NP-hard and linear variants of hypergraph partitioning

Laurent Lyaudet

LIP (UMR CNRS, ENS Lyon, INRIA, Univ. Claude Bernard Lyon 1),
École Normale Supérieure de Lyon,
46 allée d'Italie 69364 LYON cedex 07 FRANCE
Laurent.Lyaudet@ens-lyon.fr

Abstract This article presents an infinite family of combinatorial problems that shows abrupt changes of complexity between neighbour problems. We define problem P_k^l as a purely constraint-driven variant of hypergraph partitioning with parameters k and l as follows: Given a hypergraph on n vertices and k sizes of colours $t_1, \ldots, t_k$ of sum n, can we colour the vertices with k colours of given size such that each hyperedge intersects at most l colours? We show that, for fixed parameters k and l, P_k^l is: polynomial when $l = 1$, and NP-complete when $l \neq 1$ on the class of hypergraphs; NP-complete when $l = 1$, and linear when $l \neq 1$ on the class of hypergraphs with pairwise disjoint hyperedges. This inversion of complexity is possible since hypergraphs with disjoint hyperedges can be encoded in a more compact way, namely $\Theta(m \log(n))$ instead of $\Theta(mn)$ bits (n and m are the number of vertices and edges of the hypergraph).

1 Introduction

In this article, we present a family of parametrised combinatorial problems which are very natural and have various complexities. These problems are variants of the hypergraph partitioning problem. The hypergraph partitioning problem consists in finding a partition of the vertex set of the hypergraph such that a number of constraints are satisfied and/or an objective function is minimised. In general, this problem is NP-hard. A notable exception is the MinCut Bipartition problem that is polynomial (see, e.g. [8]). Variants of hypergraph partitioning are used in many areas, like VLSI design [1], efficient storage of large databases on disks [16], operations on sparse matrices [4,5], information retrieval [17], data mining [6,9], and study of semi-Markov processes [3]. Practical applications need the partition classes to be of almost equal size. For this reason, almost all these variants are NP-hard since one can reduce the MinCut Bisection problem to them (MinCut Bisection problem is to minimise the weight of a cut between a bisection, i.e. a bipartition into two parts of equal size). Many heuristic algorithms have been developed over the last thirty years (see, e.g. [10]) but, to my knowledge, no approximation algorithm (except [14]) or PTAS is known for classical variants of this problem even on hypergraphs subclasses.

The variants of hypergraph partitioning we will study are purely constraint-driven. Given two parameters k and l, $1 \leq l < k$, the problem is defined as follows: Let $\mathcal{H} = (V, \mathcal{E})$ be a hypergraph and $t_1, \ldots, t_k$ be non-negative integers such that $|V| = n = \sum_{i=1}^{k} t_i$ and $|\mathcal{E}| = m$. Does there exist a colouring (partition) of V in k subsets of size $t_1, \ldots, t_k$ such that the vertices of each hyperedge in $\mathcal{E}$ are coloured with at most l colours?[1] We will denote this decision problem by P_k^l.

The problem P_3^2 is strongly related to the branchwidth problem on graphs (see [11], [12]). In [11], Kloks *et al.* show that P_3^2 is NP-complete, proving in particular that the branchwidth problem is NP-complete on splitgraphs and bipartite graphs.

In a first part, we will study the complexity of these problems on arbitrary hypergraphs. The input size of an instance is in $\Theta(nm + k\log(n)) = \Theta(nm)$ bits. We will then show that P_k^1 is solvable in time $O(nm+n^k)$ and P_k^l is NP-complete for any $k > l \geq 2$. We can suggest the following interpretation of this problem: Given a set of nodes that represent the tables of a database, a set of hyperedges each representing the tables needed for a query, and the storage capacity of the servers, can we store the database on the servers such that each query needs only data from at most l servers? Here we make the improper assumption that all the tables have the same size; but it proves that the "real life" problem is NP-complete.

In a second part, we investigate the complexity of these problems on a restricted subclass of hypergraphs, namely the hypergraphs with disjoint hyperedges (or maximum degree 1). Although the structure might seem very simple, the interest for the problem on this subclass is twofold: firstly the input size is very compact in $\Theta(m\log(n) + k\log(n)) = \Theta(m\log(n))$ bits because we only have to give the size of each hyperedge (see details in Section 4), secondly it corresponds to the following natural scheduling (or packing) problem.

Consider a set of m programs, each program being associated to a natural number that represents the number of unitary tasks that communicate together to achieve the program. Consider also a set of k fixed processors and let $t_1, \ldots, t_k$ be the "computing capacity" of these processors. Let n be the sum of the t_is. Can we execute the tasks on k processors such that each program is dispatched on at most l processors?

Here we will have complexity results that are inverse to those of the first part. We will show that P_k^1 is NP-complete for all k, and P_k^l, for $l \geq 2$, is solvable in time less than $m + f(k,l)$ where $f(k,l)$ is exponential in k and l. Hence the problem P_k^l is FPT (Fixed Parameter Tractable) for parameters k and l when $l \geq 2$.

In Section 2, we introduce the notations and recall the notions of weak and strong polynomiality (resp. NP-completeness). In Section 3, we show that the problem P_k^l is polynomial when $l = 1$ and NP-complete when $l > 1$ for arbitrary hypergraphs. The section 4 is on hypergraphs with disjoint hyperedges and is

[1] Note that k and l are not part of the input, and that the coloring does not have to be proper.

divided in four parts: first the proof of NP-completeness of P_k^1, then a first result of linearity of P_k^l for $k - 2l < 0$, followed by a study of the structure of the solutions ending up with the proof of linearity for $l \neq 1$, and finally we give applications of the linear time algorithm to related optimisation problems.

2 Preliminaries

We recall that a hypergraph $\mathcal{H}$ is a couple $(V, \mathcal{E})$ where V is the set of vertices and $\mathcal{E}$ is the set of hyperedges. Each hyperedge is a subset of V. In the rest of this article, $t(e)$ will be the size of the hyperedge e, $t(c)$ the size of the colour c and $c(e) = e(c) = |c \cap e|$ the number of units of the colour c in e. We will say that the hyperedge e "sees" the colour c or reciprocally that the colour c sees the hyperedge e if $c(e) = e(c) > 0$. C denotes the set of colours, $C(e)$ the set of colours that are seen by e, and $E(c)$ the set of hyperedges that are seen by c. We will say that a colouring of the vertex set is admissible if the colours have the specified sizes and each hyperedge sees at most l colours. We will use $\mathcal{P}_{\geq a, \leq b}(X)$ for denoting the set of X's subsets with cardinal between a and b.

Weak and strong polynomiality (resp. NP-hardness) are defined as follows. Consider a problem P whose instances contain some integers as a part of their definition. Let n be the number of bits of an input and $maxI$ be an upper bound on the integers in the input. P is said weakly (resp. strongly) polynomial if there is a polynomial time algorithm solving P on instances such that $maxI \leq f(n)$, for all polynomial function $f(n)$ (resp. on all instances). P is said weakly (resp. strongly) NP-hard if some NP-hard problem reduces to P without any assumption (resp. assuming $maxI \leq f(n)$, for some polynomial function $f(n)$).

