dispatched in $O(t)$ different ways between them. We end up with a number of sub-problems that is exponential in $\log(t)$.

Why is it strange? The preceding remark could let us fear that the closest problems to those that are NP-complete are polynomial while those that are a little more distant are only weakly polynomial and why not NP-complete. We would then have a family of NP-complete problems with a hole inside.

The following theorems will show us that the situation is not so complicated. They will help us simplify the structure of the following graph that modelize the structure of a coloring.

Definition 3 *Being given a coloring, we call* coloring graph *the undirected bipartite graph $G = (I \cup C, E)$ where $(i, c) \in E$ for $i \in I$ and $c \in C$ if and only if the color c sees the interval i. We can add weights to the edges that corresponds to the number of units shared by the concerned interval and color. Let us denote this weights by $p(i, c) = i(c) = c(i)$.*

Here are some obvious propositions on the properties of a coloring graph.

Proposition 2 *Let G be a coloring graph then $\sum_{c \in N(i)} p(i, c) = t(i)$, $\forall i \in I$*

Proposition 3 *Let G be a coloring graph. If I covers totally X then $\sum_{i \in N(c)} p(i, c) = t(c)$, $\forall c \in C$*

The first theorem gives us a caracterisation of the good solutions when they exist.

Theorem 2 *Let (t, I, C) be an instance of P_l^k with thickness 1. If this instance is satisfiable then there is an acceptable coloring such that his graph is a forest.*

Proof :

Let $G = (I \cup C, E)$ be the corresponding graph of any acceptable coloring. Assume that there is a cycle of length 4 containing exactly two colors c_1,c_2 and two intervals i_1 et i_2. Assume without loss of generality that $c_1(i_1) \leq c_2(i_2)$ then i_1 can say to i_2 : "Take my marbles of color c_1 and give me the same number of marbles of color c_2". Since i_1 has big muscles, i_2 obtemperate and we obtain a coloring in which i_1 sees one color less than before and i_2 sees at most as many color than before. More formally, the new values are $c_1(i_1) = c_1(i_1) - c_1(i_1) = 0$, $c_2(i_1) = c_2(i_1) + c_1(i_1)$, $c_1(i_2) = c_1(i_2) + c_1(i_1)$ and $c_2(i_2) = c_2(i_2) - c_1(i_1) \geq 0$. Thus we can obtain a coloring such that there is no cycle of length less than 5. Assume now that there is an elementary cycle of any length in this graph. Let us call it $\mu = (i_1, c_1, \ldots, i_m, c_m)$. i_1 says to i_2 : "Take my $c_1(i_1)$ marbles" and i_2 answers : "I can't take more than my $c_2(i_2)$ marbles and only if i_3 accepts to take them". i_2 turns toward i_3 and ask him to take $\min(c_1(i_1), c_2(i_2))$ marbles and so on. At the end they agree to exchange $\min_{x \in \{1,\ldots,m\}}(c_x(i_x))$ marbles. More formally, the new values are $c_x(i_x) = c_x(i_x) - \min_{x \in \{1,\ldots,m\}}(c_x(i_x)) \geq 0$ and $c_x(i_{x+1}) = c_x(i_{x+1}) + \min_{x \in \{1,\ldots,m\}}(c_x(i_x))$. In this exchange, an i_x such that x realize the minimum has lost all his marbles of color c_x, it sees one color less and it is no more linked to his successor in the cycle. All the others intervals sees at most as many colors than before. By repeting this method, we destroy all the cycles and we obtain an acceptable coloring such that the corresponding graph is a forest.

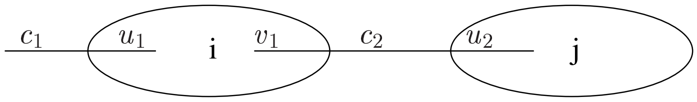
∎

The following theorem enables us to refute the possibility of a complexity that is weakly polynomial and not polynomial.

