Faster Steiner Tree Computation in Polynomial-Space

F. Fomin[1] F. Grandoni[2] D. Kratsch[3]

[1]University of Bergen
Bergen, Norway

[2]Università di Roma "Tor Vergata"
Roma, Italy

[3]LITA
Université Paul Verlaine - Metz
Metz, France

ESA 2008, Karlsruhe, Germany
September 15-17, 2008

Exact Exponential Time Algorithms

(try to) cope with NP-hardness

- solve NP-hard problems by producing an (optimal) solution for all inputs
- worst-case running time analysis

- dating back to the early sixties [Davis Putnam 1960] and [Held Karp 1962]
- major design techniques : branching algorithms, dynamic programming, inclusion exclusion, treewidth based algorithms

Polynomial vs. Exponential Space

typical space requirements

- Branching algorithm : polynomial or even linear space
- Dynamic programming, Treewidth based : exponential space

Why polynomial space ?

- algorithms of very high space complexity unlikely to be fast in practice
- main reason : frequent external memory accesses
- phenomena not captured by standard RAM model

Polynomial vs. Exponential Space

Goal

Searching for algorithms with low memory requirements, even if they are asymptotically slower than their exponential-space counterpart.

For various NP-hard problems the best known exponential time algorithm need exponential space complexity :

Travelling Salesman problem, Coloring, Treewidth, Pathwidth, Steiner Tree, etc.

Polynomial-space exponential time algorithms :

Hamiltonian Path [Bax 1993, Gurevich Shelah 1987, Karp 1982],
Coloring [Björklund Husfeldt ICALP 2006,
Björklund Husfeldt FOCS 2006], etc.

The Steiner tree problem (ST)

The *Steiner tree problem* is a well-known optimization problem.

Steiner Tree problem (ST)

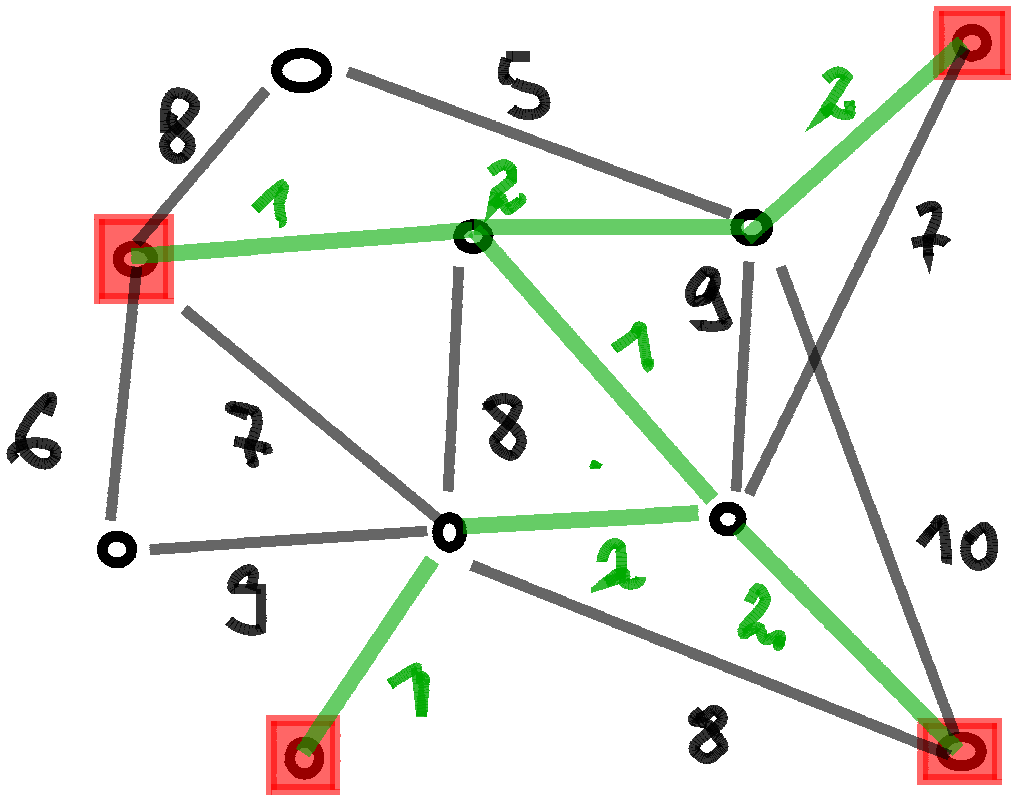
Input : Connected graph $G = (V, E)$ on $n = |V|$ nodes, edge costs $c : E \to \mathbb{R}^+$ and a set $T \subseteq V$ of $k = |T|$ *terminals*
(k natural parameter)

