HAL
archives-ouvertes.fr

Q-intersection Algorithms for Constraint-Based Robust Parameter Estimation

Clément Carbonnel, Gilles Trombettoni, Philippe Vismara, Gilles Chabert

To cite this version:

Clément Carbonnel, Gilles Trombettoni, Philippe Vismara, Gilles Chabert. Q-intersection Algorithms for Constraint-Based Robust Parameter Estimation. AAAI Conference on Artificial Intelligence, Jul 2014, Québec City, Canada. pp.2630-2636. hal-01084606

HAL Id: hal-01084606

https://hal.archives-ouvertes.fr/hal-01084606

Submitted on 21 Nov 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Q-intersection Algorithms for Constraint-Based Robust Parameter Estimation

Clement Carbonnel
LAAS-CNRS
Université Toulouse, France
carbonnel@laas.fr

Gilles Trombettoni
LIRMM
Université Montpellier, France
gilles.trombettoni@lirmm.fr

Philippe Vismara
LIRMM
SupAgro Montpellier, France
vismara@lirmm.fr

Gilles Chabert
LINA
EMN Nantes, France
gilles.chabert@mines-nantes.fr

Abstract

Given a set of axis-parallel n-dimensional boxes, the q-intersection is defined as the smallest box encompassing all the points that belong to at least q boxes. Computing the q-intersection is a combinatorial problem that allows us to handle robust parameter estimation with a numerical constraint programming approach. The q-intersection can be viewed as a filtering operator for soft constraints that model measurements subject to outliers. This paper highlights the equivalence of this operator with the search of q-cliques in a graph whose boxicity is bounded by the number of variables in the constraint network. We present a computational study of the q-intersection. We also propose a fast heuristic and a sophisticated exact q-intersection algorithm. First experiments show that our exact algorithm outperforms the existing one while our heuristic performs an efficient filtering on hard problems.

1 Introduction

The combinatorial *q-intersection* operator can handle a specific class of numerical (real-valued) Constraint Satisfaction Problems. In numerical CSPs, the domain of the n variables is an n-dimensional box (a Cartesian product of n intervals).

The q-intersection appears in the context of *parameter estimation*, a fundamental task in control theory, robotics and autonomous systems, where the state (position, velocity, etc) has to be estimated from noisy sensor data. The purpose of parameter estimation is to estimate the unknown "internal" parameters of a system from a set of measurements. A system can be thought here as any physical process that transforms inputs into outputs. The estimation is made from the measurements of outputs obtained with different inputs.

Using a constraint programming approach, each measurement provides a constraint system $y = f(x, p)$ where x is the input vector, y is the output vector, f is a vector of real-valued functions and p is a vector of unknown parameters. As an illustrative example, consider a simple spring for which the constant k is the parameter to be determined. It is well-known that a mass m attached to the spring has a simple harmonic motion whose angular frequency is $\omega := \sqrt{m/k}$. On this example, the input is m, the output is ω, the parameter is k and we have $f(x, p) = \sqrt{x/p}$. We perform pairs of measurements (x_i, y_i), each measurement being associated with some uncertainty. For instance, attaching to the spring different loads of given masses m_i, one obtains a set of measurements ω_i of the frequencies.

In many applications uncertainty is modeled by a Gaussian noise.[1] In the interval constraint programming approach, uncertainty is modeled by a bounded error ϵ (Walter and Piet-Lahanier 1993). In this context, one can enforce consistency algorithms on the ith measurement constraint:

$$y_i - \epsilon_i \leq f(x_i, p) \leq y_i + \epsilon_i$$

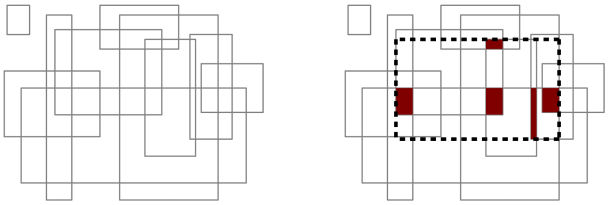
to get a filtered/contracted box $[p]_i$ that rigorously encloses the true value of the parameters. The overall approximation is then obtained by a simple intersection: $p \in \bigcap_i [p]_i$. This filtering approach is nice because it gives a safe enclosure of the parameters, regardless of the model. However, it is not robust to *outliers*, i.e., points that correspond to experimental errors and lead to an empty intersection. Another idea is to assume that at least q measurements are valid, the other ones being potential outliers. The q-intersection studied in this paper replaces the previous intersection by a union of all the intersections obtained with q measurements. More formally, given a set S of boxes and an integer q, we say that $p \in \mathbb{R}^n$ is a (S, q)-intersection point if p belongs to at least q boxes in S. The q-intersection of S is thus defined as follows (where $\mathbf{I}\mathbb{R}$ is the set of all intervals over $\mathbb{R}$).

