Efficient Computations in Tree-Like Graphs

Andrzej Proskurowski
Maciej M. Syslo

CIS-TR-89-16
September 23, 1989

Department of Computer and Information Science
University of Oregon

Efficient Computations in Tree-Like Graphs

Andrzej Proskurowski [*]   Maciej M. Sysło [†]

September 23, 1989

Abstract

Many discrete optimization problems are both very difficult and important in a range of applications in engineering, computer science and operations research. In recent years, a generally accepted measure of a problem's difficulty became a worst-case, asymptotic growth complexity characterization. Because of the anticipated at least exponential complexity of any solution algorithm for members in the class of $\mathcal{NP}$-hard problems, restricted domains of problems' instances are being studied, with hopes that some such modified problems would admit efficient (polynomially bounded) solution algorithms. We survey investigations of the complexity behavior of $\mathcal{NP}$-hard discrete optimization problems on graphs restricted to different generalizations of trees (cycle-free, connected graphs.) The scope of this survey includes definitions and algorithmic characterization of families of graphs with tree-like structures that may guide the development of efficient solution algorithms for difficult optimization problems and the development of such solution algorithms.

1 Motivation

The framework in which we are interested in tree-like graphs consists of finding restricted classes of graphs on which many generally difficult decision and optimization problems are efficiently solvable (in the worst case). These graphs often exhibit some decomposability properties. Our own research has concentrated recently on algorithmic aspects of graph representations of orders and on partial k-trees, also known as

[*]Department of Computer and Information Science, University of Oregon, Eugene, Oregon 97403, USA. Research supported in part by the Office of Naval Research Contract N00014-86-K-0419.

[†]FB 3-Mathematik, TU Berlin. On leave from Institute of Computer Science, University of Wrocław, Przesmyckiego 20, 51151 Wrocław, Poland. Research supported by the grant RP.I.09 from the Institute of Informatics, University of Warsaw and by a grant from the Alexander von Humboldt-Stiftung.

graphs with tree-width k. Those graphs are all partial subgraphs of chordal graphs with the maximum clique size bounded by k. We will discuss alternative views of these graphs and also other families of graphs, classes of problems efficietly solvable on these graphs, and the relevant algorithm design paradigms.

Application domain problems often translate into optimization problems on the graphs representing the application; these *combinatorial problems* are often very difficult. A measure of complexity of a combinatorial problem is the *worst-case, asymptotic* behavior of the time to compute a solution as a function of the problem *size*. A class of problems notorious for their difficulty is that of $\mathcal{NP}$*-complete* problems.

Let us first state our vocabulary for discussing *discrete optimization problems* defined on *combinatorial graphs*. A (combinatorial) graph $G = (V, E)$ consists of the set V of *vertices* and the set E of *edges*, each edge *incident* with its two *end-vertices* (which are thus *adjacent*). A *subgraph* of a given graph $G = (V, E)$ *induced* by a subset of vertices $V' \subset V$ consists of all edges from E that are not adjacent to any vertex from $V - V'$. A *partial subgraph* on the same set of vertices involves a subset of edges. A sequence of different vertices $v_0, v_1, \ldots, v_n$ such that v_{i-1} and v_i $(0 < i \leq n)$ are adjacent is called a *path* of length n; if v_0 and v_n are identical $(n \geq 2)$, we have a *cycle*. A graph with n vertices and edges between all pairs of distinct vertices is called *complete* and denoted by K_n. A graph in which there exists a path between any two of its vertices is said to be *connected*; a set $S \subset V$ such that the subgraph of a graph $G = (V, E)$ induced by $V - S$ is not connected is called a *separator* of G. A *tree* is a connected graph without any cycles (it is easy to see that any minimal separator in a tree consists of exactly one vertex).

2 Definitions of Some Tree-Like Graphs

We first present definitions of several classes of graphs by their *generative* descriptions. Often they also have *analytical* descriptions based on a process of *decomposition*. Such a description gives a *parse tree* in which each node corresponds to a subgraph of the original graph.

2.1 Generative definitions of classes of graphs

Many families of graphs admiting recursive descriptions that invoke some kind of decomposability can be described by their *iterative construction*, often expressible by a recursive (hierarchical) construction rules as well. The former consists of primitive graphs and composition rules, the latter takes often a formal linguistic form.