Objective : Find a subtree S of G spanning T such that the cost of S (i.e. the total cost of its edges) is minimum.

Cardinality Steiner Tree problem (CST)

(ST) where all edges of input instance have cost 1

Example



Fixed parameter tractable algorithms

FPT algorithms ($k \ll n$)

- classical $O^*(3^k)$ time dynamic programming algorithm [Dreyfus Wagner 1972] for (ST) :
  "probably most popular algorithm for solving different variants of the Steiner tree problem in practice"
- for any constant $\epsilon > 0$, $O^*((2+\epsilon)^k)$ time dynamic programming algorithm [Mölle et al. STACS 2006]
- $O^*(2^k)$ time algorithm using fast subset convolution for the version of (ST) where edges have bounded integer weights [Björklund et al. STOC 2007]

All known fixed parameter algorithms need exponential space
(for large k : $k = \omega(\log n)$)

An exponential-time polynomial space algorithm I

Observation

- Each leaf of any optimal Steiner tree is a terminal.
- Thus an optimal Steiner tree has no more than k Steiner nodes T' of degree greater than 2.
- Given T', the Steiner tree problem is equivalent to the polynomial time solvable minimum spanning tree problem on $G_M[T \cup T']$, where G_M is the metric closure of G.

Algorithm

List all subsets $T' \subseteq N := V \setminus T$ of size at most k, and for each one apply above observation.

An exponential-time polynomial space algorithm II

Time

$O(\sum_{i=1}^{k} \binom{n-k}{i} n^{O(1)})$ implying ...

for $k \ll n$, this running time is $O(n^k)$.

Running Time

The enumeration algorithm has running time $O(1.6181^n)$ (for arbitrary values of k).

Our results : Polynomial space algorithms

Divide & Conquer

- $O(5.96^k n^{O(\log k)})$ time and polynomial space algorithm using a simple variant of the classical tree-separator theorem
- works also for (ST)

Branching algorithm

- Branching algorithm with running time $O(1.5949^n)$ time needing polynomial space using a charging mechanism
- Measure & Conquer time analysis
- works only for (CST)

Outline

The Dreyfus Wagner algorithm : Main Idea

Let S be a Steiner tree of (G, T), $|T| \geq 3$. Take an internal node $s \in S$, not necessarily a terminal, such that the subtrees of S rooted at s can be partitioned in two forests $\mathcal{R}_1$ and $\mathcal{R}_2$, each one containing at least one terminal.

Let S be an optimum Steiner tree of (G, T), $s \in S$ an internal node and T_i be the terminals in $\mathcal{R}_i$, $i \in \{1, 2\}$. If we compute the optimum Steiner trees on $(G, T_1 \cup \{s\})$ and on $(G, T_2 \cup \{s\})$, and we merge them, we obtain an optimum Steiner tree for (G, T).

Of course we do not know s nor (T_1, T_2) a priori, but we can guess them by enumerating all the possible cases.

$\Rightarrow$ DYNAMIC PROGRAMMING

The Dreyfus Wagner algorithm : Dynamic Programming

Minimum cost

$st_G(T) = st(T)$ is the *minimum cost* of a Steiner tree of G on terminal set T

Recurrence D&W

$$st(T) = \min_{s \in V} \min_{(T_1, T_2) \in \mathcal{P}(s,T)} \{st(T_1 \cup \{s\}) + st(T_2 \cup \{s\})\}$$

where $\mathcal{P}(s, T)$ is the set of possible partitions (T_1, T_2) of $T \setminus \{s\}$ in two non-empty subsets

Dreyfus Wagner algorithm applies recurrence D&W to any subset of T, in a bottom-up fashion, storing each partial solution computed for later computations.