Definition 1. Let S be a set of boxes of $\mathbf{I}\mathbb{R}^n$. The q-intersection of S, denoted by $\cap^q S$, is the box of smallest perimeter that encloses the set of (S, q)-intersection points.

For instance, the box in dotted lines in Fig. 1–b is the 4-intersection of the 10 two-dimensional boxes (in plain lines). The q-intersection allows a (soft) constraint programming approach to parameter estimation. It handles soft numerical constraint networks where at least q numerical constraints must be verified. Each subsystem $y_i - \epsilon_i \leq f(x_i, p) \leq y_i + \epsilon_i$ corresponding to one measurement contracts the box (filters the domain) using consistency methods, e.g., numeric constraint propagation (Lhomme 1993;

[1] When the noise is described by a probability distribution, the parameter estimation problem is tackled by statistical methods (see e.g., (Beck and Arnold 1977)). If the function f is linear, the *least squares* method gives the maximum-likelihood estimator (by the Gauss-Markov theorem). Otherwise, no clear property holds on the result, which motivates the constraint programming approach.

Copyright © 2014, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.



(a) A set S of 2-dimensional boxes

(b) The dashed box is $\cap^4 S$. Zones that belong to at least 4 boxes are darkened.

Figure 1: Illustration of q-intersection for q=4, n=2.

Benhamou et al. 1999). Then, an operator computes the q-intersection of all the filtered boxes. For some applications, these two steps are called at each node of a search tree.

The problem of parameter estimation with outliers in a bounded-error context has been introduced first in (Jaulin, Walter, and Didrit 1996), where the q-intersection has also been formalized. Our paper presents the first theoretical study of q-intersection, along with efficient incomplete and exact algorithms to handle this problem.

Intervals, boxes and orderings A closed interval is a set of the form $[a, b] = \{x \in \mathbb{R} : a \leq x \leq b\}$. For an interval $I = [a, b]$, we use the notation $\underline{I} = a$ and $\overline{I} = b$. An n-dimensional box is a Cartesian product of n intervals, i.e., an element of $\mathbf{I}\mathbb{R}^n$. The ith coordinate of a box B (an interval) is denoted by $B[i]$, and its *perimeter* is defined as $|B| = \sum_{i=1}^{n} \overline{B[i]} - \underline{B[i]}$. We call *direction* a couple (i, o), where i is a dimension and $o \in \{-, +\}$ is the orientation. For a box B, we denote $\mathcal{L}^d(B)$ the left bound of B w.r.t. d, where $\mathcal{L}^{d=(i,+)}(B) = \underline{B[i]}$ and $\mathcal{L}^{d=(i,-)}(B) = \overline{B[i]}$. Symmetrically, we define $\mathcal{R}^d(B)$, the right bound of B w.r.t. d, where $\mathcal{R}^{d=(i,+)}(B) = \overline{B[i]}$ and $\mathcal{R}^{d=(i,-)}(B) = \underline{B[i]}$. We define the order $\geq_o$ on $\mathbb{R}$ as $\geq$ if $o = +$, and $\leq$ if $o = -$. Given a direction $d = (i, o)$ and a set of boxes $S = \{B_i\}_{i \in \mathbb{N}}$, we denote by $>_d$ the strict order on S such that $B_i >_d B_j$ if and only if $(\mathcal{L}^d(B_i) >_o \mathcal{L}^d(B_j))$ or $(\mathcal{L}^d(B_i) = \mathcal{L}^d(B_j)) \wedge (i > j)$.

The q-intersection operator A straightforward exact algorithm for computing the q-intersection is the *grid algorithm* introduced in (Jaulin and Bazeille 2009). The core idea is described in Figure 2: using the projections of the boxes onto every axis, one can generate a grid of $(2p - 1)^n$ cells, where p is the number of boxes and n is the number of dimensions. Note that for every $x \in \mathbb{R}^n$ belonging to q boxes, there exists a cell C containing x that is included in at least q boxes. Therefore, the algorithm iterates through the cells, and computes the minimum-perimeter bounding box of all the cells that are included in at least q boxes. The overall complexity is $O(np(2p - 1)^n)$. This algorithm is polynomial if n is fixed, but not in the general case.

In practice the grid algorithm is rather slow, even for small values of n. Moreover, the q-intersection operator is typically used to filter the search space at each node of a search tree. Hence, if exact q-intersection algorithms are too slow for this task, computing a reasonable enclosure of $\cap^q S$ may be satisfactory (provided it can be done in polynomial time). This can be achieved using a method first introduced in (Chew and Marzullo 1991), called here *projective filtering*, that solves the problem on each dimension independently. Formally, for each dimension i, the algorithm computes the projection $S[i]$ of the boxes and solves the q-intersection problem on $S[i]$. Since $S[i]$ is a set of intervals each $\cap^q S[i]$ can be computed in polynomial time (a specialized version of the grid algorithm is $O(p \log(p))$). It is clear that $(\cap^q S)[i] \subseteq (\cap^q S[i])$, so $\prod_{i \in (1..n)} (\cap^q S[i])$ is a valid enclosure of $\cap^q S$. The overall complexity is $O(np \log(p))$. Figure 3 illustrates the algorithm. This method has no guaranteed approximation ratio but can be fairly efficient on problems with few parameters (Jaulin and Bazeille 2009).

