Initial question (Luc Jaulin): When dealing with outliers in a set estimation context using interval methods, we meet the following problem: Consider p boxes (or interval vectors) of $\mathbb{R}^n$ denoted by $X_1, \ldots, X_p$. Let q be an integer smaller than p. Does it exist a vector $\mathbf{x}$ which belongs to at least q of these p boxes ?

My question is the following : could we find a polynomial algorithm which solves my problem ? (or if you prefer, is my problem NP-hard ?).

---

1) **Marek Gutowski**

I don't know any papers with the description of this problem, but here is the first step I would apply:

1. Partition your set $\mathbb{X} = \{X_1, \ldots, X_p\}$ into subsets called clusters from now on. A cluster is a subset of $\mathbb{X}$ with property: each element has a non-empty intersection with at least one other element of the said cluster OR a single interval vector, if it doesn't intersect any other member of $\mathbb{X}$. To find all clusters you have to perform O(p^2) intersections. Easy, isn't it?

2. Now you have k clusters, $k \leq p$. Your solution(s), if any, must hide in just discovered cluster(s), nowhere else. If $k = p$ then you have exactly k=p different solutions for $q = 1$ and no solutions for any $q > 1$. $k = 1$ (exactly one cluster) means we didn't make any progress, sorry. Otherwise, that is if $k > 1$, you can safely forget all clusters with less than q members. Clusters having exactly q members are easy to examine: the solution is either the intersection of all members of such a cluster or none, if this intersection is an empty set.

Failing all of the above tests may mean that you have to examine all subsets with exactly q members of every cluster large enough. This, indeed, is generally an NP-hard problem. Yet, if k is sharply less than p, then its complexity has just been significantly lowered. At this point you may want to switch to quite another point of view. Consider a graph representing a selected cluster: its vertices are interval boxes and an edge exists between two boxes, if and only if their intersection is non-empty. You may have already build such a graph as a byproduct of the clustering procedure. Any solution(s) must intersect the vertices of order $\geq q - 1$, that is with number of adjacent edges equal at least to $q - 1$ - isn't it? Having the graph in a form of adjacency matrix, all you have to do is computing the sums of each row (or column, if you prefer) and compare those numbers with (q-1). This is a task linear in k, while the adjacency matrix construction requires $O(k^2)$ operations. The overall complexity is therefore $O(k^2)$, even if you include all the subsequent necessary checks of so selected vertices. Please let me know, if you see a hole in my reasoning. Does this solve your problem? With regards, Marek Gutowski

---

2) **Ramon Moore**

Given q boxes, the question whether there is a point $\mathbf{x}$ in at least $p < q$ of them is equivalent to asking whether there are p boxes with non-empty intersection. To find that out requires fewer than q^p computations of intersections, doesn't it? Perhaps I do not understand the question.

---

3) **Luc Jaulin**

Dear Marek, Thank you very much for you detailed and interesting answer. No, I do not see any hole in your reasoning. But it does not really solve my problem, since I still don't know if my problem is NP-hard. I agree that partitioning into cluster may reduce the computation burden. For the complexity, I am searching for the worst case complexity, which amounts to say that $k = 1$. For my application, I usually have $k = 1$. See e.g. Figure 3 of the paper http://www.ensieta.fr/e3i2/Jaulin/qminimax.pdf. The graph algorithm is interesting and I agree that it is polynomial. But, I think that it only computes sufficient conditions for emptyness. If your algorithm fails, then we will know that the q-intersection is empty. But, if it succeed, we cannot conclude that the q-intersection is not empty. Isn't it ? It is a quick answer, but I need to think a little more about it. Maybe I can find a counterexample. Thank you again, Luc

---

4) **Vladick Kreinovich**

Dear Ray, Dear Friends,

From Ray Moore. Given q boxes, the question whether there is a point x in at least $p < q$ of them is equivalent to asking whether there are p boxes with non-empty intersection.

This is absolutely correct.

To find that out requires fewer than q^p computations of intersections, doesn't it ?

Correct. If we test all q^p intersections, this requires time which grows exponentially with p. Luc Jaulin's question is whether it is possible to have an algorithm whoe computation time is bounded by a polynomial of p and q.

---

5) **Alexandre Goldzsteijn**

At least if $p = q$ then the problem is trivial, as the question is equivalent to ask whether the

intersection of all the boxes is empty or not. Now in general:

Claim : If we have p boxes, then their projections on one axe form at most $2p+1$ areas which decide wether we are or not in the projection of each box.