The Dreyfus Wagner algorithm : Space and Time Complexity

Exponential space

Storing the partial solutions takes $\Omega(2^k)$ space.

Time Complexity

For each subset T of size at most k, apply recurrence D&W. Thus the running time is

$$n \cdot \sum_{i=1}^{k} \binom{k}{i} 2^i = n \cdot 3^k$$

.

Bottom-up $\Rightarrow$ Top-down

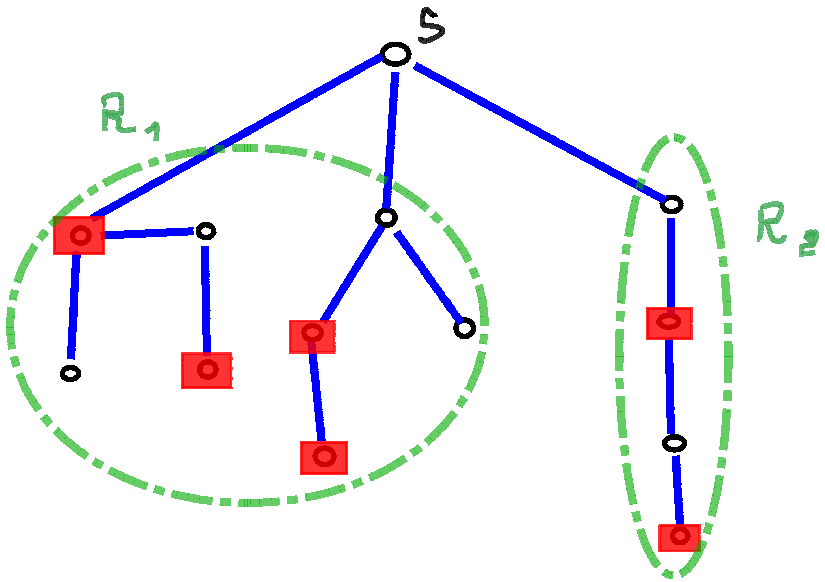
Easy transformation of Dreyfus Wagner algorithm into a polynomial-space algorithm ?

- Apply recurrence D&W recursively, in a top-down fashion, without storing any partial solution.
- Approach leads to a very high running time.
- The main reason is that, by applying recurrence D&W, one generates some subproblems with almost the same number of terminals as in the original problem.

Divide & Conquer : Tree separator theorem

Separator Lemma : [Bodlaender 1998]

Let S be a Steiner tree on the set of terminals T, $|T| = k \geq 3$. Then there exists an internal node $s \in S$ (Steiner-separator), not necessarily a terminal, whose removal divides the tree in two forests, each one containing at most $2k/3$ terminals.



Divide & Conquer algorithm I

By the separator lemma, in recurrence D&W we do not really need to consider all the partitions in $\mathcal{P}(s, T)$

It is sufficient to consider only the subset $\mathcal{B}(s, T) \subseteq \mathcal{P}(s, T)$ of ("almost balanced") partitions (T_1, T_2) where $|T_1| \leq |T_2| \leq 2k/3$:

balanced recurrence

$$st(T) = \min_{s \in V} \min_{(T_1, T_2) \in \mathcal{B}(s,T)} \{st(T_1 \cup \{s\}) + st(T_2 \cup \{s\})\}$$

Using only balanced partitions does not help in Dynamic Programming ...

Divide & Conquer algorithm II

Our recursive Steiner tree algorithm uses only "balanced partitions" :

(base case) If $T = \{v\}$, return v. If $T = \{v, w\}$, return the shortest path from v to w.

(recursive case) For every $s \in V$ and for every partition (T_1, T_2) of $T \setminus \{s\}$, $|T_1| \leq |T_2| \leq 2k/3$, compute recursively the optimum Steiner trees S_1 and S_2 over $T_1 \cup \{s\}$ and $T_2 \cup \{s\}$, respectively. Return the cheapest Steiner tree $S_1 \cup S_2$ obtained.