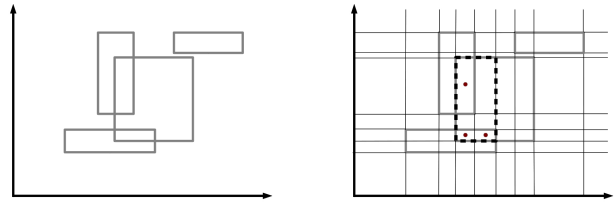


Figure 2: Principle of the grid algorithm for q=2, n=2. Cells contained in q boxes are dotted. The dashed box is $\cap^2 S$.

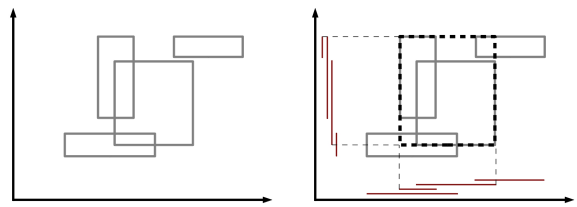


Figure 3: Principle of *projective filtering* for q=2, n=2. The method outputs the dashed box, a gross overestimate of $\cap^2 S$.

2 Theoretical analysis

In this section we give a brief overview of the complexity of problems closely related to the q-intersection operator.

Given a set S of boxes, the *intersection graph* of S is the graph whose vertices are in one-to-one correspondence with the boxes in S and in which two vertices are adjacent if and only if the corresponding boxes have a non-empty intersection. In graph theory, the boxicity of a graph G is the least integer k such that G is the intersection graph of some set of k-dimensional boxes. For instance, interval graphs are exactly the graphs of boxicity one. It has been observed in (Roberts 1969) that the boxicity is properly defined for every graph $G = (V, E)$, and satisfies $box(G) \leq |V|$. The next lemma follows from the fact that Roberts' proof is constructive.

Lemma 1. *There is a polynomial-time algorithm that takes as input a graph G and constructs a set S of boxes whose intersection graph is G.*

Given a set S of boxes of $\mathbb{R}^n$ and an integer q, we call QPOINT the problem of deciding if an (S, q)-intersection point exists. The following proposition will be instrumental.

Proposition 1. QPOINT *is NP-complete.*

Proof. A vector $x \in \mathbb{R}^n$ contained in q boxes is a witness, so the problem is in NP. For hardness, we reduce from k−CLIQUE. For every instance $\langle G = (V, E), k\rangle$ of k−CLIQUE, we can obtain a box-representation S of G in polynomial time using Lemma 1. If there is a (S,q)-intersection point, then G has a k-clique by definition of the intersection graph. Conversely, if G has a k-clique, there exists a subset $S' \subseteq S$ of size k whose members have a pairwise non-empty intersection. Since axis-parallel boxes are 2-Helly (Graham, Grötschel, and Lovász 1995), there exists an (S', q)-intersection point, which concludes the proof. □

Recall that the initial goal behind the use of the q-intersection is to reduce search space. Therefore, a natural conversion of the q-intersection operator into a decision problem takes as input a set of boxes S and asks whether $|\cap^q S| \leq k$ for some fixed rational number (or integer) k. We call this problem k−QINTER, and QINTER−OPT the problem of determining the minimum k such that $|\cap^q S| \leq k$.

Theorem 1. k−QINTER *is coNP-complete for every k and* QINTER−OPT *is not polynomially approximable with a constant factor unless P = NP.*

Proof. For simplicity, we denote by k−coQINTER the complementary problem of k−QINTER. Since $Q = \{x \in \mathbb{R}^n : x \text{ is a } (S, q)\text{-intersection point}\}$ is a finite union of intersections of boxes, which are closed sets, Q is a closed set and contains $2n$ extremal points. If $|\cap^q S| > k$, then these extremal points provide a witness so k−coQINTER is in NP. For hardness, we reduce from QPOINT. Let $I = (S, q)$ be an instance of QPOINT. We pick $x \in \mathbb{R}^n$ such that for every point y that belongs to a box in S, $|x - y| \geq k + 1$. Then, we add a set S' of q identical boxes of perimeter k centered on x. By construction, the boxes from S' have an empty intersection with the boxes from S. We denote by $I' = (S \cup S', q)$ the instance of k−coQINTER obtained. If there is a (S, q)-intersection point, then $\cap^q(S \cup S')$ must contain both x and a point contained in at least one box from S. By the choice of x, this implies $|\cap^q (S \cup S')| > k$ and I' is a yes-instance. Conversely, if $|\cap^q (S \cup S')| > k$ then $\cap^q(S \cup S') \neq \cap^q S'$ because by construction $|\cap^q S'| = k$. Thus, there must exist an (S, q)-intersection point, and I is a yes-instance.