Proof : by induction. For one box, we have 3 areas. If the projection of p boxes forms $2p+1$ areas, then adding on box adds at most 2 areas. Then, we just have to test all possible choice of one area on each axe, hence $(2p+1)^n$ intersections, where n is the dimension of the problem... So, the problem may be exponential with the dimension (though this is not proved) but it is polynomial once the dimension is fixed. We can even be quicker if for each area on each axe we record the number of boxes which have a non empty projection with it. Then you don't even have to check areas for whose number of non empty intersection is less than q.

---

6) **Luc Jaulin**

Dear Ray and all, Thank you very much for your answer. I agree that that the problem can be solved with less than p^q computations of intersections.

But p^q is not polynomial (by polynomial, I mean polynomial in both p and q). When the dimension n is one (i.e., the boxes are intervals of $\mathbb{R}$), one could easily find an algorithm which has a complexity of $O(p.\log(p))$ (it suffices to sort lower and upper bounds all together and to count). But when $n > 1$, I do not know if this polynomial complexity can still be obtained. Regards, Luc.

---

7) **Bruno Lang**

Dear Luc and all. At least for $n = 2$ there is a (low-degree) polynomial algorithm for this problem. In two dimensions this problem is similar to the 3D visibility problem in the following sense: identify each box n° i with a rectangle at depth i and determine which (parts of the) sides of each rectangle are visible from the origin. This problem can be solved in $O((p+k).\log p\ )$ operations, where k $= O(p^2)$ is the total number of intersection points between any sides. (The algorithm is based on Bentley/Ottmann and the so-called connection graph of the scene.). A slight modification of this technique will also yield, for each (part of a) side, the number of rectangles between the line segment and the origin (i.e., the number of rectangles obstructing it). Thus, there is a q-fold intersection iff any of these numbers is $\geq q-1$. In addition, the intersection can be extracted from the connection graph. Best regards. Bruno.

---

8) **Jean Pierre Merlet**.

In the plane (n=2) a sweeping line algorithm will have a complexity of $(p+q).(\log p)$. But complexity may be dangerous because we forget about the constant: in term of computation time the p^q algorithm may be in fact faster than the $(p+q).(\log p)$ one if p, q are low enough and if your p boxes are already sorted for some reasons. JPM

---

9) **Dirk Pattinson**

Hi all, My hunch is that the problem is NP-complete if we assume the planes that bound the boxes are parallel to the coordinate planes. The argument goes as follows:

first note that the following are equivalent:

- (a) there are q boxes with nonempty intersection
- (b) there are q boxes with pairwise nonempty intersection.

Clearly (b) ==> (a) and to get (a) from (b) note that the (additional?) assumption that the boxes are parallel to the coordinate planes lets us argue in each dimension separately, and (b) $\Rightarrow$ (a) in the case where the boxes are intervals (this needs to be checked more carefully than I have done). Now to the problem: Consider the graph whose vertices are the boxes with an edge between box b1 and b2 if b1 and b2 have non-empty intersection. Finding q boxes whose intersection is non-empty amounts to finding a clique of size q in this graph (by the above observation) and we know that this problem is NP-complete. This might be bad news; on the other hand – if the observation above is correct – it means that one can immediately make use of a lot of quite clever graph theoretic algorithms. hope this helps, Dirk Pattinson.

---

10) **Sylvain Pion**

This looks much more like a computational geometry question, so you may be more lucky asking it on the compgeom-discuss mailing list. I believe a closely related work is Frank Nielsen's PhD thesis which you can find here: http://www.inria.fr/rrrt/tu-0418.html . It is in French, but there might be related articles available in English as well. Sylvain Pion

---

11) **Marek Butowski**

Dear Luc, Indeed my answer was pretty quick, but it is correct nevertheless. It doesn't mean, of course, that no better algorithm exists. See my other comment below, within your text.