Divide & Conquer : Space and Time

The Steiner tree algorithm takes time $O((27/4)^k n^{O(\log k)})$ and polynomial space.

Basic ideas of time analysis

- recursion depth $O(\log k)$
- $P(k)$ number of base instances generated by the algorithm
- time complexity of algorithm
  $O(\log k\, P(k)\, n^{O(1)}) = O(P(k)\, n^{O(1)})$
- remains to bound $P(k)$

Divide & Conquer : Bounding $P(k)$

$$P(k) \leq n \sum_{i=k/2}^{2k/3} \binom{k}{i} (P(i+1) + P(k-i+1)) \leq 2n \sum_{i=k/2}^{2k/3} \binom{k}{i} P(i+1)$$

$$\leq 2n\, P(2k/3+1) \sum_{i=k/2}^{2k/3} \binom{k}{i}$$

By induction it can be shown that

$$P(k) \leq C n^{c \ln k} \alpha^k$$

for $\alpha = 27/4$ and sufficiently large constants $C > 0$, $c > 0$.

Branching Algorithm : Contraction

Cardinality Steiner Tree (CST) only !

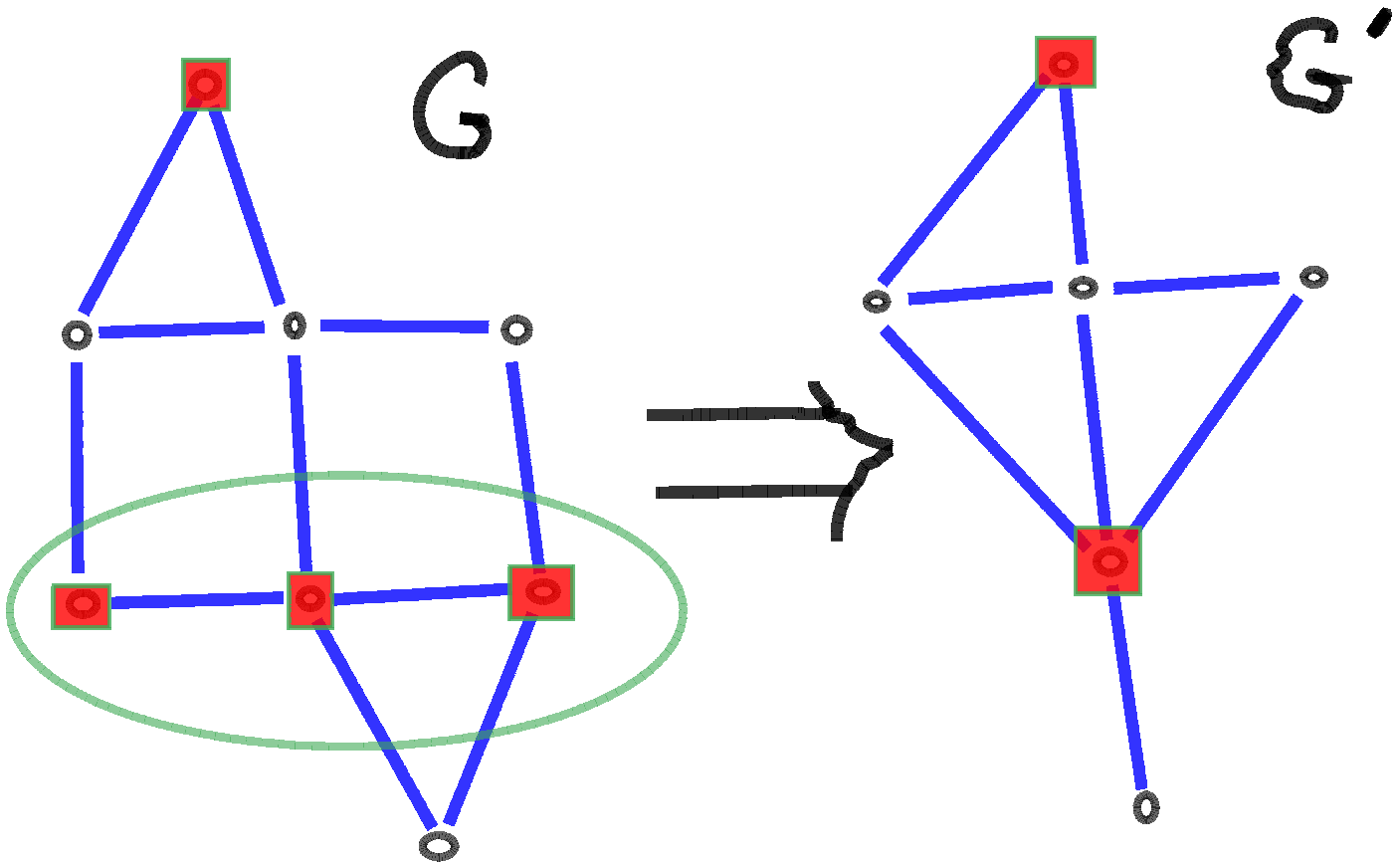
- $st_G(T)$ denotes the minimum number of edges of a Steiner tree of graph G spanning the terminal set T.
- *contracting* a subset of nodes V', means (i) removing V' from the graph, (ii) add a new node v', and (iii) add one edge between v' and each neighbor of V' not in V'.