The grammatical approach to defining families of graphs is exemplified by *context-free hyper edge replacement* grammars [76], [41], [39] [40], [49], [50], [51]. A grammar consists of a finite set of non-terminal labels N with a distinguished start label $s \in N$ and a finite set of rules, each having the *left-hand side*, a hyperedge with a label $a \in N$, and the *right-hand side*, a labeled hypergraph H. During an application of the rule, a hyperedge labeled a is replaced by the hypergraph H in such a way that some distinguished nodes of H ('terminals', 'sources') are identified with the corresponding nodes of the replaced hyperedge. The righ-hand side of a *terminal rule* has only unlabeled two-edges, so that the language of such grammar contains only combinatorial graphs. Below, we list some other formalisms following this approach in defining tree-like classes of graphs.

k-trees are defined recursively as either K_{k+1}, the completely connected graph on $k+1$ vertices, or two k-trees 'glued' along K_k subgraphs. Iterative definition includes K_{k+1} as the primitive graph and defines an $n+1$-vertex k-tree ($n > k$) as an n-vertex k-tree T augmented by an extra vertex adjacent to all vertices of a K_k subgraph of T (see [12]).

hook-up graphs generalize k-trees by allowing any base graph A and any subgraph B of a Hook-up(A, B) graph to which a new vertex is made adjacent. (Thus, k-trees are Hook-up(K_{k+1}, K_k) graphs, see [47].)

k-terminal graphs are closed with respect to a finite number of composition operations, where two graphs G_1 and G_2 with terminal's label sets $\{1..k\}$ each are composed either by identifying terminals in G_1 and G_2 that induce isomorphic subgraphs, or by adding edges between terminals in G_1 and terminals in G_2. The composition is completed by determination of terminal vertices in the new graph. A k-terminal family contains also basis graphs with all vertices terminal [82], [83], [84].

k-terminal recursive family involves recursive operations on graphs with at most k *terminal* vertices labeled $1 \ldots |T|$. An operation is determined by a *connection matrix* that indicates which of the input graphs' terminal vertices are identified to create a vertex of the resulting graph; some of these vertices are labeled terminals of the new graph. There is also a set of k-terminal base graphs with no non-terminal vertices, and a set of operations [18].

Any graph family defined by a context-free hyperedge replacement grammar (or an equivalent formalism) has bounded tree-width. For any given k, there is a context-free hyperedge replacement grammar generating all partial k-trees. Specifically, such a grammar would have $N = \{a\}$, the only label of a hyperedge with k vertices and the replacement rule set that consists of (terminal) rules substituting each of the $2^{\binom{k}{2}}$ edge combinations for the hyperedge and of the rule replacing such a hyperedge with a hypergraph consisting of $k+1$ hyperedges spanned on $k+1$ vertices (including the k original vertices). Similarly, all graphs generated by any k-terminal recursive family of graphs have bounded tree-width. It is also possible to generate all partial k-trees using that formalism but we will not show it here for the large size of rules. Actually, the formalisms of hyperedge replacement and that of k-terminal recursive family are equivalent.

2.2 Decomposability

2.2.1 Undirected graphs

k-trees are alternatively defined as connected graphs with no K_{k+2} subgraphs and with all minimal separators inducing K_k [68]. This property of the existence of small (constantly bounded) separators is inherited by their subgraphs, partial k-trees. Namely, partial subgraphs of k-trees are exactly the *k-decomposable* graphs: a graph G is k-decomposable if and only if it either has at most $k+1$ vertices, or has a separator S with at most k vertices ($G - S$ has $m \geq 2$ connected components $C_1, C_2, \ldots, C_m$) such that each graph G_i obtained from the component C_i ($1 \leq i \leq m$) extended by S with its vertices completely connected is k-decomposable (Arnborg and Proskurowski [5]). This motivates a closer look at partial k-trees, their recognition and embedding algorithms, as well as efficient algorithms solving $\mathcal{NP}$-hard optimization problems restricted to partial k-trees (for fixed k).

The *tree-width* of a graph G ([67]) is defined as one less than the maximum size of vertex sets $V_1, V_2, \ldots, V_m$ into which one can pack vertices of G (one vertex belonging possibly to more than one set) such that

- $V_1 \cup \ldots \cup V_m = V(G)$,
- edges are only between vertices in the same sets,
- G is representable by a tree T which has V_i as nodes and, for $V_i, V_j, V_l \in V(T)$, if there is a path in T between V_i and V_l containing V_j then $V_i \cap V_l \subseteq V_j$.

It is not too difficult to see that partial k-trees are exactly graphs with tree-width k (assuming that the definition calls for the minimum value of the parameter k): In one direction, given a partial k-tree, take an embedding k-tree and its $(k+1)$-cliques as V_i's. In the other direction, since V_i's intersect on at most k vertices, complete each V_i by adding edges between all pairs of its vertices, and then increase the cliques' sizes to $k+1$ by adding edges between some vertices of neighboring cliques.