3 Complexity of the problem on arbitrary hypergraphs

In this section, we first prove that P_k^1 is solvable in polynomial time and then show that P_k^l is NP-complete for any $l \geq 2$.

3.1 The polynomial case

Theorem 1 *The problem P_k^1 is solvable in time $O(nm + n^k)$, for any fixed $k \geq 2$.*

Proof :

As each hyperedge must see at most one colour, if two hyperedges have a non-empty intersection they must see the same colour. Hence if $A = (\mathcal{H} = (V, \mathcal{E}), t_1, \ldots, t_k)$ is an instance of P_k^1 such that $\exists e, e' \in \mathcal{E}$ which intersect, then A is positive if and only if $A' = (\mathcal{H} = (V, \mathcal{E} \backslash \{e, e'\} \cup \{e \cup e'\}), t_1, \ldots, t_k)$ is positive. So we can repeat this fusion of hyperedges until we obtain a hypergraph with disjoint hyperedges. Given a hypergraph, it is easy to see that we can obtain the corresponding hypergraph with disjoint hyperedges in time $O(nm)$. Now on a hypergraph with disjoint hyperedges, the problem P_k^1 is as follows:

Can we find a partition $(E_1, \dots, E_k)$ of $\mathcal{E}$ such that $\sum_{e \in E_i} t(e) \leq t_i, i = 1..k$? This question can be solved as an instance of the k-Subset-Sum problem in time $O(n^k)$ (see [7]). Hence the problem P_k^1 can be solved in time $O(nm+n^k)$.

■

3.2 The NP-complete case

Theorem 2 ([11], theorem 1) *P_3^2 is NP-hard on instances with colours of equal size.*

Surprisingly enough a direct generalisation of the NP-hardness proof of P_3^2 in [11] will show the NP-hardness of P_k^{k-1} only if k is a prime number.

Lemma 1 *If P_k^2 is NP-hard on instances with colours of equal size, then P_{k+1}^2 is also NP-hard on instances with colours of equal size.*

Proof :

Let $A = (\mathcal{H} = (V, \mathcal{E}), t_1 = \cdots = t_k = q)$ be an instance of P_k^2, $|V| = kq$. We construct the instance $A' = (\mathcal{H}' = (V', \mathcal{E}'), t_1 = \cdots = t_k = t_{k+1} = q)$ of P_{k+1}^2 with $V' = V \cup X$ and $\mathcal{E}' = \mathcal{E} \cup \{X\}$, where $V \cap X = \emptyset$ and $|X| = q$. If A is a positive instance, then we consider an admissible colouring of $\mathcal{H}$. We can extend this colouring by colouring X with the new colour and obtain an admissible colouring of $\mathcal{H}'$. If A' is a positive instance, then we consider an admissible colouring of $\mathcal{H}'$. If this colouring uses only one colour in X, then its restriction to V is an admissible colouring of $\mathcal{H}$. Otherwise, as X is a hyperedge, it sees at most two colours c_1 and c_2 and as $|X| = q$, there is the same number of units of c_2 (vertices c_2-coloured) in X as the number of units of c_1 in V. If we exchange all the units of c_2 in X with all the units of c_1 in V, then X sees now only one colour, and the other hyperedges in V see no more colours than before. This new colouring restricted to V is an admissible colouring of $\mathcal{H}$. It is straightforward to see that A' can be constructed from A in polynomial time and so the result follows.

■

Thanks to Theorem 2, we have the following corollary.

Corollary 1 *P_k^2 is NP-hard on instances with colours of equal size, $\forall k \geq 3$.*

Lemma 2 *If P_k^l is NP-hard on instances with colours of equal size, then P_{k+1}^{l+1} is also NP-hard on instances with colours of equal size.*

Proof :

Let $A = (\mathcal{H} = (V, \mathcal{E}), t_1 = \cdots = t_k = q)$ be an instance of P_k^l, $|V| = kq$. We construct the instance $A' = (\mathcal{H}' = (V', \mathcal{E}'), t_1 = \cdots = t_k = t_{k+1} = q)$ of P_{k+1}^{l+1} with $V' = V \cup X$ and $\mathcal{E}' = \{e \cup X \mid e \in \mathcal{E}\}$, where $V \cap X = \emptyset$ and $|X| = q$.
If A is a positive instance, then we consider an admissible colouring of $\mathcal{H}$. We can extend this colouring by colouring X with the new colour and obtain an admissible colouring of $\mathcal{H}'$ because each hyperedge in $\mathcal{E}'$ sees at most l colours in its restriction to V and one colour in its restriction to X.

If A' is a positive instance, then we consider an admissible colouring of $\mathcal{H}'$. If this colouring uses only one colour in X, then its restriction to V is an admissible colouring of $\mathcal{H}$. Otherwise, as X is included in all the hyperedges of $\mathcal{E}'$, all the hyperedges see the colours in X. Let c_1 be one of these colours. We can exchange all the units of c_1 in V with the units of the other colours in X. This way the number of colours seen by the hyperedges does not increase. This new colouring restricted to V is an admissible colouring of $\mathcal{H}$. It is straightforward to see that A' can be constructed from A in polynomial time and so the result follows. ■

Theorem 3 *P_k^l is NP-complete, $\forall 2 \leq l < k$.*

Proof :

Given a colouring of a hypergraph, we can check in polynomial time if it is an admissible colouring so P_k^l is in NP. According to Corollary 1, P_{k-l+2}^2 is NP-hard. Using $l-2$ times Lemma 2, we obtain the NP-hardness of P_k^l and so P_k^l is NP-complete. ■

By theorems 1 and 3, we see that we have a complexity distribution with polynomial problems on the left border and NP-complete problems everywhere else:

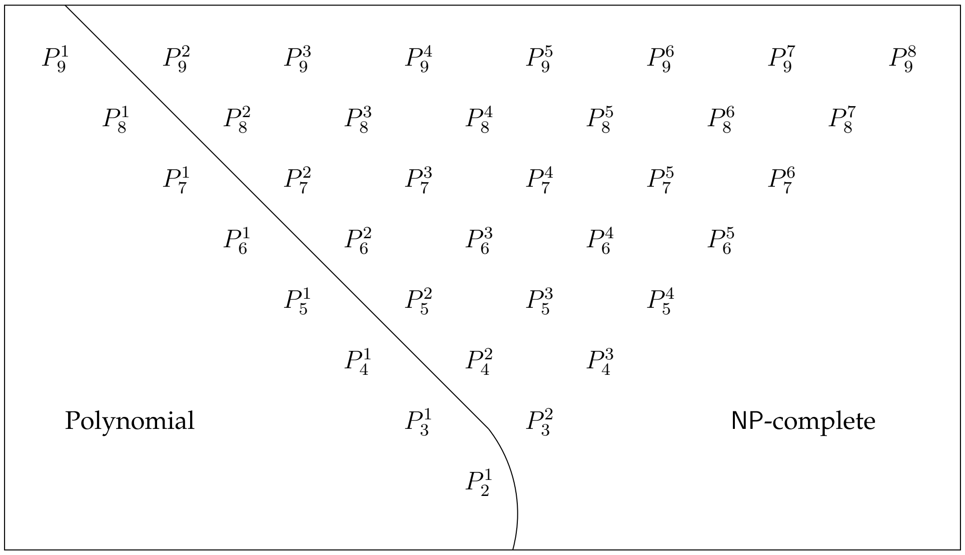

P_9^1 P_9^2 P_9^3 P_9^4 P_9^5 P_9^6 P_9^7 P_9^8
P_8^1 P_8^2 P_8^3 P_8^4 P_8^5 P_8^6 P_8^7
P_7^1 P_7^2 P_7^3 P_7^4 P_7^5 P_7^6
P_6^1 P_6^2 P_6^3 P_6^4 P_6^5
P_5^1 P_5^2 P_5^3 P_5^4
P_4^1 P_4^2 P_4^3
Polynomial
P_3^1 P_3^2
NP-complete
P_2^1

Figure 1. Complexity on general hypergraphs.

4 Complexity of the problem on hypergraphs with disjoint hyperedges

When the hyperedges are disjoint, it is convenient to see the hypergraph as an interval hypergraph. Indeed we can find a total order on V such that $\forall e \in \mathcal{E}, \forall x, y \in e$, either $x \leq y$ and $\forall z$ such that $x \leq z \leq y, z \in e$, or $y \leq x$ and $\forall z$ such that $y \leq z \leq x, z \in e$. V can be seen as an interval and the hyperedges can be seen as disjoint sub-intervals of V. For this reason, we will rename $\mathcal{E} = I$ and talk about intervals rather than hyperedges for the rest of this section. With this representation, it is clear that we define the hypergraph if we give the "left" and "right" extremities of each interval. So we have a compact encoding in $\Theta(m \log(n))$ bits.

We can notice that the problem has a dual nature when we consider disjoint hyperedges. Indeed, we can see I and C as two families of disjoint hyperedges on V. The decision problem can be rephrased as follows: Given two families of hyperedges I and C that are disjoint and given the size of each hyperedge, is there a positioning of these hyperedges such that every hyperedge of I is adjacent to at most l hyperedges of C?

In the rest of the section, we will assume that comparisons, additions, and subtractions on integers are done in constant time. Polynomial complexities will therefore be independent of n.

4.1 The NP-complete case

The following theorem is obtained by a reduction from the 2-Partition problem.

Theorem 4 *The problem P_k^1 is NP-complete on instances with colours of equal size, $\forall k \geq 2$.*

Proof :

The problem P_k^1 consists in finding a partition of V in k colours such that each element of I sees exactly one colour. Recall that an instance of the 2-Partition problem consists in a set of m natural numbers $e_1, \ldots, e_m$ whose sum is n even. It is encoded in size $m \log(n)$. The problem has a solution if and only if we can find $m' < m$ natural numbers in $e_1, \ldots, e_m$ whose sum is $n/2$. We will transform an instance of the 2-Partition problem in an instance of P_k^1 by associating to each natural number e_i a sub-interval of size e_i of an interval V of size $n + \frac{(k-2)n}{2}$, adding $k-2$ sub-intervals of size $n/2$, and fixing $t_1 = t_2 = \cdots = t_k = n/2$. This instance is of size $\Theta((m+k-2)\log(n+\frac{(k-2)n}{2})+k\log(n)) = \Theta(m \log(n))$ bits because k is fixed; it can be constructed in polynomial time. It is obvious that the constructed instance is positive if and only if the instance of 2-Partition is, hence P_k^1 is NP-complete.

■

4.2 The linear case

Now that we know that problems on the left border of the family are NP-complete, we will be interested in the members of the other border. We first show that problems in the right half ($k - 2l < 0$) have a linear complexity.

Definition 1 (t_p^{max}, t_p^{min}) *Given an indexing of the colours ordered by increasing size, $c_1, c_2, \ldots, c_k$ such that $t(c_1) \leq t(c_2) \leq \cdots \leq t(c_k)$, we will use the following notations $t_r = t(c_r)$, $t_p^{min} = \sum_{r=1}^{p} t_r$ the size of the p smallest colours, and $t_p^{max} = \sum_{r=k-p+1}^{k} t_r$ the size of the p largest colours.*

We put $L := t_l^{max}$ to denote the limit size. It corresponds to the maximal size that can be covered by l colours. Hence, if there is an interval whose size exceeds L, there is no admissible colouring. This gives an unconditional obstruction.

Definition 2 (megalomaniac) *A megalomaniac is an interval whose size exceeds the limit size L.*

We lean on a particular type of colouring for reasoning:

Definition 3 *Given a total order on V such that $\mathcal{H} = (V, I)$ is an interval hypergraph for this order, we will say that a colouring of V is* continuous *relatively to this order if all the colours are intervals except maybe one that is the union of an interval containing the minimum of the order and an interval containing the maximum of the order. A colouring is said to be* continuous *if there is an order on V such that it is continuous relatively to this order.*

We say that an interval i exhausts a colour c if all the c-coloured elements of V are contained in i. Suppose we colour V continuously beginning with colour 1, then with colour 2 when we have exhausted the colour 1, ... This procedure fails if an interval exhausts at least $l - 1$ colours and it sees two more colours (one on the "left" and one on the "right"). This gives a conditional obstruction:

Definition 4 *A* troublemaker *is an interval that can exhaust at least $l - 1$ colours and see two more colours,* i.e. *it has size at least $L' := t_{l-1}^{min} + 2$.*

Remark 1. If there is a troublemaker, then the largest interval is a troublemaker. When there is no troublemaker, any continuous colouring is admissible; but we can use a troublemaker if there is one with the following proposition.

Proposition 1 *Let g be a troublemaker that is not megalomaniac. Then there is a partial colouring of V restricted to the vertices of g that colours g with l colours and exhausts $l - 1$ of these colours.*

Proof :

We start with the $l - 1$ smallest colours and add the largest colour. If these l colours are not sufficient to cover g, we change the smallest used colour for the largest unused colour. We repeat these changes until the sum of the sizes of the l used colours is larger than the size of g. As $t(g) \leq L$ this procedure will always stop with l used colours that are sufficient to cover g. Moreover we know that the $l - 1$ small colours are smaller than g so we can exhaust them.

■

The following result when $k - 2l < 0$ is an easy case of linearity. It will be used as a base case in the general algorithm.

Theorem 5 *If $k - 2l < 0$, then an instance of P_k^l is positive if and only if there is no megalomaniac. Hence P_k^l can be decided in linear time.*

Proof :

$\Rightarrow$: By contraposition, if there is a megalomaniac, there is no admissible colouring.
$\Leftarrow$: If there is no troublemaker, any continuous colouring is admissible. If there is one, say g, we start a continuous colouring from its border using the l colours found in Proposition 1 (in the order of increasing size). This way g is correctly coloured and there is at most $k - (l - 1) \leq l$ colours that are outside of g. Hence the remaining intervals are correctly coloured.