Contraction Lemma

Let (G, T) be an instance of (CST). Let V' be a connected component of terminals, G' be the graph resulting from contracting V' in a unique node v', and $T' = T \cup \{v'\} \setminus V'$.

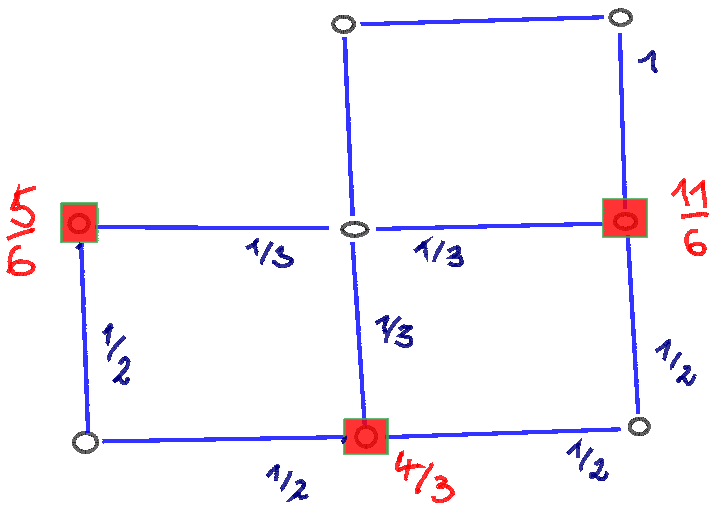
$$st_G(T) = |V'| - 1 + st_{G'}(T')$$

Example



Branching Algorithm : Charging

1. Initially assign load one to each non-terminal s.
2. Each non-terminal s evenly distributes its load among the terminals adjacent to it.
3. The final load $w(t)$ of each terminal t is the sum of the loads received by its non-terminal neighbors.



Branching Algorithm I

base If $|T| \in \{0, 1\}$, trivially $st_G(T) = 0$:

contraction If there is a connected component V' of at least 2 terminals, apply the contraction lemma.

reduction If there is a terminal t adjacent to a unique (non-terminal) node s, add s to the terminals since s must belong to any Steiner tree.

small k If $k \leq n/4$, apply $O(5.96^k n^{O(\log k)})$ Divide & Conquer polynomial space algorithm.

Branching Algorithm II

simple branch Assume a non-terminal s is adjacent to at least 3 terminals. Branch by either removing s from the graph, or by adding it to the terminals.

$$\texttt{steiner}(G, T) = min\{\texttt{steiner}(G \setminus \{s\}, T), \texttt{steiner}(G, T \cup \{s\})\}$$

multiple branch Let t be a terminal of minimum load $w(t)$ (always less than 3). Let $s_1, s_2, \ldots, s_p$ be the (not-terminal) neighbors of t, sorted in decreasing number of adjacent terminals. Branch to the p subproblems obtained by removing $s_1, s_2, \ldots, s_{i-1}$, and adding s_i to the terminals, for $i \in \{1, 2, \ldots, p\}$.