By a similar construction, one can see that graphs generated by the other formalisms mentioned above have bounded tree-width, as well.

Hochberg and Reischuk, [43], define (k, μ)-decomposable graphs for which any decomposition into k-connected components yields μ as the maximum size of a component. It is again easy to see that these graphs have constantly bounded tree-width.

Lauteman [49,50] defines s-decomposition trees similarly to the parse trees of the above k-terminal graphs. He gives a finite set of *rewrite rules*, where left-hand-side graphs have distinguished terminal vertices to be identified with the corresponding terminal vertices of the right-hand-side graphs. The constant bound s is equal to the maximal size of the right-hand-side graphs of the rewrite rules.

2.2.2 Directed decomposable graphs

In the past decade, there has been very active research in the area of a general theory of set decomposition and in particular directed graph (digraph) decomposition. Cunningham and Edmonds, [29], discuss general decomposition of sets, followed by [28] who discusses digraph decomposition. A *composition* of two diagraphs G_1 and G_2 is defined as the digraph with the vertex set equal to union of their vertices (with the exception of a special vertex v repeated in both). The arcs of the resulting digraph reflect transitivity of arcs through v (*i.e.*, $(u_1, u_2) \in E$ if and only if $((u_1, v) \in E_1 \wedge (v, u_2) \in E_2) \vee ((u_1, v) \in E_2 \wedge (v, u_2) \in E_1)$. Each strongly connected digraph has a unique minimal decomposition into components that can be 'prime' (non-decomposable), 'brittle' (every partition is a *split*), or 'semi-brittle' (*circular splits*). Furthermore, the decomposition of semibrittle digraphs can proceed into distars and '*circles of transitive tournaments*'.

A special case of the composition operation is the *substitution*, when $G_2 = vG_2'$ is a *pointed* graph, with vertex v adjacent to all the remaining vertices of G_2. Graphs obtained by substitution can be described using the notion of '*autonomous sets*': A subgraph G_2 of G_1 is autonomous if and only if for every vertex u in G_2' either $\forall v \in V(G) : (u, v) \in E(G)$ or $\forall v \in V(G) : (u, v) \notin E(G)$. This treatment allows to include in the consideration graphs with disconnected components. There is also a unique decomposition theorem for autonomous sets: Each graph has the composition tree whose nodes are blocks of partitions into autonomous sets of the graph, each denoted by D (degenerate, disconnected, 'parallel'), C (complete, 'series'), or P (prime, decomposable arbitrarily into C and D components). Fast algorithms for finding decomposition of directed graphs into such subgraphs are presented in [19].

Decomposition often allows efficient solution algoritms for some discrete optimization problems. However, the range of those problems is severely limited in the split case requiring strong connectivity. Decomposition by substitution allows using the *divide and conquer* algorithms parallelling a natural factorization of objective functions in many discrete optimization problems. An excellent survey is given in [62]. A subsequent work ([42]) treats problems on posets and uses decomposition with prime elements of bounded size.

2.3 Other combinatorial structures with parse trees

Chordal graphs can be defined by a similar recursive construction (or decomposition) description as the graph families from section 2.1: Starting with a single vertex, any chordal graph (and only such graphs) can be constructed by adding a new vertex adjacent to all vertices of any complete subgraph of a chordal graph ([30], [36], [75]). Here, the generic definition of an *infinite set* of primitive graphs (K_k for *any* value of k) makes the major difference. Nevertheless, chordal graphs can be represented by their parse trees (clique trees) with help of which some algorithmic problems can be solved efficiently. Chordal graphs constitute an example of graphs decomposable by clique separators, however of unbounded size ([37,80]).

Chordal graphs can be interpreted as intersection graphs of subtrees in trees, see for instance Golumbic [38]. An important subfamily of chordal graphs consists of *interval graphs*, intersection graphs of intervals of a line. On the pther hand, class of intersection graphs not properly contained in chordal graphs is the class of circular-arc graphs, intersection graphs of intervals of a circle.

The existence of the unique parse tree (corresponding to the constructive definition of this class of graphs) contributes to the design of many efficient algorithms for *complement reducible* graphs. These are the graphs that can be reduced to single vertices by recursively complementing all connected subgraphs (Corneil *et al.* [23]).

3 Complexity of parsing of tree-like graphs

The problem of recognition and embedding of a partial k-tree for a fixed value of k is polynomially solvable, see Arnborg, Corneil and Proskurowski [2]. The algorithm recognizing a partial k-tree with n vertices has complexity $\mathcal{O}(n^{k+2})$; any lower bounds result on the complexity of the problem might help to explain difficulties with finding a system of confluent rewrite rules recognizing partial k-trees for $k > 3$. A related – and very important from the applications point of view – problem is that of finding the minimum value of k for which a given graph is k-decomposable (or, equivalently, is a partial k-tree). This problem is $\mathcal{NP}$-hard, as shown by Arnborg *et al.* [2].