It shows that it is sufficient to test if the condition on the limit size is satisfied for all the intervals, for answering the decision problem. If we want to give an admissible colouring, we just need to colour continuously starting from the border of a troublemaker if there is one. Finding l colours such that l cover and $l-1$ can be exhausted in a troublemaker can be done in constant time (l is fixed). Hence, the problem is linear. ■

For now we have the following distribution:

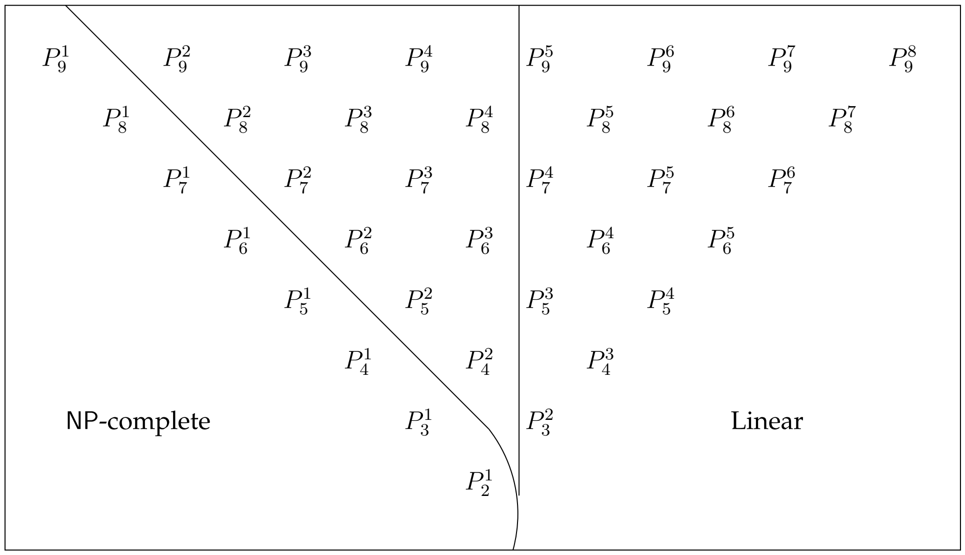

P_9^1 P_9^2 P_9^3 P_9^4 P_9^5 P_9^6 P_9^7 P_9^8
P_8^1 P_8^2 P_8^3 P_8^4 P_8^5 P_8^6 P_8^7
P_7^1 P_7^2 P_7^3 P_7^4 P_7^5 P_7^6
P_6^1 P_6^2 P_6^3 P_6^4 P_6^5
P_5^1 P_5^2 P_5^3 P_5^4
P_4^1 P_4^2 P_4^3
NP-complete
P_3^1 P_3^2
Linear
P_2^1

Figure 2. Partial complexity on hypergraphs with disjoint hyperedges.

We obtained this first result of linearity thanks to the notion of troublemaker. More exactly, using the fact that we can exhaust enough colours in a single troublemaker so that other intervals will necessarily be coloured with at most l colours.

We will generalise this result using the structure of the solutions. 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.

4.3 Structure of the solutions

All our approach comes down to exhaust colours until there are at most l left. If we prove that every satisfiable instance has a solution in which some interval exhausts at least one colour, then the problem is weakly polynomial.

Why polynomial? Indeed, we can construct an algorithm that will exhaust at least one colour at each step; hence $k - l$ steps are sufficient before the algorithm stops. At each step, we can do exhaustive search and generate for all intervals all the partial colourings that are exhausting one colour in this interval. Then we check if the resulting instance of $P^l_{k'}$ is satisfiable ($k' < k$). Each sub-problem at a given step will generate $O(m)$ (choice of the interval) multiplied by $O(\sum_{r=1}^{l} \binom{k}{r})$ (choice of the set of colours colouring the interval) sub-problems for the next step. We obtain an algorithm with complexity relatively to m with order $(\sum_{r=1}^{l} \binom{k}{r})^{k-l} \times m^{k-l}$, which is $O(m^{k-l})$ (assuming that k and l are fixed).

Why weakly? If we take into account the size n and all the sizes of the colours and intervals that are of the same order of magnitude, we remark that we encode them in $\log(n)$-space. But if $l \geq 3$, the interval i that we chose can exhaust one colour and see two others without exhausting them. It means that the excess of these two colours compared to the size of i can be dispatched in $\Theta(n)$ different ways between them (when l grows, it becomes worse since the excess can be dispatched in $\Theta(n^{l-2})$ different ways). We may end up with a number of sub-problems that is exponential in $\log(n)$.

In order to have a strongly polynomial algorithm we must ensure a stronger result. We must prove that every satisfiable instance has a solution in which an interval exhausts all the colours it sees except at most one (of course, it must also see at least two colours to exhaust at least one).

The following theorems will prove that we are in the situation of strong polynomiality. They will reveal properties of the following graph that models the structure of a colouring.

Definition 5 *Given a colouring, we call* colouring 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 colour c sees the interval i. On each edge we can add a weight corresponding to the number of units shared by the concerned interval and colour. We denote these weights by $w(i, c) = i(c) = c(i)$.*

We first list some obvious propositions on the properties of a colouring graph ($N(v)$ denotes the set of neighbours of v in G).

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

Proposition 3 *Let G be a colouring graph. If all elements of V are covered by an interval in I, then $\sum_{i \in N(c)} w(i,c) = t(c), \forall c \in C$.*

The first theorem gives a property of "good" solutions (solutions minimising $\sum_{i \in I} |C(i)|$, the sum of the numbers of colours seen by the intervals, under the constraints $|C(i)| \leq l, \forall i$).

Theorem 6 *Let $A = (\mathcal{H} = (V,I), C)$ be an instance of P_k^l. If this instance is satisfiable, then there is an admissible colouring such that its graph is a forest.*

Proof :

Let $G = (I \cup C, E)$ be the corresponding graph of any admissible colouring. Suppose G contains an elementary cycle $i_1, c_1, \ldots, i_r, c_r$. Let $\min\{w(i_j, c_j)\} = w(i_x, c_x) = p$. Then for every $j = 1, \ldots, r$ recolour p vertices of colour c_j in interval i_j into colour c_{j-1} (indices are taken by modulo r). We obtain the graph without edge $i_x c_x$. In analogous way we can destroy all the cycles in G and obtain an admissible colouring such that the corresponding graph is a forest. ∎

The following theorem ensures that the complexity is strongly polynomial. We can not prove[2] that all "good" solutions satisfy the following property but the proof of the theorem will show that there exists at least one "good" solution satisfying this property.

Theorem 7 *Let $A = (\mathcal{H} = (V,I), C)$ be an instance of P_k^l and $i \in I$ be an interval of maximum size. If this instance is satisfiable, then there is a solution such that i exhausts all the colours that it sees except for at most one.*

Proof :

Assume that the instance is satisfiable. We consider the colouring graph G of an admissible colouring satisfying the conditions of Theorem 6. G is a forest. If i is the only interval in its connected component, it exhausts all the colours it sees so the statement is true. Else we want to make i a pseudo-leaf i.e. all the adjacent colours to i except for one are leaves. Assume that i is not alone in its connected component and it is not a pseudo-leaf. In order that i becomes a pseudo-leaf, we will put all the other intervals of the component in one unique branch starting from i.