The second claim follows from the fact that the point x can be chosen to be arbitrarily distant from the boxes of S. Hence, if we force x to be at a distance larger than $\epsilon k + 1$ from any box of S in every dimension, any polynomial ϵ-approximation algorithm would decide of the satisfiability of I, which is impossible unless P = NP. □

Unless P = NP, these results rule out the prospect of polynomial time algorithms for the q-intersection operator, even if we are only looking for a constant-factor approximation. However, computing $\cap^q S$ can be harder than computing $|\cap^q S|$, so the analysis is not fully satisfactory. We now show that the following problem already exhibits high complexity. QINTER−CHECK: Given a set S of boxes, an integer q and a box B, is the q-intersection of S equal to B ?

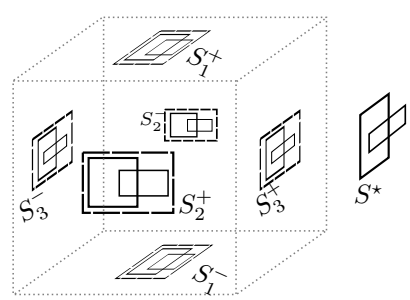
The class DP, introduced in (Papadimitriou and Yannakakis 1982), is the set of all decision problems that can be rephrased as the intersection of an NP problem and a coNP problem. Notorious DP-complete problems include CRITICAL−SAT, which asks whether a given CNF is unsatisfiable but removing any clause makes it satisfiable, or EXACT−CLIQUE, which is the problem of deciding whether the clique number of a graph is exactly k. Note that DP-complete problems are both NP-hard and coNP-hard.

Theorem 2. QINTER−CHECK *is DP-complete.*

Proof. We reduce from EXACT−CLIQUE. Consider an instance $\langle G = (V, E), k\rangle$ of EXACT−CLIQUE: *Does the largest clique in G has size exactly k?* We have to compute, in polynomial time, a set of boxes S^f and a box β such that $\cap^q S^f = \beta$ if and only if k is the clique number of G.

Using Lemma 1, we can build a set S of n-dimensional boxes ($n = |V|$) such that there exists an (S, k)-intersection point if and only if G has a clique of size k. Let B_h be the minimum-perimeter bounding box of S. We denote by G' the graph obtained by adding a universal vertex to G and we define $S' = S \cup \{B_h\}$. Since B_h intersects with all boxes in S, there exists an $(S', k+1)$-intersection point if and only if G' has a clique of size $k + 1$. For simplicity, we will assume that every point in a box from S' has coordinate in $]-1, +1[$ for every dimension (if this is not the case, we can enforce it by scaling down S' and translating it closer to the origin).

We build an $(n + 1)$-dimensional instance of QINTER−CHECK as illustrated in the following figure ($n = 2$):



For every dimension i, we define the set of boxes S_i^+ that includes a box B_i^+ for every $B' \in S'$, such that $B_i^+[j] = B'[j]$ for $j < i$, $B_i^+[i] = [1, 1]$, and $B_i^+[j] = B'[j - 1]$ for $j > i$. Intuitively, S_i^+ is a copy of S' extended to be $(n+1)$-dimensional, in which all the boxes have width $[1, 1]$ in the extra dimension i. Symmetrically, we define a set S_i^- for every dimension i, where $B_i^-[i] = [-1, -1]$ for any box in S_i^-. Finally, we define $S^\star$ as an $(n + 1)$-dimensional extension of S setting $B[n + 1] = [2, 2]$ for the missing dimension.

Let $S^f = \cup_{i=1..(n+1)} \left(S_i^+ \cup S_i^-\right) \cup S^\star$ and $\beta = [-1, 1]^{n+1}$. We will prove that $\cap^{k+1} S^f = \beta$ if and only

if the clique number of G is k. Suppose the clique number of G is k. Then, every S_i^+ (resp. S_i^-) contains an $(S^f, k+1)$-intersection point, but $S^\star$ does not (because G' has a $(k+1)$-clique, but not G). Therefore, for each dimension i, there is a pair of $(S^f, k+1)$-intersection points x_i^+, x_i^- such that $x_i^+[i] = 1$ and $x_i^-[i] = -1$. Since every box of $S^f \backslash S^\star$ has its coordinates in $[-1, 1]$, we have $\cap^{k+1} S^f = \cap^{k+1}(S^f \backslash S^\star) = [-1, 1]^{n+1}$. The converse implication is similar : if $\cap^{k+1} S^f = [-1, 1]^{n+1}$, then there is no $(S^\star, k+1)$-intersection point, thus G has no $(k+1)$-clique. Furthermore, for every i there exists an $(S_i^+, k+1)$-intersection point, and by construction an $(S', k+1)$-intersection point, which means that G has a k-clique.