From Luc: *The graph algorithm is interesting and I agree that it is polynomial. But, I think that it only computes sufficient conditions for emptyness. If your algorithm fails, then we will know that the q-intersection is empty. But, if it succeed, we cannot conclude that the q-intersection is not empty. Isn't it ?*

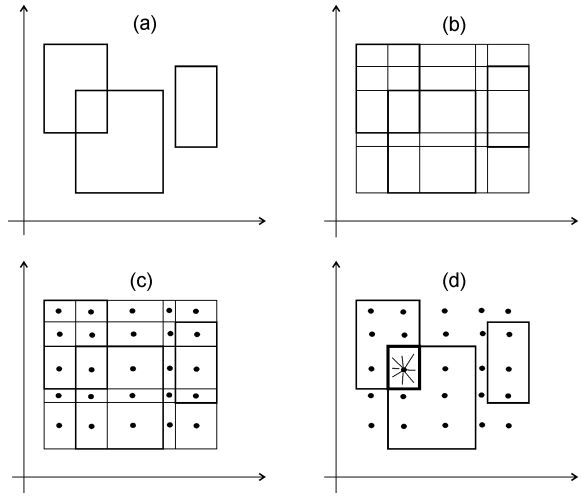
Yes and no. Vertices of order below $q-1$ are not interesting at all, q-intersections are simply impossible there, so obviously empty. Therefore this is a sufficient condition for emptiness of intersection. On the other hand the connectedness of q or more boxes is only a necessary condition for their intersection to be non-empty. Therefore the final, direct check is unavoidable; you have most likely overlooked the last part of my sentence " .. even if you include all the subsequent necessary checks of so selected vertices." Yes, you have to check them one after another as the counterexamples are easy to present: see for example five squares sharing their corners, as in Sierpinski carpet, or even sharing their edges and thus forming of a cross. What *I* have overlooked is the fact that this step increases the overall complexity: $O(k^2) \leq O(qk^2) \leq O(k^3)$, I believe. Anyway, this is the overestimate of the worst case complexity and still better than anticipated NP-hardness. Thank you for the link to your very interesting paper, (not published yet, right?). I already considered similar ideas, in connection with the experimental data analysis, namely with curve fitting. In exchange, and FYI I enclose the relevant paper. I introduce the notion of "crude solutions" there - it is in essence nearly your problem. For clarity, I discuss only a simple linear case i my paper, but the transition to other, nonlinear models is straightforward, if you apply my "box slicing algorithm" also presented in this paper. As for today there is no continuation of this work. And the reason is simple: the experimental data, even in the form of intervals, are never 100% certain. I still don't see how to relate the probabilities (suppose we know the quite precisely) of each measurement to be covered by an interval X_i with the number of measurements, which might be, or perhaps even should be, eventually disregarded by a fitting procedure. In summary: forget the clustering procedure. This was my first idea, not leading directly to the desired result, and in fact - not necessary. Do not waste the computer resources and start directly with the graph picture, this is enough. If finding the "nearest neighbors", i.e. boxes with non-empty intersection with current box, is executed simultaneously with evaluating the intersection itself, then you can expect the worst case complexity $O(p^2)$, at least for vertices of order $k < q$. For higher order vertices - we are at the beginning again, I'm afraid. But aren't you seeking for the highest possible value of q for a given set of boxes? If so, then your problem should be, perhaps, reformulated. Maybe the ideas from my paper are of some help here, if the "faulty" boxes need not neither to be identified nor known at the beginning of the procedure. Regards, Marek Butowski.

---

12) **Luc Jaulin**

Dear all, Thank you for all your answers. After some discussions with A. Goldsztejn, it seems that

for n fixed, the problem can be solved in a polynomial time. The principle of such an algorithm is illustrated by the following figures.



1. Generate the $(2p-1)^n$ as on figure (b).
2. Take the center of all the boxes as on figure (c)
3. Check if there exists one of these center which belong to q of the p boxes (subfigure (d)). The complexity of such a test is $O(p.n)$. The whole algorithm has a complexity of $O(p^{n+1})$.

This is a good new for my application where n is small ($= 4$). But, in a theoretical point of view, it will be nice to know if the problem has a complexity polynomial in p, q and n. Best regards, Luc

---

13) **Dirk Pattinson**

Hi Luc, thanks for the clarification. In hindsight, my argument shows not that your problem is NP-hard, but that it is not harder than np. In order to show NP-hardness, one would of course need to give a translation in the opposite direction, i.e. encode every graph in terms of a system of intersecting boxes, which is what I haven't done. Thanks for pointing out the error, best, Dirk.