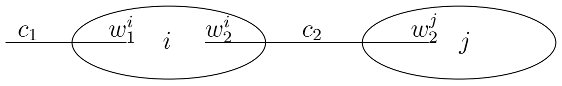
We consider the couple (cc, sb) where cc is the number of connected components of the colouring graph and sb is the maximum number of vertices

[2] Consider the following instance of P_3^2: $t(c_1) = t(c_2) = 5$, $t(c_3) = 1$, $t(i_1) = t(i_2) = 3$, $t(i_3) = 4$, $t(i_4) = 1$. Clearly $w(i_1, c_1) = 3$, $w(i_2, c_2) = 3$, $w(i_3, c_1) = w(i_3, c_2) = 2$, and $w(i_4, c_3) = 1$ is a "good solution"; but the largest interval i_3 does not satisfy the property.

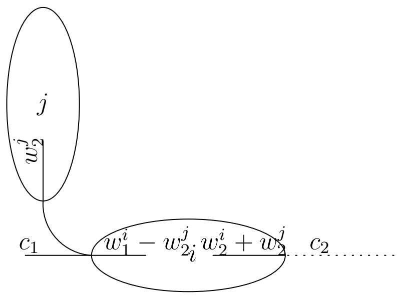
(corresponding to intervals of $\mathcal{H}$) in a branch starting from i. We call max-branch for i such a branch. In the following we use four transformations such that the resulting colouring has either one more connected component, or one of the max-branch for i contains at least one more vertex corresponding to an interval. Thus, these transformations will give colouring graphs with strictly increasing value of (cc, sb), according to the lexicographic order.

We visualise colours as semi-rigid wires connecting the intervals. In the following figures, colours and intervals are respectively represented by lines and ellipses. Sentences in italic font correspond to the explanation of the continuous distortions we visualise. The proof can be read without these explanations.

Let c_1 and c_2 be two colours seen by i and another interval. We consider that c_1 is in a max-branch for i. Let $j \neq i$ be an interval that sees c_2. *We will move i from c_1 to c_2.* Let us denote $w(i, c_1) = w_1^i$, $w(i, c_2) = w_2^i$, and $w(j, c_2) = w_2^j$.

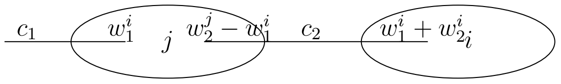


Transformation 1: *If $w_2^j \leq w_1^i$, then we can push all the units of the colour c_2 that are in j inside of i and make go out w_2^j units of c_1 in order to reconnect j behind i.* More formally, if $w_2^j \leq w_1^i$, then we exchange the w_2^j units of c_2 in j with w_2^j units of c_1 in i.

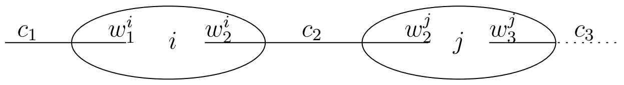


If $w_2^j = w_1^i$, this transformation has deconnected i from its max-branch along with the branch containing j and the number of connected components has increased. Otherwise we have increased by at least one the number of intervals (and thus of vertices) that are in the max-branch of i. In both cases j sees the same number of colours and i sees one colour less or the same number of colours than before.

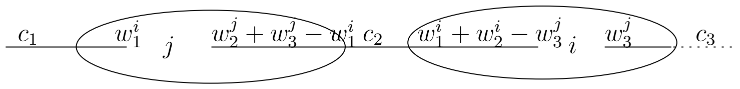
Transformation 2: If $w_2^j > w_1^i$ and j sees less than l colours then *we can exchange i and j positions*, i.e. exchange w_1^i units of c_1 in i with w_1^i units of c_2 in j. Thus, j goes again in the max-branch of i. j sees one more colour and i one less.



Transformation 3: Assume that j sees l colours then there exists a colour $c_3 \neq c_2$ seen by j. Let us denote $w(j, c_3) = w_3^j$. If $w_2^j > w_1^i$ and $w_1^i + w_2^i \geq w_3^j$, then *we can again exchange i and j positions*, i.e. give the w_1^i units of c_1 to j, give the w_3^j units of c_3 to i and balance the exchange with units of c_2. Formally, j will contain w_1^i units of c_1 and $w_2^j + w_3^j - w_1^i$ units of c_2 ($w_2^j + w_3^j - w_1^i > 0$ since $w_2^j > w_1^i$). i will contain w_3^j units of c_3 and $w_1^i + w_2^i - w_3^j$ units of c_2 ($w_1^i + w_2^i - w_3^j \geq 0$ since $w_1^i + w_2^i \geq w_3^j$).



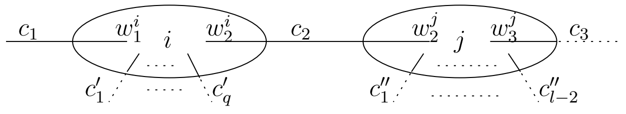
becomes



The exchange has been made on 3 colours and it is done so that i and j are alone to see one of the 3 so both see the same number of colours as before. If $w_1^i + w_2^i = w_3^j$ this transformation has deconnected i from its max-branch along with the branch containing j and the number of connected components has increased. Otherwise we have increased by at least one the number of intervals that are in the max-branch of i.

We must remark here that if i sees only the colours c_1 and c_2 then $t(i) = w_1^i + w_2^i > w_3^j$ because i is the largest interval. Hence the proof is already complete for the case $l = 2$.

Transformation 4: Assume now that $w_2^j > w_1^i$ and $w_1^i + w_2^i < w_3^j$. By the preceding remarks, we can suppose that i sees a number q between 1 and $l-2$ of colours c'_x different from c_1 and c_2 and j sees $l-2$ colours c''_x different from c_2 and c_3 (note that c'_x and c''_y are distinct , $1 \le x \le q$, $1 \le y \le l-2$, since the coloring graph is a forest).