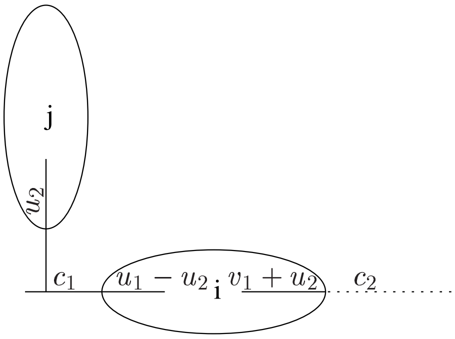
Theorem 3 *Let (t,I,C) be an instance of P_l^k and $i \in I$ the biggest interval. If this instance is satisfiable then there is a solution such that i exhausts all the colors that he sees except for at most one.*

Proof :

Assume that the instance is satisfiable. Let us consider the graph G obtained thanks to the Theorem 2. If i is the only interval in his connected component, it exhausts all the colors he sees so we have won. Else we want to make i a pseudo-leaf i.e. all the adjacent colors to i except for one are leaves. Assume that i is not alone in his connected component and it is not a pseudo-leaf. In order that i becomes a pseudo-leaf, we will put all the others intervals of the component in one unique branch starting from i. Let c_1 and c_2 be two colors seen by i and its neighbors. We will give a rocket to i in order that it moves from c_1 to c_2. Let j be a neighbour of i that sees c_2. Let us denote $p(i, c_1) = u_1$, $p(i, c_2) = v_1$, and $p(j, c_2) = u_2$

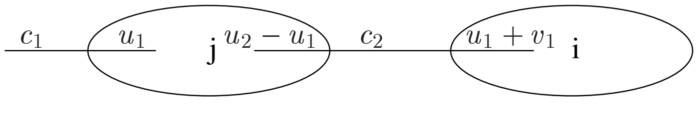


If $u_2 \leq u_1$, then we can push all the units of the color c_2 that are in j inside of i and make go out u_2 units of c_1 in order to reconnect j behind i. Moreover, if c_2 is seen only by i and j, then it is now totally in i. We have increased by at least one the number of intervals that are in the good branch of i.

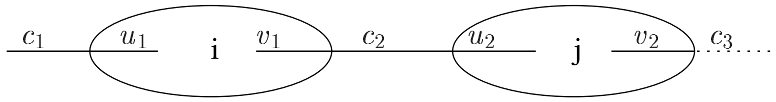


We can remark that if $u_2 = u_1$, we have succeed to break the connected component of i in two parts and decrease by one the numbers of colors seen by i. If that is the case and i is neither a pseudo-leaf nor the unique interval in its component, we restart the process on the new connected component which is strictly smaller.

If $u_2 > u_1$ and j sees less than k colors then we can exchange i and j, i.e. make eat u_1 units of c_1 and one part of the units of c_2 by j and the resting units of c_2 by i. Thus, j goes again in the good branch of i. j sees one more color and i one less.

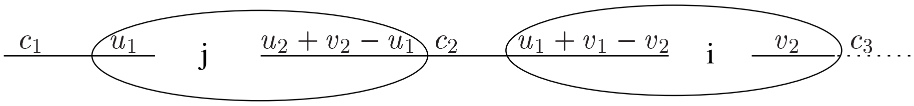


Assume that j sees k colors then there exists c_3 a color seen by j different that c_2 (c_3 may be seen only by j so what follows is correct even if j is a pseudo-leaf). If $u_2 > u_1$ and



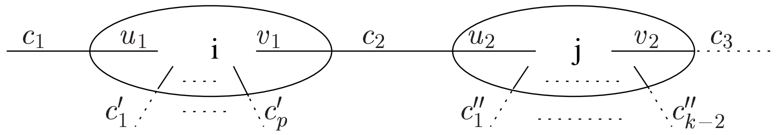
$u_1 + v_1 \geq v_2$, then we can exchange i and j, i.e. make eat the u_1 units of c_1 by j and the v_2 units of c_3 by i next fill in the holes by exchanging units of c_2.