Any sequence of applications of rewrite rules that reduce a given partial 2-tree to the empty graph determines also an embedding of the graph in a full 2-tree. This is so, because the reduction 'reverses' a feasible generation process of the full 2-tree. An application of a reduction rule can be thought of as 'pruning' of a *2-leaf* (vertex of degree 2) which is deleted, leaving as a trace an edge connecting its two neighbors. A similar pruning of 3-leaves (completion of a triangle spanned on neighbors of a vertex of degree 3, in a 'star-triangle substitution' process) in recognition of partial 3-trees must be done with care, since not all vertices of degree 3 in a partial 3-tree can be 3-leaves of an embedding in a full 3-tree, and an indiscriminate pruning may lead to irreducible graphs (Arnborg and Proskurowski [8]).

The system of confluent rewrite rules reducing any partial 3-tree (and only a graph from this class) to the empty graph allows for an efficient (but not linear) recognition of a partial 3-tree and construction of its embedding in a full 3-tree. One could describe those rules reducing vertices of degree 3 as based on a combination of 'strength' of their neighborhood (existing edges between their neighbors), and of 'relation' to other vertices of degree 3 (the nature of shared neighborhoods with those vertices). The rewrite rules are given in Arnborg and Proskurowski [8]. Thus, one could suspect that for a safe reduction of a vertex v of degree k in a partial k-tree G, there seems to be required certain trade-off between the amount of mutual connection among the k neighbors of v, the number of other vertices of degree k sharing their neighborhood with v, and the strength of this sharing. For some general rules see Arnborg and Proskurowski [6].

Attempts to generalize this approach to higher values of k have not brought any success, so far. A reason might be that while the two abovementioned rules of thumb are straightforward enough for $k = 3$ (although already the 'cube rule' is a strange mixture of the two), the sheer number of combinations to consider for $k > 3$ is difficult to handle. Another reason might be that such a complete system of confluent rewrite rules does not exist for higher values of k.

4 Problems with efficient solution algorithms on tree-like graphs

Discrete optimization problems that do not involve counting and that are defined on graphs, can be viewed simply as graph properties that a given graph has or does not have. Typical examples are 2-colorability ('Is a given graph 2-colorable?') and Hamiltonicity ('Does a given graph have a Hamilton cycle?'). These properties can be expressed as well-formed formulea in some formalism utilizing variable symbols, relational symbols (over some domains), logical connectives, and quantifiers. Depending on the restrictions on the use of these symbols, one defines languages of varying descriptive power. For instance, one could restrict relations to a single domain or use many-sorted structures, allow only existential quantification, restrict the domains of quantifiers, and so on. It is important to find formalisms that balance their power of expression and the ease of analysis (the complexity of property recognition).

In [24], Courcelle presents an excellent survey of the interaction between logic languages and graph properties, defining and analyzing the power of First Order Logic, Second Order Logic, Monadic Second Order Logic, and their extensions.

First Order Logic: The domain: graph elements (vertices and edges).
Basic relations: $V(x), E(x), R(x, y, z)$ denoting vertex set, edge set, and edge with incident vertices, respectively.
Quantification: over domain variables.
Examples: A given graph labeling is a proper coloration. All vertices have degree bounded by a given integer.

Second Order Logic: Variables: graph elements, relations over graph elements.
Quantification: over binary relation variables (and, consequently, over relational variables of any arity).
Example: Two given graphs are isomorphic.

Monadic Second Order Logic: Restriction: relational variables denoting sets only (relations on one variable).
Examples: A given graph is Hamiltonian. A given graph is m-colorable.

Although First Order Logic is a rather weak formalism as far as the expressive power is concerned, it is in general undecidable whether a general graph has a property described in this language. Thus, an interesting avenue of investigations is to consider the status of problems defined in these formalisms but restricted to some narrower classes of graphs. For instance, when applied to context-free hyperedge replacement graphs, even Monadic Second Order Logic (*MSOL*) is decidable. When the class of graphs is restricted to confluent Node Label Controled graphs, (*NLC* [70]), Monadic Second Order Logic with quantification only over vertex sets ($MSOL_v$) is decidable. Thus, it makes sense to inquire about the computational complexity of such problems on those graphs. An important connection between investigations of decidability of logical theories and the tree-like graphs is established by the following statement: 'For a property described by a Monadic Second Order Logic expression, one can decide in polynomial time whether a given partial k-tree has this property.' ([27],

[3].) Similarly, if the property is expressed in *MSOL*$_v$ and the graph belongs to the class of confluent *NLC* language, then there exist efficient decision algorithms, as well.