The above proof only shows hardness. For membership in DP, an instance (S, q, B) of `QINTER-CHECK` is a yes-instance if and only if (a) there is an (S, q)-intersection point on each face of B (clearly in NP) and (b) there is no (S, q)-intersection point outside B (which is a coNP problem). □

3 (q-1)-core filtering and greedy coloring

We first propose a new polynomial-time heuristic algorithm for the q-intersection problem, based on the intersection graph G of the box set S. Even if the intersection graph alone does not capture the whole problem geometry, it is a convenient structure supporting many efficient operations.

Observe that the boxes which are not part of a q-clique in G can be safely removed since they cannot contain an (S, q)-intersection point. While deciding whether a given vertex is part of a q-clique is NP-hard, weaker elimination rules can be applied in polynomial time. A simple necessary condition for a vertex to be in a q-clique is that it must belong to the $(q-1)$-core of G, defined as the largest induced subgraph of G whose vertices have degree at least $q-1$. If G has p vertices and m edges, the $(q-1)$-core of G can be computed in time $O(m+p)$ (Batagelj and Zaversnik 2011).

We can improve this filtering with a greedy coloring run in every direction. For any coloring, each clique of the intersection graph must contain one vertex of color at least q. Therefore, if for a given direction d (for instance going "from left to right") we perform a greedy coloring of the boxes using the order $<_d$, the left bound of the first box with color q will be smaller than the left bound of $\cap^q S$. Our algorithm `coreF` (Algorithm 1) repeats that procedure on all $2n$ directions to obtain a valid enclosure of $\cap^q S$.

The overall complexity of the procedure `coreF` is $O(np^2 + m + p + np\log(p) + nm) = O(np^2)$, to be compared to the complexity $O(np\log(p))$ of projective filtering.

Proposition 2. `coreF` *is strictly stronger than projective filtering.*

Proof. We first prove that `coreF` is at least as tight as projective filtering. Let i be a dimension. We will prove that $\underline{\cap^q S[i]} \leq L[i]$ (the proof $\overline{\cap^q S[i]} \geq R[i]$ is symmetric). By construction of $L[i]$, we know that the box B colored with q (where $L[i] = \mathcal{L}^d(B)$) is adjacent to $q-1$ boxes in G (these boxes are colored with $1..(q-1)$). So B intersects with $q-1$ boxes whose left bound is smaller or equal to $L[i]$. Hence $L[i]$ is a $(S[i], q)$-intersection point.

Algorithm 1: coreF(S, q)

Data: A set S of p n-dimensional boxes, an integer q

$G \leftarrow$ The intersection graph of S
$G \leftarrow$ The (q-1)-core of G
Remove from S all the boxes that are no longer in G
for $i = 1..n$ **do**
  Perform a greedy coloring with respect to $<_{d=(i,+)}$
  $L[i] \leftarrow$ The left bound of the first box with color q
  Perform a greedy coloring with respect to $<_{d=(i,-)}$
  $R[i] \leftarrow$ The right bound of the first box with color q
return $[L[1], R[1]] \times ... \times [L[n], R[n]]$

A simple example on which `coreF` computes a strictly tighter box than projective filtering comes with the 3 following rectangles: $B_1 = [0, 3] \times [0, 1]$, $B_2 = [0, 1] \times [0, 3]$ and $B_3 = [2, 3] \times [2, 3]$. For $q = 2$, `coreF` returns $[0, 1] \times [0, 1]$ while projective filtering returns $[0, 3] \times [0, 3]$. □

4 Exact algorithm

In this section we present a new exact algorithm for the q-intersection operator. Like `coreF`, the algorithm operates mainly on the intersection graph but makes an efficient use of the geometric information available through the boxes.

By the Helly property, the q-cliques of the intersection graph G are in one-to-one correspondance with the sets of q boxes that share a common point. Therefore, the q-intersection can be computed by enumerating the q-cliques of G: starting from an empty box H, for each q-clique found with associated boxes $B_1, ..., B_q$ we expand H to include $B_1 \cap ... \cap B_q$. Once every q-clique has been processed, H contains every (S, q)-intersection point and has minimum perimeter, so $H = \cap^q S$. This enumeration procedure can then be improved by a Branch-and-*Bound*. Consider the search tree whose depth-k nodes are exactly the cliques of size k, and the children of a node labeled with a clique C are labeled with cliques of the form $\{C \cup v\}$. At each node, if the intersection U of the boxes associated with the current clique is such that $U \subseteq H$, we know that no q-clique originating from this branch can improve H and we can backtrack.

To be efficient in practice, this B&B must be able to generate good bounds quickly. It appears that if the goal is only to compute $\mathcal{L}^d(\cap^q S)$ for some direction d (which corresponds to a single face of $\cap^q S$), simple branching rules can guarantee that the first q-clique found determines $\mathcal{L}^d(\cap^q S)$.

Therefore, despite the potential redundancy, the best q-intersection algorithm we have devised runs $2n$ times the "directional" B&B, i.e., once for each direction/face.