becomes

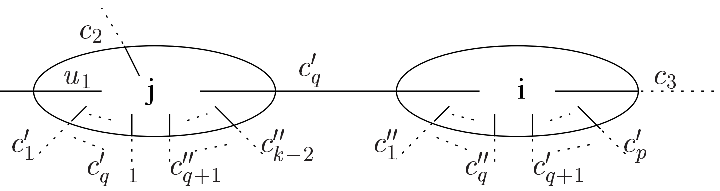


The exchange has been made on 3 colors and we have made it so that i and j are alone to see one of the 3 so both see the same number of colors as before. We must remark here that if i sees only the colors c_1 and c_2 then $t(i) = u_1 + v_1 > v_2$ because i is the biggest interval. So the problem is settled if $k = 2$.

Assume now that $u_2 > u_1$ and $u_1 + v_1 < v_2$. By the preceding remarks, we can suppose that i sees a number p between 1 and $k-2$ of colors c'_x different that c_1 and c_2 and j sees $k-2$ colors c''_x different that c_2 and c_3.



We must now remark that p satisfies $\sum_{x=1}^{p} i(c'_x) + \sum_{x=p+1}^{k-2} j(c''_x) + u_1 + v_1 + u_2 > t(j)$ because $t(i) = \sum_{x=1}^{p} i(c'_x) + u_1 + v_1 \geq u_2 + v_2 + \sum_{x=1}^{k-2} j(c''_x) = t(j)$. Let q be the smallest number such that $\sum_{x=1}^{q} i(c'_x) + \sum_{x=q+1}^{k-2} j(c''_x) + u_1 + v_1 + u_2 \geq t(j)$. Since $u_1 + v_1 < v_2$, we will say at j to leave all its units of colors c_3,$c''_1, \ldots, c''_q$ to i in exchange of the units of c_1, c_2, $c'_1, \ldots, c'_{q-1}$ and one part of the units of c'_q of j.



Thus, each sees as many colors than before and j goes again in the good branch of i.

Thanks to those four transformations, we can choose an adjacent branch to i and put any interval in this branch, until i is a pseudo-leaf.

∎

Corollary 1 *Let (t,I,C) be an instance of P_l^k which possess a troublemaker and let $g \in I$ be the biggest troublemaker. If this instance is satisfiable then there is a solution such that g exhausts all the colors he sees except at most one and such that it sees at least two colors.*

Proof :

Thanks to the Theorem 3, we know that there is a solution such that g exhausts all the colors he sees except at most one. Assume that it does not exhausts any because it sees only one c_g. We can exchange all the units of the $k-1$ smallest colors against units of c_g. Thus, g sees k colors and exhausts $k-1$. The intervals that were seeing one of the k little colors see as many colors as before and those that were seeing two or more see less colors than before. Remark: with thoses exchanges, we have perhaps created cycles in G. ■

Remark: if there is a solution such that g exhausts all the colors it sees except at most one, we can suppose that the non-exhausted color is the biggest color seen by g.

Theorem 4 *The Algorithm 1 decides if an instance of P_l^k is satisfiable and its complexity is in $O(1)$ for l and k fixed provided that the $l-2k+1$ biggest intervals are given sorted at the end of the array. We can decide the problem P_l^k in time $O(n)$, where n is the number of intervals.*

Proof :

The Algorithm 1 starts by looking if the number of intervals is 0 or if the total number of colors is less than k. If that is the case, we return "TRUE" because all colorings are acceptable. For deciding in constant time if there is a troublemaker, we check then if the biggest interval g is one. If there is no troublemaker, we return "TRUE" because any linear coloring is acceptable. Next, we look if there is k colors ables to cover g. If g is too big, we return "FALSE". If g is not too big and $l-2k<0$ then we return "TRUE" because we can exhaust $k-1$ colors in g and they will be at most k left. This first stage is done in constant time α.