---

14) **Gilles Chabert**

Dear all, This problem is already well-known in graph theory and formulated as follows: "The problem of [optimizing the size of] maximum cliques in n-boxicity graphs can be solved in polynomial time". This comes from the fact that the particular class of graphs corresponding to the intersection of boxes

has a number of maximal cliques bounded by a polynomial in p (instead of 2^p, as in the general case). Here is an example of reference I found on the web: http://www.cs.ualberta.ca/~stewart/Pubs/few_cliques.p (See Corollary 4 and the beginning of Section 3) However, since the upper bound on the number of maximum cliques is exponential in n, it should be easy to prove that the problem cannot be polynomial in n. Gilles Chabert.

---

15) **Vladik.**

Dear Gilles, Many thanks for the reference. I do not think, however, that it will be easy to prove that a polynomial algorithm is not possible: if you prove that, then you prove that P is different from NP! At best, we can hope to prove that the general problem is NP-hard. Vladik.

---

16) **Vladik.**

Luc Jaulin's problem – of checking whether at least p out of q given boxes have a non-empty intersection – is NP-hard. Here is a proof.

We can consider boxes in which each side is either [0,0], or [1,1], or [0,1]. We can even limit ourselves to the case when in each box, no more than two variables have [0,0] or [1,1];the rest are [0,1]. If this particular case is NP-complete then the general problem is NP-complete as well. In this particular case, if the intresection is non-empty, then this interesection contains a Boolean vector with all coordinates 0 or 1. If we associate 1 with true and 0 with false, then belonging to each box means that a&b, where a is xi if the box's i-th coordinate is xi=1 and not xi if it is [0,0]. So, we have a sequence of statements a&b, and we need to check whether there exists a boolean vector which satisfies at least p of them. This means that at most q-p of these statements can be false. The falsity of each statement is represented by a 2-clause not a or not b. So, we can reformulate the problem as follows: we have a list of q clauses, and we need to find whether there exists a Boolean vector which makes no more than q-p of them true. This auxiliary problem is known to be NP-complete (see attached). Thus, the original Luc's problem is also NP-complete. Vladik

P.S. I am sending remotely, from an interent cafe near a Polish conference where I am attending an interval session, I already found out that what I send comes out as not an ascii file, I apologize for the inconvenience.

---

17) **Alexandre Goldzsteijn**.

Dear Vladik and Professor Moore, When you count q^p, I think you count the number of ways to

choose q intervals among a set of p intervals, but you allow choosing several times the same interval. If we choose q *different* intervals among a set of p intervals, then we have $C(p,q)$ possible choices. Isn't it? Also, I think that if we use the projections of the intervals on the axis then we can find an algorithm of complexity $(2p+1)^n$. For fixed dimension, we obtain a polynomial complexity w.r.t. p . However, for fixed p and increasing dimension, the $C(p,q)$ algorithm is better ! Best regards! Alexandre

---

18) **Dirk Pattinson**

Hi all,

From Ray Moore : Given p boxes, the question whether there is a point x in at least $q < p$ of them is equivalent to asking whether there are q boxes with non-empty intersection. To find that out requires fewer than p^q computations of intersections, doesn't it? Perhaps I do not understand the question.

My hunch is that the problem is NP-complete if we assume the planes that bound the boxes are parallel to the coordinate planes. The argument goes as follows: first note that the following are equivalent:

- (a) there are q boxes with nonempty intersection
- (b) there are q boxes with pairwise nonempty intersection.

Clearly (b) $\Rightarrow$ (a) and to get (a) from (b) note that the (additional?) assumption that the boxes are parallel to the coordinate planes lets us argue in each dimension separately, and (b) $\Rightarrow$ (a) in the case where the boxes are intervals (this needs to be checked more carefully than I have done). Now to the problem: Consider the graph whose vertices are the boxes with an edge between box b_1 and b_2 if b_1 and b_2 have non-empty intersection. Finding q boxes whose intersection is non-empty amounts to finding a clique of size q in this graph (by the above observation) and we know that this problem is NP-complete. This might be bad news; on the other hand – if the observation above is correct – it means that one can immediately make use of a lot of quite clever graph theoretic algorithms. Hope this helps, Dirk.

---

19) **Wayne**

From Dirk Pattinson: Consider the graph whose vertices are the boxes with an edge between box b_1 and b_2 if b_1 and b_2 have non-empty intersection. Finding q boxes whose intersection is non-empty

amounts to finding a clique of size q in this graph (by the above observation) and we know that this problem is NP-complete.

