Robust Algorithms and EP theorems

Michel Habib,
LIAFA, Paris Diderot

Algorithm and Pretty Theorems, Feb. 8-12 at IHP, Paris

8 février 2010

Schedule of the talk

A good axample

Kurt Mehlhorn and his group working on LEDA Library have a programm for planarity testing :

- **YES** Answer = a planar drawing
- **NO** Answer = "This graph is not planar"

Only two years after they realized that the programm has a flaw (or a bug)

A good progamm for planarity testing :

- **YES** Answer = a planar drawing
- **NO** Answer = "an obstruction $K_{3,3}$ or K_5"

Based on Kuratowski's theorem, provoding a certificate that can be checked in $O(n+m)$ in both cases.

How this idea can be expressed ?

Main ideas = certificate and testing.
One can find in the litterature two close notions

First a pragmatic way

Algorithms with certificates ¨easy¨ to check. Certifying algorithms
Easy = polynomial.

The second approach coming from algorithmic complexity

Les (Existencially Provable) EP theorems, J. Edmonds 1990.
Robusts algorithms, J. Spinrad 2002.

Main references

- K. Cameron, J. Edmonds, Existencially polytime theorems, Dimacs Series in DMTCS, (1990), 83-100.
- D. Kratsch, R.M. Connell, K. Mehlhorn, J. Spinrad, Certifying algorithms for recognizing interval graphs and permutation graphs, SODA 2003.
- V. Raghavan, J. Spinrad, Robust algorithms for restricted domains, J. of Algorithms 48 (2003) 160-172.
- J. Spinrad, Efficient graph representations, Fields Institute Monographs, 2003.
- H. Wasserman, M. Blum, Software reliability via run-time Result checking, JACM 44 (1997) 826-849.

Certificate versus Proof

Each time a programm is used, one can check its result by testing a certificate given by the algorithm

Else we should :

- Prove the algorithm (using invariants)
- Proving the transformation from an algorithm to a programm
- Prove the programm itself (Data structures . . .)
- Be confident to (or prove) the compiler, the Operating System . . .

2-coloration

YES answer : a bipartition of the vertices into 2 independent sets $O(n+m)$
NO answer : an odd cycle $O(n)$

Bad cases

When the certificate is the algorithm itself.

Good cases

The two certificates for YES and NO Answers can be checked independently from the algorithm.

Very good cases

Algorithms **very easy to check** : the certificates can be tested within an algorithmic complexity not greater than the algorithm

Some examples

- 2-colorable graphs (bipartite) ($O(n+m), O(n)$).
- Cographs ou P_4-free graphs ($O(n+m), O(1)$).
- Interval graphs , permutation graphs ($O(n+m), O(n)$). (Kratsch et al 2003)

Minimum spanning trees

A spanning tree can be produced in $O(n + mlogn)$

Checking its minimality can be done in $O(n + m)$ (Good exercise)

Shortest paths SSSP

Dijkstra's algorithm computes in $O(n + mlogn)$ a tree T rooted in s providing a path form s to the others vertices.

The minimality of T can be checked in $O(n + m)$.
$\forall e = (x, y) \in G - T$, $d_T(y) \geq d_T(x) + \omega(e)$

Characterisitic linear ordering of the vertices

Many graph algorithms can be seen as the computation of some characteristic ordering of the vertices.
Examples : simplicial elimination scheme, transitive orientation, chordal graphs, interval graphs, unit interval gaphs, permutation graphs, cographs, distance-hereditary graphs, factoring permutation for modular decomposition. . . .

2-step algorithms

1. Computation of an ordering of the vertices suposed to have some property α
2. Testing the property α.

These algorithms often produces certifying algorithms

Other known certifying algorithms

- Interval graph recognition, permutation graph recognition (Krastch et al 2003)
- Boolean matrices having the 1-consecutiveness property. (McConnell 2004).
  An associated graph is bipartite iff the matrix has the property.

Definitions

Chordal graph

A graph is chordal if has no cycle of lentgth ≥ 4 without a chord.

Simplicial vertex

A vertex is simplicial if its neighbourhood is a clique.

Simplicial elimination scheme

$\sigma = [x_1 \dots x_i \dots x_n]$ is a simplicial elimination scheme $\forall i$, x_i is simplicial in the subgraph $G_i = G[\{x_i \dots x_n\}]$

Usefull Characterisations

Characterization [Dirac 1961]

A graph is chordal iff it admits a simplicial elimination scheme.

Characterization LexBFS [Rose, Tarjan et Lueker 1976]

A graph G is chordal iff every "backwards" LexBFS ordering of G is simplicial.