To be able to deal with discrete problems optimizing over some objective functions, the *MSOL* formalism has been extended by Courcelle [26] and by Arnborg *et al.* [3] allowing counting set cardinalities, and evaluating sums of functions of sets, respectively. Thus, the properties in the above statement have to be extended to those described by *EMSOL* and *CMSOL* expression, respectively.

The importance of the bounded tree-width is shown by the following theorem of Seese [73]: Any class of graphs that has a decidable *MSOL* property has bounded tree-width.

Arnborg *et al.* [3] present a detailed description of applications of *MSOL* to partial k-trees. The authors' main result is the efficient solvability of a number of problems $\mathcal{NP}$-hard for general graphs. They prove it constructively by reducing decidability of an *EMSOL* property for a partial k-tree G to the problem of deciding the corresponding, linearly definable property of a binary tree representing parsing of G (its 'tree decomposition'). For the latter problem, they construct a tree automaton that computes a solution in linear time. This tree automaton is found using results about Decision Problems in *SOL*, obtained in the 1960's (Doner [31], Thatcher and Wright [81]). Since the transformation itself (the derived property) is linear and the parse tree is assumed to be given with the input graph, a linear time solution algorithm for the original problem is obtained.

An interesting exception to the spirit of the recent results on efficient algorithms for problems on partial k-trees is the polynomial-time algorithm for the graph isomorphism problem (Bodlaender [16]), since that problem is *not* expressible by the proposed extensions to *MSOL*.

We should mention other recent attempts to characterize problems solvable efficiently on partial k-trees, notably Bodlaender's [17] and Scheffler's [71]. Each of those authors defines languages for some 'locally verifiable' properties, extends them by conjuctions with some 'non-local' statements (designed mainly to deal with the notion of connectivity), and designs a paradigm for constructing a solution algorithm for a given property and a given bound k on the tree-width of the problem instance.

Scheffler [71] considers optimization problems that can be described by formulea involving predicates expressing properties of a bounded neighborhood of a vertex. These are existentially quantified over a fixed set of subgraphs and universally quantified over all vertices of a graph. (The author follows the aproach introduced by Seese [73].) She extends the class of properties by allowing conjuction with connectedness and acyclicity, and presents algorithm paradigms for deciding the above properties for a partial k-tree given together with an ordering of vertices corresponding to a perfect elimination of an embedding chordal graph. These algorithms use the given ordering of vertices and combine the properties of individual vertices (expressed by values of the objective function) into the global answer. Assuming an additive objective function, the corresponding optimization problem is solved following the general dynamic programming strategy.

The time complexity of these algorithms, while linear in the size of the input graph, depends exponentially on the problem (the number of subgraphs defining the property) and on the parameter k defining the class of input graphs.

Borie *et al.* [18] define *regular* properties of graphs based on the existence of a homomorphism between members of a given k-terminal, recursive family of graphs and some finite set. These properties are preserved under the homomorphism and the integrity of composition operators is maintained. (Their definition follows that of Bern *et al.* [11].) They prove constructively that the recognition, optimization, and enumeration of solutions for a given regular property are linearly solvable on recursively consructed graph families.

Monien *et al.* [60] use the notion of tree-width to investigate completeness for the class of languages that are acceptable by non-deterministic auxiliary push-down automata in polynomial time and logarithmic space (equal to $LOGCFL$ complexity class). They define the tree-width of a conjunctive form of a prepositional formula as the tree-width of the corresponding hypergraph and show that many algorithms reducing 3-SAT with bounded tree-width preserve this bound for the instances of problems to which 3-SAT is reduced. This allows them to show these problems to be $LOGCFL$-complete when restricted to instances with tree-width bounded by $\log n$.

5 Algorithm design paradigm

Already 19th century physicists knew that certain difficult problems, hopeless in general can be solved in som 'tree-like graphs': the series-parallel reduction computing the equivalent resistance of a ladder circuit, or the star-triangle replacement in other electrical networks. However, the theory of these operations had to wait until 1980's. Slisenko [76] observed that the Hamilton cycle problem can be efficiently (in time polynomial in the size of the graph) solved on graphs obtained by the context-free replacement of hyperedges by hypergraphs, with terminal replacements of a hyperedges by edges between some of its vertices. Takamizawa *et al.* [79] developed a methodology for solving many such hard problems ($\mathcal{NP}$-hard) in linear time on series-parallel graphs. Intuitively, this 'good' algorithmic behavior of partial 2-trees can be explained by their *bounded decomposability* property that follows from a separation property of ('full') 2-trees: every minimal separator consists of both end-vertices of an edge.