Except I think you've translated in the wrong direction. Remember, to prove a problem is NP-complete, you need to translate an *existing* NP-complete problem *to* your problem. The translation you've described above is in the opposite direction: you've shown that we can use clique-finding to find box intersections. But there's no guarantee that clique-finding is the most efficient way to solve box intersection. A better algorithm might exist. The direction you've translated above would be akin to saying that sorting takes exponential time because we can solve the sort problem by trying every possible permutation until we find one that's sorted. Obviously, such an algorithm would work but doesn't prove that sorting in general takes exponential time. :-) To prove that box intersection is NP-complete, you'd need to show how to *use* box intersection to *solve* clique-finding. As a first step, this requires demonstrating that an arbitrary graph can be "coded" into box intersections. I'm not sure that's possible; I haven't thought about it, but it seems to me that it might be possible to create a graph that cannot possibly represent box intersections in the model you've described above. The way I remember which direction to translate is to think in terms of an actual program. If you give me a program to solve your problem, then my task (to prove your problem is NP-complete) is to *use* your program to solve an NP-complete problem in polynomial time. To do that, I need to figure out some way to translate an existing NP-complete problem to your problem, in polynomial time. Such a translation proves that your problem is NP-complete, because it demonstrates that your problem is at least as hard as existing NP-complete problems, in the sense that if somebody finds a polytime algorithm to solve your problem, then I can use your program and my translation to solve an existing NP-complete problem in polynomial

time. Wayne.

---

20) **Wayne Hayes**

I just realized that there were a few typos in my previous message that may have made it unclear. Here's another attempt to explain *why* the translation must be from an existing NP-complete problem, to the new problem. Let's say you give me a "black box" that solves the box intersection problem. Then, let's say that I find some way to encode the input and output of the clique-finding problem into the input and output format for your black box, such that the clique problem has a positive answer if and only if your black box returns a valid solution to the box intersection problem (which, recall, is an encoding of the clique-finding problem). Furthermore, the translation of both inputs and outputs takes polynomial time. Then, *if* your "black box" always runs in polytime, then I've magically found a way to solve clique-finding problems in polytime. Hope that's clear. As I said in my previous message, I would guess that it would be difficult or impossible to encode

arbitrary graphs into intersecting boxes (or at least, to do so in polynomial time). This doesn't mean that the box intersection problem isn't NP-complete; it just means that, if it's NP-complete, then clique-finding isn't the right problem to translate from. Wayne Hayes.

---

21) **Marek Gutowski**

Vladik, I'm impressed. This is to say I do not understand the very first sentence of your proof. Anyway, you don't mention what is the dimension of the boxes. So, how should I understand your proof if the 1D case is evidently NOT NP-hard? Something has to be wrong. Marek Gutowski.

---

22) **Baker**

Vladik, This might be minor. However, at the beginning, you assert that the problem is "np-hard," whereas at the end of your proffered proof, you say, "therefore, the original problem is np-complete." Can you clarify the distinction ("np-hard" versus "np-complete")? Baker

---

23) **Wayne Hayes**

Marek, Somehow I didn't see Vladik's original message, but the phrase "We can consider boxes in which each side is either $[0,0]$, or $[1,1]$, or $[0,1]$" refers to the "sides" of the box either being points, or intervals of length one. For example, in three dimensions, the "box" (ie., line) going along the x-axis from the origin to $(x,y,z) = (1,0,0)$ has "sides" $x \in [0,1], y \in [0,0], z \in [0,0]$. The line starting at the end of that previous line and going up paralell to the y-axis has "sides" $x \in [1,1], y \in [0,1], z \in [0,0]$. The line starting from that point and going up parallel to the z-axis has "sides" $x \in [1,1], y \in [1,1], z \in [0,1]$. Similarly, the unit box in the (x,y) plane has "sides" $x \in [0,1], y \in [0,1], z \in [0,0]$. Vladik's example can be phrased in higher dimension, but has similar cases.

---

24) **Wayne Hayes**

Vladik, I'm sorry, but it seems to me that your transformation is in the wrong direction. You've shown that the intersection problem reduces to the boolean problem, which is the wrong direction of transformation for a proof of NP-completeness (see my message from last night for details). Instead, we need to show that an existing NP-hard problem reduces to our problem, in order to demonstrate that our problem is also NP-hard. Your proof also doesn't mention p, q, or n (the dimension of the boxes), as Marek points out.