We know, by the Corollary 1, that if there is a solution, there is one where g exhausts all the colors it sees except perhaps the biggest. In the worst case, g never exhausts the biggest color it sees, we must test all the possibilities and we do not stop before $l-2k<0$ for all the possibilities. Let $T(l,k)$ be the complexity in the worst case of the algorithm for the values l and k. If $l-2k<0$, $T(l,k)=\alpha$ where alpha is a constant. Else, $T(l,k)=\alpha+\sum_{p=2}^{k} C_l^p T(l-p+1,k)$. Since $l \geq 2k$, $C_l^p \leq C_l^k$ for $p=2..k$. So we have $T(l,k) \leq \alpha+\sum_{p=2}^{k} C_l^k T(l-p+1,k) \leq \alpha+\sum_{p=2}^{k} C_l^k T(l-1,k) = \alpha+(k-1)C_l^k T(l-1,k) < \alpha+(k-1)l^k T(l-1,k)$. If $l=2k+q$, $T(l,k)<\alpha\sum_{i=0}^{q}((k-1)l^k)^q < \alpha((k-1)l^k)^l$. We obtain a constant with respect to n because we never touch the array containing the intervals; each recursive call only look at one different cell of the array depending on its recursion depth. Finally, we obtain the linear complexity because $l-2k+1$ passes are sufficient for partially sorting the intervals. ■

We obtain a linear complexity $an+b$ with two constants. The multiplicative one $a \approx l-2k$ is linear in l and k and for l fixed is increasing when k is decreasing, i.e. when we come closer to the NP-complete part. The additive one $b \approx ((k-1)l^k)^l$ is exponential in l and k but

Algorithm 1 `Color`: Decision algorithm for P_l^k

input:

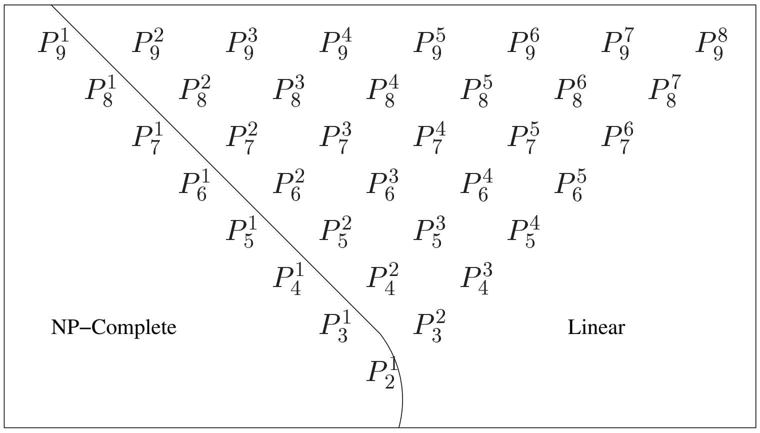
(l, k, I, C, n) where (I, C) is an instance of P_l^k, $|I| \geq n$, C is sorted in size increasing order and I is partially sorted in this order, i.e. that at least the $l - 2k + 1$ biggest intervals are sorted at the end of the array.

output:

TRUE: if there is a coloring such that each interval sees at most k colors.

begin
  if $n = 0$ or $l \leq k$ **then**
    return TRUE
  L':=2;
  forall m=1..k-1 **do**
    L'+=C[m];
  if I[n] < L' **then**
    return TRUE
  L:=0
  forall m=0..k-1 **do**
    L+=C[l-m];
  if I[n] > L **then**
    return FALSE
  if $l - 2k < 0$ **then**
    return TRUE
  forall $C' \in \mathcal{P}_{\geq 2, \leq k}(C)$ **do**
    c:= the biggest color in C'
    if $t(C') - t(c) < I[n]$ **then**
      if $t(C') > I[n]$ **then**
        c' := the rest of the color c of size $t(C') - I[n]$
        if Color($l - |C'| + 1$,k,I,$C \backslash C' \cup c'$,$n - 1$) **then**
          return TRUE
      if $t(C') = I[n]$ **then**
        if Color($l - |C'|$,k,I,$C \backslash C'$,$n - 1$) **then**
          return TRUE
  return FALSE
end

this time it is decreasing when we come closer to the NP-complete part. I do not know if this phenomenon is due to a bad approximation or if this is really what happens in the worst case. Nevertheless, I do not think that this phenomenon is still happening when we study the average complexity.
Thanks to the Theorem 4, we can conclude that we have the following complexity distribution:



Hence there is a violent threshold effect for the complexity of the problem between $k = 1$ and $k > 1$.