The approach taken by Arnborg and Proskurowski in [5] to attack hard discrete optimization problems restricted to partial k-trees given with their embedding follows the general dynamic programming strategy. In a k-decomposable graph, the decomposability structure (an embedding in a full k-tree) is followed in solving pertinent subproblems. Solutions to these subproblems mutually interact only through the bounded interface of a minimal separator. Assuming that in the instances of discrete optimization problems of interest there are many partial k-trees for relatively small values of k (say, about 10), the following algorithm paradigm for solving optimization problems on partial k-trees is of practical interest (Arnborg and Proskurowski [5]):

Depending on the problem being solved for partial k-tree G, each minimal separator S of a full k-tree embedding G is assigned a number of 'states'. Each such state represents constraints on a subproblem of optimization on the graph G_i (*cf.* the definition of decomposability), where feasible solutions agree on the graph induced in G by S. A solution to the problem corresponding to a state of S associates with the state the optimal value of the objective function. The algorithm requires successive 'pruning' of the k-leaves of the embedding k-tree (and of the resulting k-trees). In each pruning step, it solves the corresponding subproblems and updates the values of states of the corresponding minimal separator. When pruning a k-leaf v, this state update of the separator S (the remaining neighbors of v) involves combination of solutions to k subproblems (represented by the k separators of G consisting of v and $k-1$ vertices of S). To find a solution to the overall problem, the eventual 'root optimization' is necessary, whereby the states of the k minimal separators constituting the definitional K_{k+1} root of the embedding full k-tree are combined to yield the solution. If the problem being solved admits a constant (in n) time pruning steps and a constant time 'root optimization', the resulting algorithm is linear in the size of the input graph. (So do for instance, *Independent Set, Vertex Cover, Chromatic Number, Graph Reliability*, *cf.* Arnborg and Proskurowski [5].) This follows from the fact that the number of states is independent of the size of the graph (although it can grow quite rapidly with k), and the number of minimal separators to consider is only linear with the size of the graph. It is important to realize, that the low order polynomial time complexity of the algorithm is achieved when the input consists of a suitable embedding of the given graph in a full k-tree. Otherwise, the complexity of the exact optimization algorithm is likely to be dominated by the complexity of an embedding algorithm.

A similar idea of combining states of components of a k-terminal graph according to its parse tree has been expressed by Wimer *et al.* [84] who list a score of families of

k-terminal graphs and several dozens of problems to which their methodology applies.

Although the approach of [5] was the first attempt to describe efficient algorithms on partial k-trees by a common paradigm, it did not address the question of mechanical derivation of an efficient algorithm solving a difficult problem on those graphs from the problem description. It took several more years for some of those problems to be identified.

The important results of Arnborg *et al.* [3] have been mentioned in the preceding section. The efficient algorithm solving a given *EMSOL* problem is constructed as a tree automaton following the formal description of the corresponding property.

Borie *et al.* [18] describe an algorithm design paradigm based on their definition of k-terminal, recursive family of graphs and of regular properties. Following the decomposition tree of the graph in the problem's instance and using the "states" indicated by the homomorphism classes (by definition, there is only a finite number of those), the dynamic programming technique is used to compute a solution to the problem in linear time.

6 Graph minors and existence of polynomial time algorithms

Major progress has been made possible by the results of Robertson and Seymour [67]. Their results on minor containment gave rise to a new non-constructive tools for establishing polynomial-time solvability [67] and a new interest in *forbidden substructures* characterization of classes of graphs [4], [33].

A graph H is a *minor* of a graph G if it can be obtained from a subgraph of G by contracting edges (*contracting* an edge introduces a new vertex replacing the two end vertices of the contracted edge and inheriting their adjacencies). Robertson and Seymour proved that every class of graphs closed under minor-taking has a finite number of *minimal forbidden minors* (graphs not in the class with all minors belonging to the class). Because every such class of graphs has constantly bounded tree-width, the membership of a graph in the class can be decided in time growing at most with the cube of the graph size, but with astronomical multiplicative constants. Similarly, many problems are now known to be decidable in low-degree polynomial time, based on the knowledge of the finite set of forbidden minors for a given class of graphs. However, but for a very few exceptions, there is no indication of how those graphs can be efficiently found, and even if they are known, the complexity of solution algorithms exhibit multiplicative constants of astronomical magnitude.

