Counting in AT-free graphs : Independence and Domination

D. Kratsch

Laboratoire d'Informatique Théorique et Appliquée
Université Paul Verlaine - Metz
57045 Metz Cedex 01
France

DIMAP Workshop, University of Warwick, UK
April 4-7, 2011

I. Counting

Many algorithmic graph problems have natural versions as

- decision,
- optimization,
- counting, and
- enumeration problem.

Research in exact exponential algorithms has stimulated a new interest in the relation of the complexity of different versions of a problem.

Counting vs. Optimization

Counting versions of polynomial-time solvable decision or optimization problems might be #P-hard.

[Valiant 1979]

- Counting the number of perfect matchings in a bipartite graph is #P-complete.
- Computing the permanent of a $(0,1)$-matrix is #P-complete.

Counting vs. Optimization

Question:

- Consider an NP-hard optimization problem and a natural counting analogue.
- Is counting (much) harder than optimization?

Independent Set

- The best known algorithm to compute a maximum independent set of a graph runs in time $O(1.2109^n)$ [Robson 1986].
- The best known algorithm to count the number of k-independent sets of a graph, for all k, runs in time $O(1.2377^n)$ [Wahlström, private communication].

Counting vs. Optimization

Question:

- Consider an NP-hard optimization problem and a natural counting analogue.
- Is counting (much) harder than optimization?

Independent Set

- The best known algorithm to compute a maximum independent set of a graph runs in time $O(1.2109^n)$ [Robson 1986].
- The best known algorithm to count the number of k-independent sets of a graph, for all k, runs in time $O(1.2377^n)$ [Wahlström, private communication].

Counting vs. Optimization

Dominating Set

- The best known algorithm to compute a minimum dominating set of a graph runs in time $O(1.5048^n)$ [van Rooij et al. 2009].
- This algorithm counts the number of k-dominating sets of a graph, for all k, and thus solve this counting problem within the same running time.

Coloring

- The best known algorithm to compute the chromatic number of a graph runs in time $O^*(2^n)$ [Björklund Husfeldt FOCS 2006, Koivisto FOCS 2006].
- This algorithm can be modified to count the number of k-colorings of a graph, for all k within the same running time.

Counting vs. Optimization

Dominating Set

- The best known algorithm to compute a minimum dominating set of a graph runs in time $O(1.5048^n)$ [van Rooij et al. 2009].
- This algorithm counts the number of k-dominating sets of a graph, for all k, and thus solve this counting problem within the same running time.

Coloring

- The best known algorithm to compute the chromatic number of a graph runs in time $O^*(2^n)$ [Björklund Husfeldt FOCS 2006, Koivisto FOCS 2006].
- This algorithm can be modified to count the number of k-colorings of a graph, for all k within the same running time.

II. AT-free Graphs

Definitions

Asteroidal Triples [Lekkerkerker Boland 1962]

Three vertices of a graph form an asteroidal triple if between any two of these vertices there is a path avoiding the neighbourhood of the third.

AT-free Graphs

A graph is asteroidal triple-free (short AT-free) if it has no asteroidal triple.

Subclasses of AT-free graphs

Interval graphs [Lekkerkerker Boland 1962] A graph is an interval graph if and only if it is chordal and AT-free.

More Subclasses Cobipartite graphs, permutation graphs, trapezoid graphs and cocomparability graphs are subclasses of AT-free graphs.

AT-free Trees A tree is AT-free if and and only if it is a caterpillar.

Remark: Not all AT-free graphs are perfect graphs.

Structural Properties of AT-Free Graphs

[Möhring 1996]

Every minimal triangulation of an AT-free graph is an interval graph.

Dominating Pair Two vertices x and y form a dominating pair of a graph G if the vertex set of *every* x-y path is a dominating set of the graph G.

[Corneil Olariu Stewart 1997]

Every connected AT-free graph has a dominating pair (x, y) such that $d(x, y) = diam(G)$.

[Corneil Olariu Stewart 1999]

There is a linear-time algorithm (using 2LexBFS) to compute the dominating pairs of an AT-free graph.