Strategy for one face Let us fix a direction $d = (i, o)$. Let $\mathcal{Q}$ be the set of all possible nonempty intersections of q boxes from S. The goal is to compute $\mathcal{L}^d(\cap^q S)$, defined by:

$$\mathcal{L}^d(\cap^q S) = \min_{Q \in \mathcal{Q}}^{\geq_o} \mathcal{L}^d(Q) \tag{1}$$

Furthermore, it is easy to prove that every $Q \in \mathcal{Q}$ satisfies:

$$\mathcal{L}^d(Q) = \max_{B\ s.t. Q \subseteq B}^{\geq_o} \mathcal{L}^d(B) \tag{2}$$

Geometrically, it means that the left bound of Q is the left bound of the rightmost box used to compute the intersection Q. It follows that the optimal intersection $Q_{\min}$ for a direction d is such that the left bound of its rightmost box is minimum. More formally:

$$\mathcal{L}^d(\cap^q S) = \min_{Q \in \mathcal{Q}}^{\geq_o} \left(\max_{B\ s.t. Q \subseteq B}^{\geq_o} \mathcal{L}^d(B) \right) \tag{3}$$

Thus, if we consider one box B at a time in increasing order w.r.t. $>_d$ and ask for each one: "Is B the rightmost box of some intersection Q of q boxes?", then the first positive answer will provide $\mathcal{L}^d(\cap^q S)$.

The subproblem solved for each box B is equivalent to decide whether there exists an $(S', q-1)$-intersection point, where $S' = \{B' \in S | (B' <_d B) \wedge (B' \cap B \neq \emptyset)\}$, which is a standard $(q-1)$-clique problem on the intersection graph of S'. On a side note, since graphs of boxicity n have at most $O((2p)^n)$ maximal cliques (Chandran, Francis, and Sivadasan 2010), the $(q-1)$-clique subproblems are solvable in polynomial time if n is fixed.

Bound propagation and nogoods Each run of the one-direction algorithm provides valuable information that can be reused for the next calls to avoid redundant computations.

First, if we have an intersection $Q \in \mathcal{Q}$ such that $\mathcal{L}^d(\cap^q S) = \mathcal{L}^d(Q)$ (obtained after the treatment of one face), it is clear that any box $B \in S$ such that $\mathcal{R}^d(B) <_o \mathcal{L}^d(Q)$ cannot contain any (S, q)-intersection point, and hence can be removed from the data set. Second, we also know that any intersection Q of q boxes found provides an upper bound for any other direction $d' = (i', o')$: thus, we can ignore any box B such that $\mathcal{L}^{d'}(B) \geq_{o'} \mathcal{L}^{d'}(Q)$ when studying d'. The boxes removed by the second rule depend on the direction d' studied, which forces to use sets of allowed boxes that are specific to each direction. Figure 4–(a) illustrates both rules. The rule 1 definitely eliminates the rectangle 1 which is entirely in the left side of the hached region. The rule 2 is applied just before determining the bottom face of $\cap^4 S$ and states that we can ignore the rectangle 2, since it is completely in the top side of the hached area.

In addition, no (S', q)-intersection point may exist for any set S' of boxes such that $\forall B \in S'$, $\mathcal{L}^d(B) <_o \mathcal{L}^d(Q)$ (Figure 4). It may happen that subproblems to be solved in the remaining directions involve box sets with this property, and we want to avoid this situation. Thus, when one optimal intersection Q is determined for some direction $d = (i, o)$, the maximal set of boxes $S_d^N = \{B \in S | \mathcal{L}^d(B) <_o \mathcal{L}^d(Q)\}$ is stored as a nogood. Then, each time we solve a q-clique subproblem over some box set S', we perform a preliminary inclusion check $S' \subseteq S_d^N$ for all processed d, and skip the subproblem if at least one inclusion is true. Figure 4–(b) illustrates the nogood recording after having found an optimal 4-intersection for the left face. No subset S' of the colored boxes (whose left face appears before the darkened rectangle) can contain an (S', q)-intersection point.

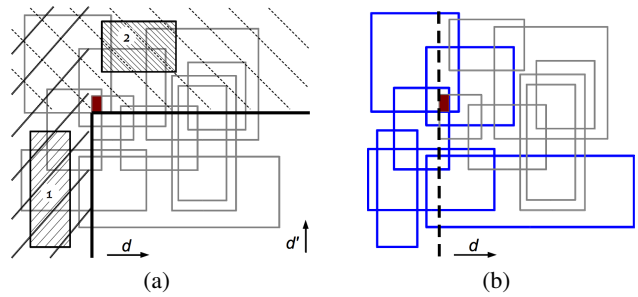


Figure 4: (a) Example of bound propagation and (b) nogood management after the determination of the leftmost 4-intersection (w.r.t d) of 2D boxes.