Theorem 5 *The problem P_l^k is NP-complete if $k = 1$, linear otherwise.*

0.7 Conclusion

In this article, we have brought to the fore the existence of a threshold effect for the complexity of neighbours combinatorial problems. This phenomenon formalised by the Theorem 5 can be seen as a kind of combinatorial analogous to the Schaeffer's theorem which separate the sub-problems of SAT in NP-complete or linear problems. The Algorithm 1 can be used to find for a given instance the smallest $k \neq 1$ such that an acceptable coloring exists in $\log(l)$ steps by dichotomic search. For this search, we use the same instance but different problems P_l^k. Another application can be to find an estimation of the best time for the execution of the programs I and a corresponding solution when each program is executed on at most k processors. This time we will always use the same problem P_l^k but we will change the instance by modifying the number of vertices uncovered by hyperedges (by changing t). We can remark that if $t > kt(I)$ then all the intervals can be covered by one color. So, we can do a dichotomic search between $t(I)$ and $kt(I)$ and in $\log(t(I)) + \log(k-1)$ steps find this estimation. The average complexity should be studied because the worst case analysis make us exhaust only one color at a time and when k is increasing the average number of exhausted colors should increase also.

In [9], I have proved that the problem P_3^2 is still linear with thickness 2. If the NP-completeness results are trivially true with superior thickness, it may not be the case for the linear results. It may be interesting to see if the border between NP-hard and polynomial cases is still the same with other thickness. Another interesting point would be to see if the problems P_l^k are always linear or NP-complete, or if some intermediate complexity shows off.

Bibliography

[1] U.V. Çatalyürek. *Hypergraph models for sparse matrix partitioning and reordering.* Computer engineering and information science, Bilkent University, November 1999.

[2] J.T. Bradley, N.J. Dingle, W.J. Knottenbelt, and H.J. Wilson. Hypergraph-based parallel computation of passage time densities in large semi-Markov models. *Linear Algebra and Its Applications*, July 2004.

[3] C.J.Alpert and A.B.Kahng. Recent directions in netlist partitionning: A survey. *VLSI Journal*, 19(1-2):1–81, 1995.

[4] D.Karger. Global min cuts in RNC, and other ramifications of a simple min-cut algorithm. In *Proc. ACM/SIAM Symp. on Discrete Algorithms (SODA 93)*, pages 21–30, 1993.

[5] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco, 1979.

[6] G.Karypis, E.Han, and V.Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[7] G.Karypis and V.Kumar. Multilevel k-way hypergraph partitioning. Technical Report 98-036, University of Minnesota, 1998.

[8] H.Zha, X.He, C.Ding, H.Simon, and M.Gu. Bipartite graph partitioning and data clustering. In *ACM International Conference on Information and Knowledge Management (CIKM 2001)*, 2001.

[9] L.Lyaudet. Largeur de branche des graphes de cordes. Master's thesis, ENSLyon, LIP, 2004. http://www.ens-lyon.fr/LIP/Pub/DEA2004.php.

[10] M.R.Salavatipour. A $(1+\epsilon)$-approximation algorithm for partitioning hypergraphs using a new algorithmic version of the Lovasz local lemma. *Random Structures and Algorithms*, 25(1):68–90, 2004.

[11] R.Cooley, B.Mobasher, and J.Srivastava. Web mining: Information and pattern discovery on the world wide web. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI '97)*, pages 558–567, Newport Beach, 1997.

[12] S.Shekhar and D.R.Liu. Partitioning similarity graphs: A framework for declustering problems. *Information Systems Journal*, 21(4), 1996.

[13] U.V.Çatalyürek and C.Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10:673–693, 1999.