Complexity of NP-hard Problems on AT-Free Graphs

Polynomial-Time Algorithms

- Independent Set [Broersma et al. 1999]
- Independent Dominating Set [Broersma et al. 1999]
- Dominating Set [K. 2000]
- Feedback Vertex Set [K. Müller Todinca 2008]
- 3-Coloring [Stacho 2010]

NP-complete

- Clique [Broersma et al. 1999]
- Partition Into Cliques [Broersma et al. 1999]
- Treewidth [Arnborg et al. 1987]
- Pathwidth [Arnborg et al. 1987]
- Bandwidth [Kloks et al. 1999]
- Min Fill-In [Yannakakis 1981]

Complexity of NP-hard Problems on AT-Free Graphs

Open

- Coloring
- Hamiltonian Path
- Hamiltonian Cycle

III. Our Results

Two #P-Hard Counting Problems

#k-IS

Given a graph $G = (V, E)$. For all $k = 0, 1, \ldots, |V|$ count the number of independent sets of size k.

Best known running time: $O(1.2377^n)$ [Wahlström 2010]

#k-DS

Given a graph $G = (V, E)$. For all $k = 0, 1, \ldots, |V|$ count the number of dominating sets of size k.

Best known running time: $O(1.5048^n)$ [van Rooij et al. 2009]

Counting in AT-Free Graphs

#k-IS

The k-independent sets of an AT-free graph can be counted for all $k = 0, 1, \ldots, n$ in polynomial time.

[Kijima Okamoto Uno 2010] Counting all dominating sets on cobipartite graphs is #P-hard.

This implies that #k-DS for all $k = 0, 1, 2, \ldots, n$ is #P-hard and that there is a k such that #k-DS is #P-hard.

#k-DS

The minimum dominating sets of an AT-free graph can be counted in polynomial time. Counting k-dominating sets in AT-free graphs for a fixed k, can be done in polynomial time if $k - \gamma(G)$ is a constant integer.

IV. Counting k-Independent Sets in AT-free Graphs

Maximum Independent Set

[Broersma et al. 1999]

There is an $O(n^4)$ dynamic programming algorithm to compute a maximum independent set of a given AT-free graph.

Dynamic Programming over Components and Intervals

Let $G = (V, E)$ be an AT-free graph.

x-components For every vertex x of G, the x-components are the connected components of $G - N[x]$.

Intervals Let x and y be two non adjacent vertices of G. Then the *interval* $I(x, y)$ is the set of all vertices z such that z belongs to the component of $G - N[x]$ containing y and z also belongs to the component of $G - N[z]$ containing x: $I(x, y) = C(x, y) \cap C(y, x)$.

Splitting Components and Intervals

Suppose we choose a vertex s to be added to the independent set and remove it and all its neighbours from a component or an interval of an AT-free graph.

Components Removing $N[s]$ from an x-component containing s, the vertex set of the remaining graph consists of $I(s, x)$ and s-components.

Intervals Removing $N[s]$ from an interval $I(x, y)$ containing s, the vertex set of the remaining graph consists of $I(s, x)$, $I(s, y)$ and s-components.

The algorithm computes a maximum independent set of the input graph via a dynamic programming on components and intervals in order of increasing size. Note that there are $\Theta(n^2)$ components and intervals.

Counting k-independent sets

Dynamic programming algorithm counts the number of k-independent sets for all $k = 0, 1, \ldots, n$, for all x-components and all intervals of G using equations based on the splitting theorems.

Let $\alpha_k(G[S])$ be the number of k-independent sets of $G[S]$. Compute for all components and all intervals the value of α_k for all k. Store all values in a vector $(\alpha_i(G[S])_{i=0..n})$.

Two-Component Lemma

$$\alpha_k(C_1 + C_2) = \sum_{i=0}^{k} \left(\alpha_i(C_1) \cdot \alpha_{k-i}(C_2)\right)$$

Time

- computing vector of two components from the vector of each component by n^2 additions and multiplications
- by splitting theorems to compute vector of a component or interval for all vertices s add vectors of at most n smaller components and intervals by successively adding two vectors
- for every s, $O(n^3)$ operations; in total $O(n^4)$ operations
- $O(n^2)$ components or intervals; in total $O(n^6)$ operations
- number to store needs at most n bits

Correctness

- when computing the vector of some component or interval, the algorithm sums over the values obtained for $G - N[x]$ over all vertices of the component or interval
- thus a set on k vertices is counted k times
- to avoid overcounting in the vector obtained for some component or interval, α_i is divided by i for all $i = 1, 2, \ldots n$.