$$\texttt{steiner}(G, T) = \min_{i \in \{1,2,\ldots,p\}} \{\texttt{steiner}(G \setminus \{s_1, \ldots, s_{i-1}\}, T \cup \{s_i\})\}$$

Branching Algorithm : Time Analysis

Feasible values of $(m_1, m_2, \ldots, m_p)$ for multiple branch, with corresponding load (strictly smaller than 3) and number of nodes removed in each subproblem

$(m_1, \ldots, m_p)$	load	nodes removed
$(1,1)$	$4/2$	$1,2$
$(2,1)$	$3/2$	$2,2$
$(2,2)$	$2/2$	$2,3$
$(2,1,1)$	$5/2$	$2,2,3$
$(2,2,1)$	$4/2$	$2,3,3$
$(2,2,2)$	$3/2$	$2,3,4$
$(2,2,2,1)$	$5/2$	$2,3,4,4$
$(2,2,2,2)$	$4/2$	$2,3,4,5$
$(2,2,2,2,2)$	$5/2$	$2,3,4,5,6$

Branching Algorithm : Measure & Conquer

Measure & Conquer

- Measure of instance (G, T) of a subproblem is $h := n + n_N$ where n number of nodes of G and n_N the number of non terminals.
- Running time $O(1.6011^n)$ (and polynomial space)

Refined Analysis of same algorithm

- Measure of instance (G, T) of a subproblem is $h := n + \alpha\, n_N$ for proper $\alpha > 0$
- Choosing $\alpha = 0.7297$
- Running time $O(1.5949^n)$ (and polynomial space)

Conclusions I

Our result

$O(5.96^k n^{O(\log k)})$ time and polynomial space Divide & Conquer algorithm for (ST)

Open Problem

Fixed parameter tractable polynomial space algorithm for (ST)

Conclusions II

Our result

$O(1.5949^n)$ time polynomial space branching algorithm for (CST)

Open problems

- Extend branching algorithm to (ST)
- Polynomial space branching algorithm for (ST) being faster than enumeration ($O(1.6181^n)$)

Exponential space algorithms

Best known previous to our work : $O(1.4143^n)$ for (ST)

Combining enumeration (for large k) with the FPT algorithm of [Mölle et al. STACS 2006] (for small k)

$O(1.3533^n)$ for (CST)

- Branching algorithm : Replacing our polynomial space Divide & Conquer algorithm by FPT algorithm of [Mölle et al. STACS 2006]
- Measure & Conquer time analysis


S. Arora.
Polynomial time approximation schemes for Euclidean TSP and other geometric problems.
Journal of the Association for Computing Machinery, 45 :753–782, 1998.


E. T. Bax.
Inclusion and exclusion algorithm for the hamiltonian path problem.
Information Processing Letters, 47 :203–207, 1993.


M. Bern and P. Plassmann.
The Steiner tree problem with edge lengths 1 and 2.
Information Processing Letters, 32 :171–176, 1989.


A. Björklund and T. Husfeldt.
Exact algorithms for exact satisfiability and number of perfect matchings.
In *Proceedings of the 33nd International Colloquium on Automata, Languages and Programming (ICALP 2006)*, , volume 3580 of *Lecture Notes in Computer Science*, pages 191–203, Springer, 2006.


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


A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto.
Fourier meets Möbious : Fast subset convolution.
In *Proceedings of the 39th annual ACM Symposium on Theory of Computing (STOC 2007)*, pages 67–74, New York, 2007. ACM Press.


G. E. Blelloch, K. Dhamdhere, E. Halperin, R. Ravi, R. Schwartz, and S. Sridhar.
Fixed parameter tractability of binary near-perfect phylogenetic tree reconstruction.
In *Proceedings of the 33rd International Automata, Languages and Programming, Colloquium (ICALP 2006)*, volume 4051 of *Lecture Notes in Computer Science*, pages 667–678, Springer, 2006.