LexBFS

Lexicographique Breadth First Search

Données: A graph $G = (V, E)$ and a starting vertex s

Résultat: A totla ordering of the vertices σ de V

```
Affecter l'étiquette ∅ à chaque sommet
label(s) ← {n + 1}
pour i ← 1 à n faire
    Choisir un sommet v d'étiquette lexicographique max.
    σ(i) ← v
    pour chaque sommet non-numéroté w ∈ N(v) faire
        label(w) ← label(w).{n − i}
```

Certifying recognition

Step 1

Computation of LexBFS $LexBFS(x_n) = x_n, \ldots, x_i, \ldots, x_1$ in $O(n+m)$.

Step 2

Simplical elimination scheme can be checked in $O(n+m)$ for YES Cases

For negative answers

We found some x_i which has two non adjacent neighbours a and b

Consider the paths $\mu(a)$ joining a to x_n and $\mu(b)$ joining b to x_n. path coming form the underlying tree of the LexBFS.
Let z be the first commun vertex of these two paths. The cycle $[x_i, a, \ldots, z, \ldots, b, x_i]$ contains a cycle with no chord.

Can be done in $O(n)$.

Perspectives

- A certifying algorithm for path-graphs.
- Certifying modular decompostion algorithms ...

Extensions

- Similar questions for automaton (par ex : minimal automaton)
- Same notion for enumerating algorithms.
- Calcul du diamètre d'un grand graph.
- Probabilistic Certificates (N. Alon)
- Related works for hardware : Software reliability via run-time result checking, H. Wasserman, M. Blum, JACM 1997

Another way to consider algorithms

Let us apply these ideas to well-know problems (example searching in an ordered array).

The game is to obtain the lowest complexity

Ross Mc Connell is writing a book on this

Each time you write an algorithm, ask yourself :
Does there exists another way to validate the result ?

From algorithmic complexity theory

Only for decision problems

The symmetry of the answers YES–NO force us to consider only $NP \cap co - NP$

It is hard to certify that the value given by some heuristic is less than K-times the optimum value.

To compute a graph parameter k, as for example treewidth, we need an algorithm which produces either a value $\leq f(k)$, or a certificate than the certificate is greater than à k (using Brambles for treewidth).

Good characterizations

NP is the class of decision problems with a polynomial certificate for the YES Instances.

$NP \cap co - NP$ polynomial certificate in both cases
(Already in the first Jack's ideas in 1965)

Let us only consider only graph problems to illustrate (cf. using Fagin's characterization theorems for *P* et *NP*).

Famous conjecture

$P = NP \cap co - NP$? had important consequences.

- Linear Programming (Kachian, 1979)
- Primality testing (2002)
- Perfect Graphs recognition (2003)
- Parity Games ?
- Minimal Transversal ?

EP theorems, J. Edmonds 1990

An EP (Existentially Polytime) theorem is a theorem in which each condition is polynomially testable.

Exemples :

- A good characterisation
- un graph is not perfect iff it contains an odd hole or its complement.

EP theorems

Min-Max theorems

Flow max $=$ min cut

An optimal cut provides a certificate to a flow

C. Berge and Jack Edmonds

EP theorems

Sans vraiment l'écrire explicitement, J. Edmonds pense qu'un tel théorème implique l'existence d'un algorithme polynomial (au moins ceux du type $NP \cap co - NP$).
Example :perfect graph recognition (2003).

Robust algorithms, J. Spinrad 2002

For an NP-complete optimisation problem (ex : coloration), when considering a particular class $\mathcal{C}$ of graphs, a polynomial algorithm is called robust if it satisfies the following conditions :

Robust algorithms, J. Spinrad 2002

1. If the data belongs to the $\mathcal{C}$, the algorithm gives the good answer
2. Else :
   - Either the algorithm gives the good answer
   - or the algorithm answers that the dat does not belong the class $\mathcal{C}$ and provides a certificate of it.

Robust algorithms, J. Spinrad 2002

Examples :

- Trivial : computing a 2-coloration (or bipartite recognition)
- An algorithm easy to check in both cases is robust.

Paradox

Some robust algorithms are faster than the best recognition algorithm for the class $\mathcal{C}$!

Robust Algorithms, J. Spinrad 2002

Comparability graph recognition (graph having a transitive orientation).

Computation of a transitive orientation can be done in $O(n+m)$

But testing that this orientation is transitive, is a problem "equivalent" to boolean matrix multiplication.

Robust Algorithms, J. Spinrad 2002

A very interesting example

A linear robust algorithm for the computation of a maximum clique for comparability graphs

Sketch of the algorithm

1. Computation of a transitive orientation in $O(n + m)$
2. Computation of a longest path in $O(n + m)$
3. The certificate is this longest path and can be tested in $O(n + m)$.

Robust Algorithms, J. Spinrad 2002

Clique max of a comparability graph

To check if the algorithm has provided an optimum value can be done in $O(n+m)$, but in case of failure one has to check that the given orientation is correct.The certificate is based on the algorithm itself.

Robustness

Some problems do not have robusts algorithms (unless $P = NP$).
It remains many open questions :

Open problems

Maximal Clique for visibility graphs ?
Robust algorithms for particular instances of SAT ?
. . .