V. Counting k-Dominating Sets

in AT-free Graphs

Minimum Dominating Set

[Kratsch 2000]

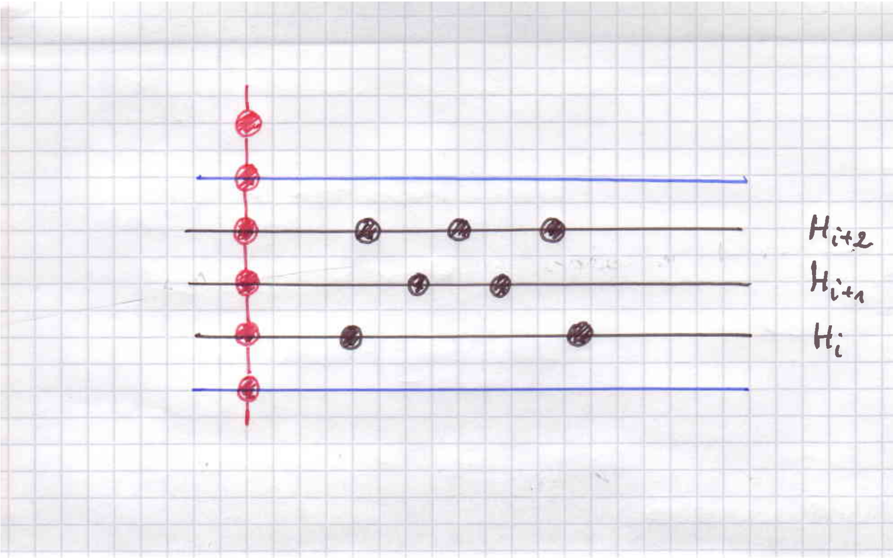
There is a $O(n^6)$ dynamic programming algorithm to compute a minimum dominating set of a given AT-free graph.

Dynamic Programming over the levels of a BFS-Layout

Take a 2LexBFS-layout of an AT-free graph. Then there is a minimum dominating set containing at most 5 vertices of every three consecutive levels. The algorithm searches for a dominating set satisfying this condition by dynamic programming over the levels of a 2LexBFS-layout.

Remark: Since the algorithm searches only for a particular (minimum) dominating sets it seems unlikely that it can be modified to count k-dominating sets.

2LexBFS-levels



Structure of Minimum Dominating Sets

Structure Theorem

Let G be a connected AT-free graph. Let $H_0 = \{x\}, H_1, \dots, H_i = \{y : d(y,x) = i\}, \dots, H_\ell$ be the 2LexBFS-levels of x. Then every minimum dominating set D of G contains at most 6 vertices of every three consecutive levels: for all $i = 0, 1, \dots, \ell - 2$: $|D \cap (H_i \cup H_{i+1} \cup H_{i+2})| \leq 6$.

Proof by contradiction

- There is path $P = x_0, x_1, \dots, x_\ell$ such that $x_i \in H_i$ for all $i = 0, 1, \dots, \ell$ such that every $y \in H_i$ is adjacent to x_{i-1} or x_i.
- Suppose there is a minimum dominating set D' and three consecutive BFS-levels such that $|D' \cap (H_j \cup H_{j+1} \cup H_{j+2})| \geq 7$. Remove the vertices of the three levels from D' and add $x_{j-1}, x_j, \dots, x_{j+4}$. The new set D'' is dominating and $|D''| < |D'|$, contradiction.

Extension to k-dominating sets

Theorem

Let G be a connected AT-free graph. Let $H_0 = \{x\}, H_1, \ldots, H_i = \{y : d(y,x) = i\}, \ldots, H_\ell$ be the 2LexBFS-levels of x. Let $k \geq \gamma(G)$. Then every k-dominating set D of G contains at most $6 + k - \gamma(G)$ vertices of every three consecutive levels: for all $i = 0, 1, \ldots, \ell - 2$: $|D \cap (H_i \cup H_{i+1} \cup H_{i+2})| \leq 6 + k - \gamma(G)$.