The class of graphs with path-width 2 has been characterized in [33] by 110 minimal forbidden minors. The class of partial 3-trees has a small set of minimal forbidden minors characterizing it [4]. The completeness of this set was proved using the knowledge of a small complete set of confluent reduction rules for this class of graphs. For higher values of k, this approach will not yield results as long as such rules are not known.

7 Parallel computation

Recent research on parallel algorithms shows that trees are amenable to the combination of the dynamic programming techniques (*pruning* of tree leaves) and the standard parallel techniques of *contraction* of long branches resulting in efficient parallel algorithms [58]. This discovery seems to generalize to graphs of tree-like structure, prime example of which are the partial k-trees. Bodlaender [15] uses it arguing the existence of poly-log algorithms for partial k-trees. Very recently, first efficient parallel algorithms for chordal graphs have been designed (Chandrasekhran and Iyengar [22], Naor *et al.* [63], Klein [48]). Chandrasekhran and Hedetniemi [20] describe an efficient parallel algorithm for the partial k-tree embedding problem.

8 Bibliography

References

[1] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability – a survey, *BIT 25* (1985), 2-23;

[2] S. Arnborg, D.G. Corneil, and A. Proskurowski, Complexity of finding embeddings in k-trees, *SIAM Journal of Algebraic and Discrete Methods 8* (1987), 277-284;

[3] S. Arnborg, J. Lagergren, and D. Seese, Problems easy for decomposable graphs, *Proceedings of ICALP 88*, Springer–Verlag Lecture Notes in Computer Science 317 (1988), 38-51;

[4] S. Arnborg, A. Proskurowski, and D.G. Corneil, Forbidden minors characterization of partial 3-trees, *UO-CIS-TR-86-07*, University of Oregon (1986), to appear in *Discrete Mathematics (1989)*;

[5] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial k-trees, *TRITA-NA-8404*, The Royal Institute of Technology (1984), *Discrete Applied Mathematics 23* (1989), 11-24;

[6] S. Arnborg and A. Proskurowski, Recognition of partial k-trees, *Proceedings of the 16th South–Eastern International Conference on Combinatorics, Graph Theory and Computing*, Utilitas Mathematica, Winnipeg, *Congressus Numerantium 47* (1985), 69-75;

[7] S. Arnborg and A. Proskurowski, Problems on graphs with bounded decomposability, *Bull. EATCS*(1985);

[8] S. Arnborg and A. Proskurowski, Characterization and recognition of partial 3-trees, *SIAM Journal of Algebraic and Discrete Methods 7* (1986), 305-314;

[9] B. Baker, Approximation algorithms for NP-complete problems on planar graphs, *Proceedings FOCS 24* (1983), 105-118;

[10] M. Bauderon and B. Courcelle, Graph expressions and graph rewritings, *Mathematical Systems Theory 20* (1987), 83-127;

[11] M.W. Bern, E.L. Lawler, and A.L. Wong, Linear Time Computation of Optimal Subgraphs of Decomposable Graphs, *Journal of Algorithms 8* (1987), 216-235;

[12] L.W. Bineke and R.E. Pippert, Properties and characterizations of k-trees, *Mathematica 18* (1971), 141-151;

[13] H.L. Bodlaender, Classes of graphs with bounded tree-width, *RUU-CS-86-22* (1986);

[14] H.L. Bodlaender, Planar graphs with bounded tree-width, *Bull. EATCS* (1988);

[15] H.L. Bodlaender, NC-algorithms for graphs with small tree-width, *Proceedings of the Workshop on Graph-Theoretic Concepts in Computer Science WG-88*, Springer–Verlag Lecture Notes in Computer Science 344 (1988), 1-10;

[16] H.L. Bodlaender, Polynomial algorithms for graph isomorphism and chromatic index on partial k-trees, *Proceedings of the Scandinavian Workshop on Algorithm Theory*, Springer–Verlag, Lecture Notes in Computer Science 318 (1988), 223-232;

[17] H.L. Bodlaender, Dynamic programming on graphs with bounded tree-width, *RUU-CS-87-22*, *Proceedings of ICALP'88*, Springer–Verlag Lecture Notes in Computer Science 317 (1988), 105-118;

[18] R.B. Borie, R.G. Parker, and C.A. Tovey, Automatic generation of linear algorithms from predicate calculus descriptions of problems on recursively constructed graph families, manuscript (July 1988);

[19] H. Buer and R.H. Möhring, A fast algorithm for decomposition of graphs and posets, *Math. Oper. Res. 8* (1983), 170-184;

[20] N. Chandrasekharan and S.T. Hedetniemi, Fast parallel algorithms for tree decomposing and parsing partial k-trees, *Proceedings of 26 Annual Allerton Conference in Communications, Control, and Computing* (1988),;