The algorithm The complete procedure for computing the q-intersection is summarized in Algorithm 2. H is the minimum-perimeter bounding box of the nonempty intersections of q boxes found by the algorithm, and is updated everytime a new intersection is found using the procedure `MinBoundingBox`. $\mathcal{N}$ is the set of nogoods. For every d, $\mathcal{A}[d]$ is the set of allowed boxes for the direction d. The subprocedure `FindMinQ` (Algorithm 3) is the algorithm detailed in the section "Strategy for one face". `Propagate` implements the propagation mechanisms discussed above (the two rules and nogood management). `Find_Clique` is a black-box clique solver that takes as input a graph and an integer k, and returns either a k-clique or $\emptyset$ if none exists. The correctness of `QInter2` follows from (3).

Algorithm 2: QInter2(S, q)

Data: A set S of boxes, an integer q

$H \leftarrow \emptyset$; $\mathcal{N} \leftarrow \emptyset$
for $d \in$ *all directions* **do** $\mathcal{A}[d] \leftarrow S$
for $d \in$ *all directions* **do**
  $Q \leftarrow$ FindMinQ$(\mathcal{A}[d], q, d, \mathcal{N})$
  if $Q \neq \emptyset$ **then** $H \leftarrow$ MinBoundingBox(H, Q)
  else if d *is the first direction* **then return** $\emptyset$
  $(\mathcal{A}, \mathcal{N}) \leftarrow$ Propagate$(S, \mathcal{A}, H, Q, d, \mathcal{N})$
return H

Algorithm 3: FindMinQ$(S, q, d, \mathcal{N})$

Data: A set S of boxes, an integer q, a direction d, a set of nogoods $\mathcal{N}$
Result: An intersection Q of q boxes optimal for d

Order S in the ascending order according to $>_d$
for $j = q..|S|$ **do**
  $B \leftarrow S[j]$; $N \leftarrow \{B' \in S | B >_d B', B \cap B' \neq \emptyset\}$
  if $\nexists N' \in \mathcal{N}$ *s.t.* $N \subseteq N'$ **then**
    $G \leftarrow$ The intersection graph of N
    $C \leftarrow$ Find_Clique$(G, q-1)$
    if $C \neq \emptyset$ **then return** $\bigcap_{B_i \in C} B_i$
return $\emptyset$

5 Implementation and first experiments

We have implemented the projective filtering, `coreF` and `QInter2` in the C++ numerical constraint programming library `Ibex` (Chabert and Jaulin 2009; Chabert 2014), which originally shipped with the grid algorithm. Q-clique problems are solved using `Cliquer` (Ostergard 2002). All experiments have been performed on a 2.2Ghz Intel Core i7.

Comparison of exact algorithms We have compared `QInter2` with the grid algorithm on randomly generated instances. The center of each of the p boxes in S and its radius in every dimension are uniformly drawn in $[0, L]$. In practice, the instances obtained using this method exhibit an abrupt phase transition: there exists a value q_c such that for $q << q_c$, $\cap^q S$ is roughly $[0, L]^n$, for $q >> q_c$ $\cap^q S$ is trivially empty, and for $q \sim q_c$ the q-intersection problem is very hard. The value q_c can be controlled by multiplying the size of each box by a scaling factor. The two values of q_c we have studied are $0.3\,p$ (many outliers) and $0.8\,p$ (few outliers). The corresponding scaling factor (for transforming each random instance into a difficult one) is determined empirically. We have run the grid algorithm and `QInter2` on a selection of such instances, with $q \sim q_c$, and Table 1 summarizes the runtimes obtained.[2] Heuristics are not included as they always return a large and inaccurate enclosure of $\cap^q S$.

n	p	$q\%$	grid (Ibex)	QInter2
2	100	0.8	0.67	0.01
2	300	0.8	17.8	0.57
2	600	0.8	147	8.53
6	100	0.8	>3,600	0.05
6	500	0.3	>3,600	3.55
6	500	0.8	>3,600	16.6
15	500	0.3	>3,600	213
25	100	0.8	>3,600	0.13
25	200	0.3	>3,600	1.98
25	200	0.8	>3,600	1.58
25	300	0.3	>3,600	379
50	100	0.8	>3,600	0.42

Table 1: Comparison of the grid algorithm and QInter2 on random instances. Running times are expressed in second.

In the light of these results it is clear that `QInter2` significantly outperforms the grid algorithm, which is extremely inefficient in high dimension. Furthermore, during the experiments used to determine the correct scaling factor, it has appeared that `QInter2` solves immediately easy instances (where q differs from q_c) while the grid algorithm is still very slow. Note that we cannot deduce an accurate correlation between values of parameters and the performance of our algorithm, as this experiment is only based on a preliminary sampling of difficult instances.

Comparison of heuristic propagators In this experiment we have compared our heuristic `coreF` with projective filtering in the broader context of parameter estimation. The n-dimensional localization problem is defined by a target $T \in \mathbb{R}^n$ of unknown position and a set of beacons, whose coordinates are known. Each beacon provides an estimation of the distance between itself and T, possibly erroneous, and the goal is to compute the coordinates of T assuming at least q beacons provided valid estimations.