Counting dominating sets

Dynamic Programming over the 2LexBFS-levels

- There is an algorithm to count the number of minimum dominating sets of an AT-free graph in time $O(n^7)$ (unit-cost model).
- There is an algorithm to count the number of k-dominating sets of an AT-free graph in time $O(n^{7+(k-\gamma(G)})$ (unit-cost model).

VI. Conclusions

Counting independent sets

What means "counting independent sets"?

- Count the number of independent sets of G.
- Count the number of k-independent sets of G for all k
- Count the number of k-independent sets of G (k part of problem)
- Count the number of maximum independent sets of G
- Count the number of maximal independent sets of G

VII. References

A. Björklund and T. Husfeldt, *Inclusion-exclusion algorithms for counting set partitions*, Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), IEEE, 2006, pp. 575–582.

A. Brandstädt, V. B. Le, and J. Spinrad, *Special graph classes–A survey*, SIAM, 1999.

H. Broersma, T. Kloks, D. Kratsch, and H. Müller, *Independent sets in asteroidal triple-free graphs*, SIAM J. Discrete Math. 12 (1999), pp. 267–287.

D. G. Corneil, S. Olariu, and L. Stewart, *Asteroidal Triple-Free Graphs*, SIAM J. Discrete Math. 10 (1997), pp. 399–430.

D. G. Corneil, S. Olariu, and L. Stewart, *Linear Time Algorithms for Dominating Pairs in Asteroidal Triple-free Graphs*, SIAM J. Comput. 28 (1999), pp. 1284-1297.

F. V. Fomin, F. Grandoni, A. Pyatkin, and A. Stepanov, *Combinatorial bounds via Measure and Conquer: Bounding minimal dominating sets and applications*, ACM Transactions on Algorithms, Vol. 5, No. 1, Article 9, 2008.

F. Fomin and D. Kratsch, *Exact exponential algorithms*, Springer, Texts in Theoretical Computer Science, 2010.

M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the Theory of NP-completeness*, Freeman, New York, 1979.

M. C. Golumbic, *Algorithmic graph theory and perfect graphs*, Academic Press, New York, 1980.

S. Kijima, Y. Okamoto, and T. Uno, *Counting the number of dominating sets in graph classes*, Proceedings of 13th Japan-Korea Joint Workshop on Algorithms and Computation (WAAC 2010), pp. 40-46.

T. Kloks, D. Kratsch, and H. Müller, *Approximating the bandwidth for asteroidal triple-free graphs*, Journal of Algorithms 32 (1999), pp. 41–57.

T. Kloks, D. Kratsch, and H. Müller, *On the structure of graphs with bounded asteroidal number*, Graphs and Combinatorics 17 (2001), pp. 295–306.

M. Koivisto, *An $O^*(2^n)$ Algorithm for Graph Coloring and Other Partitioning Problems via Inclusion–Exclusion*, Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), IEEE, 2006, pp. 583–590.

D. Kratsch, *Domination and total domination on asteroidal triple-free graphs*, Discrete Applied Mathematics 99 (2000), pp. 111–123.

C. G. Lekkerkerker and J. Ch. Boland, *Representation of a finite graph by a set of intervals on the real line*, Fundamenta Mathematicae 51 (1962), pp. 45–64.

R. H. Möhring, *Triangulating graphs without asteroidal triples*, Discrete Applied Mathematics 64 (1996), pp. 281–287.

J. M. Robson, *Algorithms for maximum independent sets*, Journal of Algorithms 7 (1986), pp. 425–440.

J. van Rooij, J. Nederlof, and T. van Dijk, *Inclusion Exclusion Meets Measure and Conquer*, Proceedings of the 17th Annual European Symposium (ESA 2009), Lecture Notes in Computer Science 5757, Springer, 2009, pp. 554–565.

L. G. Valiant, *The Complexity of Computing the Permanent*, Theor. Comput. Sci. 8: 189-201 (1979)

M. Wahlström, private communication.