We must now remark that q satisfies

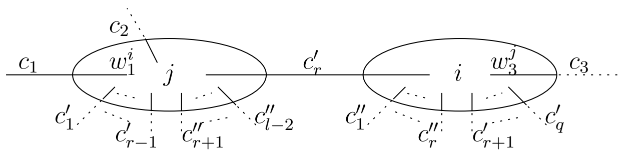
$$\sum_{x=1}^{q} i(c'_x) + \sum_{x=q+1}^{l-2} j(c''_x) + w_1^i + w_2^i + w_2^j > t(j) \text{ because}$$

$$\sum_{x=1}^{q} i(c'_x) + w_1^i + w_2^i = t(i) \ge t(j).$$

Let r be the smallest number such that

$$\sum_{x=1}^{r} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + w_1^i + w_2^i + w_2^j \ge t(j).$$

Since $t(j) = w_2^j + w_3^j + \sum_{x=1}^{l-2} j(c''_x)$ and $w_1^i + w_2^i < w_3^j$, we exchange all units of colours $c_3, c''_1, \ldots, c''_r$ in j with the units of $c_1, c_2, c'_1, \ldots, c'_{r-1}$ and one part (maybe all) of the units of c'_r in i.



i contains now w_3^j units of c_3, $j(c''_x)$ units of c''_x, $x = 1..r$, $i(c'_x)$ units of c'_x, $x = r+1..q$, and $\sum_{x=1}^{r} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + w_1^i + w_2^i + w_2^j - t(j)$ units of c'_r ($\sum_{x=1}^{r} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + w_1^i + w_2^i + w_2^j - t(j) \ge 0$ by definition of r). j contains now w_1^i units of c_1, $w_2^i + w_2^j$ units of c_2, $j(c''_x)$ units of c''_x, $x = r+1..l-2$, $i(c'_x)$ units of c'_x, $x = 1..r-1$, and $t(j) - (\sum_{x=1}^{r-1} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + w_1^i + w_2^i + w_2^j)$ units of c'_r ($t(j) - (\sum_{x=1}^{r-1} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + w_1^i + w_2^i + w_2^j) \ge 0$ since, by definition of r, $\sum_{x=1}^{r-1} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + w_1^i + w_2^i + w_2^j < \sum_{x=1}^{r-1} i(c'_x) + \sum_{x=r}^{l-2} j(c''_x) + w_1^i + w_2^i + w_2^j < t(j)$).

Thus, either $\sum_{x=1}^{r} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + w_1^i + w_2^i + w_2^j = t(j)$ and we have succeed in breaking the connected component of i in two parts and decreasing by one the number of colours seen by i, or $\sum_{x=1}^{r} i(c'_x) + \sum_{x=r+1}^{l-2} j(c''_x) + w_1^i + w_2^i + w_2^j > t(j)$, i and j see as many colours as before and j goes again in the max-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 2 *Let $A = (\mathcal{H} = (V, I), C)$ be an instance of P_k^l which possesses a troublemaker and let $g \in I$ be an interval of maximum size (hence it is a troublemaker of maximum size). If this instance is satisfiable, then there is a solution such that g sees at least two colours and exhausts all of them except maybe a colour c_i of maximum size t_i (maximum among the colours seen by g).*

Proof :

Thanks to Theorem 7, we know that there is a solution such that g exhausts all the colours it sees except at most one. Assume that it does not exhaust any because it sees only one c_g. Since g is a troublemaker, we can exchange all the units of the $l-1$ smallest colours against units of c_g. Thus, g sees l colours and exhausts $l-1$. The intervals that were seeing one of the $l-1$ little colours see as many colours as before and those that were seeing two or more see less colours than before. (With those exchanges, we have perhaps created cycles in the colouring graph; since g is a pseudo-leaf in the colouring graph, these cycles may be removed as in the proof of Theorem 6)

Assume that the non-exhausted colour, c_1, is not a colour of maximum size seen by g. Let c_2 be a colour of maximum size seen by g. Since $t(c_2) > t(c_1)$, we can exchange all units of c_1 outside of g with units of c_2 inside of g. The number of colours seen by g or any other interval is not modified. Hence we still obtain an admissible colouring. ■

The preceding corollary sums up four points relevant for the complexity of the algorithm:

- g exhausts at least one colour (it sees at least two); thus we will obtain a polynomial time algorithm in m;
- g exhausts all the colours it sees except at most one; this argument enables to prove a strongly polynomial time algorithm;
- we can exhaust one colour considering only the interval g (we do not have to try all possible intervals); we will deduce a linear complexity in m;
- the non-exhausted colour is a colour of maximum size seen by g, hence each set with at most l colours colouring g generate only one sub-problem. The complexity is slightly reduced by a factor $O(kl)$.

Algorithm 1 `Colour`: Decision algorithm for P_k^l

```
input:
    $(k, l, I, C, m)$ where $(I, C)$ is an instance of $P_k^l$, $|I| \geq m$, $C$ is sorted in
    size increasing order and $I$ is partially sorted in this order, i.e. that at
    least the $\max(1, k-2l+2)$ largest intervals are sorted at the end of the
    array. (The size of a set of colours is the sum of its colours sizes, not its
    cardinality.)
output:
    TRUE: if there is a colouring such that each interval sees at most $l$
    colours.

begin
    /* test trivial stop conditions */
    if $m = 0$ or $k \leq l$ then
        return TRUE
    L':=2;
    forall r=1..l-1 do
        L'+=C[r];
    /* If there is no troublemaker */
    if I[m] < L' then
        return TRUE
    L:=0
    forall r=0..l-1 do
        L+=C[k-r];
    /* If there is a megalomaniac */
    if I[m] > L then
        return FALSE
    if $k - 2l < 0$ then
        return TRUE /* by Theorem 5 */
    forall $C' \in \mathcal{P}_{\geq 2, \leq l}(C)$ /* for all set of 2 to $l$ colours*/ do
        c:= the largest colour in C'
        /* If $I[m]$ can exhaust almost all of the colours */
        if $t(C') - t(c) < I[m]$ then
            /* If these colours cover $I[m]$ and one colour is not exhausted*/
            if $t(C') > I[m]$ then
                $c'$ := the rest of the colour $c$ of size $t(C') - I[m]$
                if Colour($k - |C'| + 1$,$l$,$I$,$C \backslash C' \cup \{c'\}$,$m - 1$) then
                    return TRUE
            /* If these colours cover $I[m]$ and all colours are exhausted */
            if $t(C') = I[m]$ then
                if Colour($k - |C'|$,$l$,$I$,$C \backslash C'$,$m - 1$) then
                    return TRUE
    end forall
    return FALSE /* by Corollary 2 */
end
```

Theorem 8 *Algorithm 1 decides if an instance of P_k^l is satisfiable and its complexity is in $O(1)$ for l and k fixed provided that the $\max(1, k-2l+2)$ largest intervals are given sorted at the end of the array. We can decide the problem P_k^l in time $O(m)$, where m is the number of intervals.*

Proof :

Correctness: Algorithm 1 starts by looking if the number of intervals is 0 or if the total number of colours is less than l. If that is the case, we return "TRUE" because all colourings are admissible. For deciding if there is a troublemaker, we verify if the largest interval g is one. If there is no troublemaker, we return "TRUE" because any continuous colouring is admissible. Next, we look if there are l colours able to cover g. If g is megalomaniac, we return "FALSE". If g is not and $k-2l<0$, then we return "TRUE" by Theorem 5.

The recursive calls in the last loop try all possibilities for g to exhaust all the colours it sees except perhaps the largest. If any of these cases yields a positive instance with strictly less colours, then we return "TRUE". Otherwise we return "FALSE". The correctness follows by Corollary 2. (Note that each recursive call is valid since the following invariants are preserved: $\sum_{i=1}^{m} I[i] \leq \sum_{c\in C} t(c)$ and $|C| = k$.)