This problem can be solved using constraint programming and q-intersection. The algorithm is a standard tree search where q-intersection is used multiple times at each node to prune search space. The reader can refer to (Jaulin and Bazeille 2009) for a detailed description of the procedure.

In our instances, T and the positions of the beacons are drawn uniformly in $[0, L]$. Then, slightly more than q beacons make a correct measurement and return a thin interval that contains the right value. The other beacons return a random thin interval. We ran the optimization procedure using both heuristics, and measured the size of the search tree (to estimate the filtering power of the q-intersection algorithms) and the total running time. Table 2 summarizes our results.

			projF		coreF	
n	p	$q\%$	time (s)	# br	time (s)	# br
2	500	0.2	0.23	39	**0.21**	23
2	500	0.5	**0.08**	15	0.17	13
2	500	0.8	**0.04**	13	0.21	13
4	500	0.2	147	25,944	**83.5**	9,770
4	500	0.5	**0.48**	84	1.03	46
4	500	0.8	**0.08**	19	0.58	18
6	200	0.2	2,200	727,134	**1,292**	355,999
6	200	0.5	22.13	9,070	**18.74**	3,598
6	200	0.8	**1.02**	401	2.68	275

Table 2: Time to compute the position of T using various q-intersection algorithms. projF denotes the projective filtering, and # br is the number of branches in the search tree.

Our experiments show that the use of `coreF` over projective filtering is beneficial on the difficult instances tested (which happen to be the ones with the largest number of outliers), cutting both the size of the search tree and running times. However, easy instances are better handled by projective filtering. Exact algorithms are not included as they are generally not competitive, even though `QInter2` beats projective filtering on the hardest instance tested (n=6,p=200).

6 Conclusion

This paper has presented the first theoretical analysis of the q-intersection operator. We have proven that `QINTER-CHECK` is DP-complete and proposed incomplete and exact algorithms to handle the q-intersection problem. Experiments on instances randomly generated confimed that our `QInter2` algorithm outperforms the existing grid algorithm and suggest that our new heuristic does well on problems with many outliers. A future work would embed our new algorithms in real robust parameter estimation applications.

[2] The code and instances of our experiments are available in the release 2.1.2 (and subsequent) of IBEX (www.ibex-lib.org).

References

Batagelj, V., and Zaversnik, M. 2011. Fast Algorithms for Determining (generalized) Core Groups in Social Networks. *Advances in Data Analysis and Classification* 5(2):129–145.

Beck, J. V., and Arnold, K. J. 1977. *Parameter Estimation in Engineering and Science*. Wiley series in probability and mathematical statistics. John Wiley & Sons.

Benhamou, F.; Goualard, F.; Granvilliers, L.; and Puget, J.-F. 1999. Revising Hull and Box Consistency. In *Proc. ICLP*, 230–244.

Chabert, G., and Jaulin, L. 2009. Contractor Programming. *Artificial Intelligence* 173:1079–1100.

Chabert, G. 2014. `www.ibex-lib.org`.

Chandran, L.; Francis, M.; and Sivadasan, N. 2010. Geometric Representation of Graphs in Low Dimension Using Axis Parallel Boxes. *Algorithmica* 56(2):129–140.

Chew, P., and Marzullo, K. 1991. Masking failures of multidimensional sensors. In *Reliable Distributed Systems, 1991. Proceedings., Tenth Symposium on*, 32–41.

Graham, R. L.; Grötschel, M.; and Lovász, L., eds. 1995. *Handbook of Combinatorics (Vol. 1)*. Cambridge, MA, USA: MIT Press.

Jaulin, L., and Bazeille, S. 2009. Image Shape Extraction using Interval Methods. In *Sysid 2009*.

Jaulin, L.; Walter, E.; and Didrit, O. 1996. Guaranteed Robust Nonlinear Parameter Bounding. In *CESA'96 IMACS Multiconference (Symposium on Modelling, Analysis and Simulation)*, 1156–1161.

Lhomme, O. 1993. Consistency Techniques for Numeric CSPs. In *Proc. IJCAI*, 232–238.

Ostergard, P. R. 2002. A Fast Algorithm for the Maximum Clique Problem. *Discrete Applied Mathematics* 120(1–3):197–207.

Papadimitriou, C. H., and Yannakakis, M. 1982. The Complexity of Facets (and Some Facets of Complexity). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, STOC, 255–260. New York, NY, USA: ACM.

Roberts, F. S. 1969. On the boxicity and cubicity of a graph. *Recent Progresses in Combinatorics, Academic Press, New York* 301–310.

Walter, E., and Piet-Lahanier, H. 1993. Guaranteed Linear and Nonlinear Parameter Estimation from Bounded-error Data: A Survey. In *ISCAS*, 774–777.