H. L. Bodlaender.
A partial k-arboretum of graphs with bounded treewidth.
Theoretical Computer Science, 209 :1–45, 1998.


M. Davis and H. Putnam.
A computing procedure for quantification theory.
Journal of the ACM, 7 :201–215, 1960.


L. L. Deneen, G. M. Shute, and C. D. Thomborson.
A probably fast, provably optimal algorithm for rectilinear Steiner trees.
Random Structures and Algorithms, 5(4) :535–557, 1994.


R. G. Downey and M. R. Fellows.
Parameterized complexity.
Springer-Verlag, New York, 1999.


S. E. Dreyfus and R. A. Wagner.
The Steiner problem in graphs.
Networks, 1 :195–207, 1971/72.


D. Eppstein.
Quasiconvex analysis of multivariate recurrence equations for backtracking algorithms.
ACM Transactions on Algorithms, 2(4) :492–509, 2006.


J. Flum and M. Grohe.
Parameterized Complexity Theory.
Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2006.


F. Fomin, F. Grandoni, and D. Kratsch.
Measure and conquer : domination - a case study.
In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 191–203, 2005.


F. Fomin, F. Grandoni, and D. Kratsch.
Measure and conquer : a simple $O(2^{0.288\,n})$ independent set algorithm.
In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 18–25, 2006.


B. Fuchs, W. Kern, D. Mölle, S. Richter, P. Rossmanith, and X. Wang.
Dynamic programming for minimum Steiner trees.
Theory of Computing Systems, to appear, 2007.


J. L. Ganley.
Computing optimal rectilinear Steiner trees : a survey and experimental evaluation.
Discrete Applied Mathematics, 90(1-3) :161–171, 1999.


M. R. Garey and D. S. Johnson.
The rectilinear Steiner tree problem is NP-complete.
SIAM Journal on Applied Mathematics, 32 :826–834, 1977.


M. R. Garey and D. S. Johnson.
Computers and Intractability. A Guide to the Theory of NP-Completeness.
Freemann, 1979.


J. Guo, R. Niedermeier, and S. Wernicke.
Parameterized complexity of generalized vertex cover problems.
In *Proceedings of the 9th International Workshop on Algorithms and Data Structures (WADS 2005)*, volume 3608 of *Lecture Notes in Computer Science*, pages 36–48. Springer, 2005.


Y. Gurevich and S. Shelah.
Expected computation time for Hamiltonian path problem.
SIAM Journal on Computing, 16(3) :486–502, 1987.


M. Held and R. M. Karp.
A dynamic programming approach to sequencing problems.
Journal of SIAM, 10 :196–210, 1962.


F. K. Hwang, D. S. Richards, and P. Winter.
The Steiner Tree Problem.
North-Holland, Amsterdam, 1992.


A. Kahng and G. Robins.
On Optimal Interconnections for VLSI.
Kluwer, Dordrecht, 1995.


R. M. Karp.
Dynamic programming meets the principle of inclusion and exclusion.
Operation Research Letters, 1 :49–51, 1982.


B. Korte, H. J. Prömel, and A. Steger.
Steiner trees in VLSI-layout.
In *Paths, Flows, and VLSI-Layout*, pages 185–214, 1990.


D. Mölle, S. Richter, and P. Rossmanith.
A faster algorithm for the Steiner tree problem.
In *Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 561–570, 2006.


R. Niedermeier.
Invitation to fixed-parameter algorithms, volume 31 of *Oxford Lecture Series in Mathematics and its Applications*.
Oxford University Press, Oxford, 2006.


H. J. Prömel and A. Steger.
The Steiner tree problem.
Advanced Lectures in Mathematics. Friedr. Vieweg & Sohn, Braunschweig, 2002.


G. Robins and A. Zelikovsky.
Improved Steiner tree approximation in graphs.
In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 770–779, 2000.


J. M. Robson.
Algorithms for maximum independent sets.
Journal of Algorithms 3 :425–440, 1986.


G. Woeginger.
Space and time complexity of exact algorithms : Some open problems.
In *Proceedings of the 1st International Workshop on Parameterized and Exact Computation (IWPEC 2004)*, volume 3162 of *Lecture Notes in Comput. Sci.*, pages 281–290. Springer-Verlag, Berlin, 2004.