Analysis of complexity: We first bound the size of the recursive calls tree. We know, by Corollary 2, that if there is a solution, there is one where g exhausts all the colours it sees except perhaps the largest. In the worst case, g never exhausts the largest colour it sees, we must test all the possibilities, and we do not stop before $k-2l<0$ for all the possibilities. Let $T(k,l)$ be the size of the recursive calls tree of the algorithm, in the worst case, for the values k and l.

If $k-2l<0$, $T(k,l)=1$. Else, $T(k,l) = 1 + \sum_{p=2}^{l} \binom{k}{p} T(k-p+1,l)$.

Since $k \geq 2l$, $\binom{k}{p} \leq \binom{k}{l}$ for $p = 2..l$. So we have

$$\begin{aligned} T(k,l) &\leq 1 + \sum_{p=2}^{l} \binom{k}{l} T(k-p+1,l) \\ &\leq 1 + \sum_{p=2}^{l} \binom{k}{l} T(k-1,l) \\ &= 1 + (l-1)\binom{k}{l} T(k-1,l) \\ &< 1 + (l-1)k^l T(k-1,l). \end{aligned}$$

If $k = 2l+q$, $T(k,l) < \sum_{i=0}^{q} ((l-1)k^l)^q < ((l-1)k^l)^k$.

Clearly the instructions before the last loop are done in constant time $O(l)$ for each sub-problem. For each recursive call in the last loop, there is also a cost $O(l)$ of generating the set of colours and compute the number of remaining units of the non-exhausted colour. We charge this cost on the called sub-problem (this called sub-problem is always counted in the recursive calls tree because of the worst case assumptions). Hence if $C(k,l)$ denotes the worst case time complexity of the algorithm for the values k and l, then clearly $C(k,l) = O(l) \times T(k,l) = O(l((l-1)k^l)^k)$. (If one wants to take into account the complexity of arithmetic operations, then $C(k,l) = O(\alpha(n) \times l((l-1)k^l)^k)$ where $\alpha(n) = O(log(n))$ is the complexity of additions and subtractions of integers smaller than n.) We obtain a constant with respect to m because we never touch the array containing the intervals; each recursive call only looks at one different cell of the array depending on its recursion depth.

Finally, we obtain the linear complexity because we need to perform a partial sort to obtain the $k - 2l + 2$ largest intervals. Using the algorithm called SELECTSORT in [13], it can be done with optimal time $\Theta(m + (k - 2l + 2) \times \log(k - 2l + 2))$. (SELECTSORT algorithm works as follows. First we find p, the $(k - 2l + 2)$-th largest element, in optimal time $\Theta(m)$ using the algorithm SELECT from [2]. Then we collect and sort all elements larger than p.) ■

We obtain a linear complexity $m + f(k,l)$ where $f(k,l) = a + b$ is the sum of the term corresponding to the complexity of the partial sort and the term corresponding to the complexity of the main part of the algorithm. The first term $a \approx (k - 2l) \times \log(k - 2l)$ is quasi-linear in k and l, and for k fixed is increasing when l is decreasing, i.e. when we come closer to the NP-complete part. The second term $b \approx l((l-1)k^l)^k$ is exponential in k and l but this time it is decreasing when we come closer to the NP-complete part. We wonder whether this phenomenon is due to a bad approximation or if this is really what happens in the worst case. Nevertheless, we do not think that this phenomenon is still happening when we study the average complexity. Indeed, it is most probably true that in average g sees at least $\Omega(l)$ colours (maybe amortised analysis can yield a similar result for the worst case complexity).

Thanks to Theorem 8, we can conclude that we have the following complexity distribution:

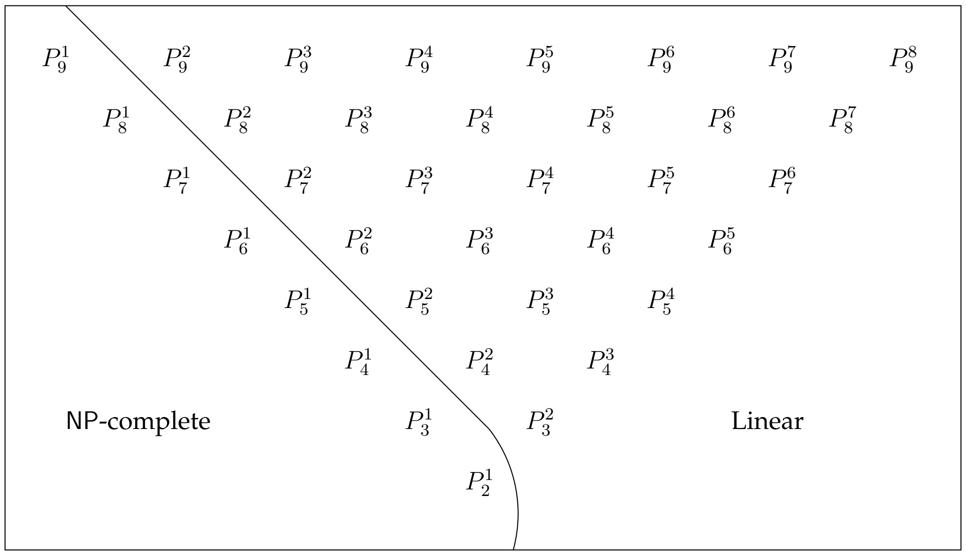

P_9^1 P_9^2 P_9^3 P_9^4 P_9^5 P_9^6 P_9^7 P_9^8
P_8^1 P_8^2 P_8^3 P_8^4 P_8^5 P_8^6 P_8^7
P_7^1 P_7^2 P_7^3 P_7^4 P_7^5 P_7^6
P_6^1 P_6^2 P_6^3 P_6^4 P_6^5
P_5^1 P_5^2 P_5^3 P_5^4
P_4^1 P_4^2 P_4^3
NP-complete
P_3^1 P_3^2
Linear
P_2^1

Figure 3. Complexity on hypergraphs with disjoint hyperedges.

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

Theorem 9 *The problem P_k^l is* NP-*complete if $l = 1$, linear otherwise.*

4.4 Applications of the algorithm

Algorithm 1 can easily be extended to give an admissible colouring when it exists. Moreover it can be used as a sub-procedure to solve a few problems related to the problems P_k^l.

It can, for example, be used to find for a given instance the smallest $l \neq 1$ such that an admissible colouring exists. $\log(k)$ steps of dichotomous search are sufficient for this application. For this search, we use the same instance but different problems P_k^l. This procedure gives us an absolute (with an additive constant equals to 1) approximation algorithm for the problem of minimising l such that the programs can be scattered on at most l processors. This problem is NP-complete because of the NP-completeness of P_k^1 but the algorithm gives the exact value except for the case it stops on $l = 2$; in this case the exact value can be 1 or 2.