25) **Guillaume Melquiond**

Wayne Hayes wrote: I'm sorry, but it seems to me that your transformation is in the wrong direction. You've shown that the intersection problem reduces to the boolean problem, which is the wrong direction of transformation for a proof of NP-completeness (see my message from last night for details). Instead, we need to show that an existing NP-hard problem reduces to our problem, in order to demonstrate that our problem is also NP-hard.

I had the same reaction when reading Vladik's mail the first time. But as far as I can tell, Vladik's transformation is sound. Consider a formula in conjunctive normal form (n variables, q clauses). You want to find a boolean vector such that at most $q-p$ clauses of the formula hold. Each clause is a disjunction, so its negation is a conjunction. Associate to this conjunction a box. If the variable i does not appear in the conjunction, the projection of the box on axis i is $[0,1]$. If the variable i appears negated, the projection is $[0,0]$. Otherwise the projection is $[1,1]$. So any vector with coordinates 0 or 1 in this box is a boolean vector which satisfies the conjunction. Solve the intersection problem for p boxes. This gives you (or not) a vector. Each coordinate that is neither 0 or 1 must be changed to either one. Due to the way the boxes were chosen, this modified vector also happens to be in the intersection. As all its coordinates are 0 or 1, this is a boolean vector, and it satisfies at least p conjunctions of the negated formula. So it means that it satisfies at most $q-p$ disjunctions (the ones associated to the other clauses) of the original formula. So you have solved the NP-complete boolean problem by using a solver for the intersection problem. As the transformations are linear in $n.q$, if the intersection solver is polynomial, then one can solve the boolean problem in polynomial time. So the intersection problem is NP-hard. And since the intersection problem is NP (checking that an intersection of p boxes is not empty is linear in np), it is NP-complete. Best regards, Guillaume Melquiond.

---

26) **Luc Longpre**

Here is Vladik's proof expressed a little more formally. The box intersection problem: Given a set of p n-dimensional boxes, and a number $q < p$. Is there a vector x which belongs to at least q of these p boxes? A previous e-mail described an algorithm that works in time p^{n+1}. This algorithm is polynomial time if n is constant, it's exponential if n is allowed to grow with input size. Vladik provided a proof of NP-completeness for the case of n not constant. The problem box intersection problem obviously in NP: it is easy to check that a given vector belongs to at least q of the boxes. So we only need a proof of NP-hardness. The proof is by reduction from the "limited clauses" problem described in the reference, which has been proved NP-complete: Given a 2-CNF formula F and a

number k, is there a Boolean vector which satisfies at most k clauses of F. Here is Vladik's proof, expressed more formally. Given a p-clauses 2-CNF formula F of n variables and a number k, we associate a dimension with each variable. Build a set of p n-dimensional boxes, one for each clause. If clause i mentions v_{i_1} and v_{i_2} variables, then the ith box has sides $[0,1]$ in all dimensions except in the dimensions associated with variables v_{i_1} and v_{i_2}. For those two dimensions, the side is $[0,0]$ if the variable occurs positively in the clause and $[1,1]$ is the variable occurs negatively. (Actually the boxes are $n-2$ dimensional objects in n-dimensional space, since two of the dimensions are degenerate. For example, the box $([0,0],[0,1],[0,1],[1,1])$ is a 2-D object in 4-D space.) Now, let $q = p - k$. The claim is that there exists a vector x which belongs to at least q of these p boxes if and only if there is a Boolean vector which satisfies at most k clauses of F. Suppose the vector x exists. Suppose box i is associated with variables v_{i_1} and v_{i_2}. For each of the box i for which x belongs, make v_{i_1} False if the box has $[0,0]$ on the side associated with variable v_{i_1}. Same for variable v_{i_2}. Because of the way the boxes were build, the boolean vector we build will make the clause associated with the box false. For example, if the clause is $v_{i_1} \mid\mid v_{i_2}$, then the box will have $[0,0]$ for the sides associated with both variable, so they will be both assigned the False boolean value, making the clause false. This means that the Boolean formula built will make at least q clauses become false. This formula will satisfy at most $k = p - q$ clauses. In the opposite direction, if there is a Boolean vector which satisfies at most k clauses of F, build a vector x which has value 0 in dimension i if the variable associated with this dimension the Boolean vector is False, and 1 otherwise. Since the Boolean vector satisfies at most k clauses of F, it makes at least $q = p - k$ clauses false. This means that the vector we build will belong to all the boxes associated with the q clauses that are false. Luc Longpre.