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 l processors ($l > 1$). For this application, we suppose we are given the “computing power” $pu_1, \ldots, pu_k$ of the k processors instead of their "computing capacity". The computing power pu_r represents the number of unitary tasks that can be handled by the processor r in one unit of time. Hence values $t_1, \ldots, t_k$ will vary according to the total time given to the processors. We always use the same problem P_k^l but we change the instance by modifying the number of vertices uncovered by hyperedges (by changing n so that $\sum_{r=1}^{k} t_r = n$). We can remark that if $n > kt(I)$ then all the intervals can be covered by one colour. So, we can do a dichotomous search between $t(I)$ and $kt(I)$ and find this estimation in $\log(t(I)) + \log(k-1)$ steps.

We note also applications to the following optimisation problem: minimise $\sigma(A) = \sum_{i \in I} |C(i)|$, the sum of the numbers of colours seen by the intervals, under the constraints $|C(i)| \leq l, \forall i$. Of course, this problem is NP-complete even without any constraints since $\sigma(A) = m$ if and only if A is a positive instance of P_k^1. However a trivial approximation algorithm exists since any continuous colouring is a $+k$-approximation for it (if the continuous colouring starts on the border of some interval, it is a $+(k-1)$-approximation) when there is no constraint. Indeed, any time such a colouring uses one colour more than actually needed to colour some interval then it exhausts one colour in this interval. As we noted before Theorems 6 and 7, there exist optimal solutions for this problem with constraints satisfying the property of these theorems. However, Algorithm 1 makes exhaustive search on solutions satisfying Corollary 2. But we can note that if we break the optimality of the solution in Corollary 2 it is only because there was an optimal solution such that the largest troublemaker was seeing only one colour and we made it see at least two in order to exhaust at least one. Thus whenever we add one to the value of the approximate solution, we exhaust one more colour. Hence the same algorithm can yield a $+(k-1)$-approximation to this problem with constraints.

For all these applications, it would be interesting to study the average complexity because the worst case analysis makes us exhaust only one colour at a time and when l is increasing the average number of exhausted colours should increase also.

5 Conclusion

In this article, we have brought to the fore the existence of a threshold effect for the complexity of neighbour combinatorial problems. This phenomenon formalised by Theorem 9 for hypergraphs with disjoint hyperedges can be seen as a kind of combinatorial analogue to Schaefer's Theorem [15], which separates sub-problems of SAT into NP-complete and linear problems. More surprising is the reversal of complexity between the class of hypergraphs and the considered subclass.

In [12], the author proved that the problem P_3^2 is still linear on interval hypergraphs[3] with maximum degree 2 when the colours are of almost equal

[3] We recall that an interval hypergraph is a hypergraph whose vertices can be ordered as an interval such that its hyperedges are sub-intervals.

size. Current work shows also that P_k^{k-1} for $k \geq 4$ is linear on interval hypergraphs with maximum degree 2 and colours with arbitrary sizes (since hypergraphs with disjoint hyperedges are hypergraphs with maximum degree one, these results generalise some of the results presented in this article). If the NP-completeness results are trivially true with superior maximum degree, it is not the case for the linear complexity results. It may be interesting to see if the border between NP-hard and polynomial cases is still the same with different maximum degree. Another interesting point would be to see if the problems P_k^l are always linear or NP-complete, or if some intermediate complexity shows off. An open question is the complexity of P_k^l on interval hypergraphs. The study of this problem seems interesting because the number of obstructions to be considered (in this paper, we have only the troublemakers and megalomaniacs) is increasing with the maximum degree, and it is not clear if the whole set of obstructions when we consider the whole class of interval hypergraphs can be easily defined. We conjecture that on interval hypergraphs all problems P_k^l are NP-complete.

A natural question asked by Stéphan Thomassé is whether the linear algorithm of Theorem 8 can be generalised to non-uniform constraints, i.e. $|C(i)| \leq l_i$, $2 \leq l_i \leq l$, $\forall i$ (each interval has its own constraint). Unfortunately, Theorem 7 is no longer true in this setting.
(Consider the following instance: $t(i) = 10$, $t(j) = t(h) = 8$, $l_i = 5$, $l_j = l_h = 2$, $t(c_1) = t(c_2) = t(c_3) = t(c_4) = 5$, $t(c_5) = t(c_6) = t(c_7) = 2$. Then there is only one solution (up to a permutation of colours of equal size): $j(c_1) = 5$, $j(c_2) = 3$, $h(c_3) = 5$, $h(c_4) = 3$, $i(c_2) = i(c_4) = i(c_5) = i(c_6) = i(c_7) = 2$. And the largest interval i does not exhaust all the colours it sees except at most one.)
However the author is currently working on some modifications to the same approach that should still yield a polynomial time algorithm.

Stéphan Thomassé also suggested us to interpret problem P_k^l on pairwise disjoint hyperedges as a transportation problem with maximum degree constraints. In this setting, P_k^l is a transportation problem with k sources (suppliers) and m sinks (clients) such that each client must have at most l suppliers (degree at most l). However, this is a transportation problem with uniform transport cost between any couple (client, supplier). An interesting open problem is whether the results of this paper can be combined with optimisation (or approximation) on the transportation cost in the non-uniform setting.

Acknowledgements. The author is grateful to Stéphan Thomassé and Ioan Todinca for useful discussions. We wish to thanks also Ioan Todinca and anonymous referees who suggested many improvements to the writing of the author.

References

1. C. J. Alpert and A. B. Kahng. Recent directions in netlist partitionning: A survey. *VLSI Journal*, 19(1-2):1–81, 1995.
2. Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert Endre Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, 1973.

3. 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.
4. U. V. Çatalyürek. *Hypergraph models for sparse matrix partitioning and reordering*. Computer engineering and information science, Bilkent University, November 1999.
5. 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.
6. 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.
7. Dorit S. Hochbaum and Anu Pathria. The bottleneck graph partition problem. *Networks*, 28(4):221–225, 1996.
8. 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.
9. G. Karypis, E. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.
10. G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. Technical Report 98-036, University of Minnesota, 1998.
11. T. Kloks, J. Kratochvíl, and H. Müller. New branchwidth territories. In *Proceedings 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99)*, volume 1563 of *Lecture Notes in Computer Science*, pages 173–183. Springer-Verlag, 1999.
12. L. Lyaudet. Largeur de branche des graphes de cordes. Master's thesis, ENSLyon, LIP, 2004. http://www.ens-lyon.fr/LIP/Pub/DEA2004.php.
13. Rodrigo Paredes and Gonzalo Navarro. Optimal incremental sorting. In *Eigth Workshop on Algorithm Engineering and Experiments (ALENEX06)*, pages 171–182, Miami, 2006.
14. M. R. Salavatipour. A (1+ϵ)-approximation algorithm for partitioning hypergraphs using a new algorithmic version of the Lovasz local lemma. *Random Structures and Algorithms*, 25(1):68–90, 2004.
15. Thomas J. Schaefer. The complexity of satisfiability problems. In *STOC*, pages 216–226, 1978.
16. S. Shekhar and D. R. Liu. Partitioning similarity graphs: A framework for declustering problems. *Information Systems Journal*, 21(4), 1996.
17. 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.