[21] N. Chandrasekharan, S.T. Hedetniemi, and T.V. Wimer, A method for obtaining difference equations for the number of vertex subsets having a given property in restricted k-terminal families of graphs, manuscript (October 1988);

[22] N. Chandrasekharan and S.S. Iyengar, NC algorithms fr recognizing chordal graphs and k-trees, *IEEE Trans. on Computers 37* (1988), 1170-1183;

[23] D.G. Corneil, H. Lerchs, and L. Stewart Burlingham, Complement reducible graphs, *Discrete Applied Mathematics 3* (1981), 163-174;

[24] B. Courcelle, Some applications of logic of universal algebra, and of category theory to the theory of graph transformations, *Bull. EATCS* (1988), 161-213;

[25] B. Courcelle, Recognizabilty and second-order definabilty for sets of finite graphs, *I-8634*, Université de Bordeaux, (1987);

[26] B. Courcelle, The monadic second order logic of graphs I: recognizable sets of finite graphs, *I-8837*, Université de Bordeaux (1988);

[27] B. Courcelle, The monadic second order logic of graphs III: tree-width, forbidden minors and complexity issues, *I-8852*, Université de Bordeaux (1988);

[28] W.M. Cunningham, Decomposition of directed graphs, *SIAM Journal of Algebraic and Discrete Methods 3* (1982), 214-228;

[29] W.M. Cunningham and J. Edmonds, A combinatorial decomposition theory, *Canadian J. Mathematics 32* (1980), 734-765;

[30] G.A. Dirac, On rigid circuit graphs, *Abh. Math. Sem. Univ. Hamburg 25* (1961), 71-76;

[31] J.E. Doner, Decidability of the weak second-order theory of two succesors, *Notices of the American Mathematical Society 12* (1966), 513;

[32] E.S. ElMallah, Decomposition and Embedding Problems for Restricted Networks, *PhD. Thesis*, University of Waterloo (1987);

[33] M.R.Fellows, N.G. Kinnersley, and M.A. Langston, Finite-basis theorems and a computational-integrated approach to obstruction set isolation, *Proceedings of Computers and Mathematics Conference* (1989);

[34] M.R.Fellows and M.A. Langston, Non-constructive advances in polynomial-time complexity, *Information Processing Letters 26* (1987), 157-162;

[35] M.R.Fellows and M.A. Langston, Non-constructive tools for proving polynomial-time decidability, *Journal of the ACM 35* (1988), 727-739;

[36] D.R. Fulkerson and O.A. Gross, Incidence matrices and interval graphs, *Pacific Journal of Mathematics 15* (1965), 835-855;

[37] F. Gavril, Algorithms on clique separable graphs, *Discrete Math. 19* (1977), 159-165.

[38] M.C. Golumbic, *Algorithmic graph theory and perfect graphs*, Academic Press (1980).

[39] A. Habel, Graph-theoretic properties compatible with graph derivations, *Proceedings of WG-88*, Springer–Verlag Lecture Notes in Computer Science 344 (1988), ;

[40] A. Habel, Hyperedge replacement: grammars and languages, Phd. Dissertation, Bremen 1989;

[41] A. Habel and H.J. Kreowski, May we introduce to you: hyperedge replacement, *Proceedings of the 3rd International Workshop on Graph Grammars and their Applications to Computer Science*, Springer-Verlag Lecture Notes in Computer Science 291 (1987), 15-26;

[42] M. Habib and R.H. Möhring, On some complexity properties of N-free posets and posets with bounded decomposition diameter, *Discrete Mathematics 63* (1987), 157-182;

[43] W. Hochberg and R. Reischuk, Decomposition graphs – a uniform approach for the design of fast sequential and parallel algorithms on graphs, manuscript (1989);

[44] D.S.Johnson, The NP-Completness column: an ongoing guide, *Journal of Algorithms 6* (1985), 434-451;

[45] D. Johnson, The NP-completeness column: an ongoing guide; *Journal of Algorithms 8* (1987), 285-303;

[46] Y.Kajitani, A.Ishizuka, and S.Ueno, A Characterization of the Partial k-tree in Terms of Certain Structures, *Proceedings of ISCAS '85* (1985), 1179-1182;

[47] M. Klawe, D.G. Corneil, and A. Proskurowski, Isomorphism testing in hook-up graphs, *SIAM Journal of Algebraic and Discrete Methods 3* (1982), 260-274;

[48] P.N. Klein, Efficient parallel algorithms for chordal graphs, *Proceedings of the 29th Symposium on FoCS* (1988), 150-161;

[49] C. Lauteman, Decomposition trees: structured graph representation and efficient algorithms, *Proceedings of CAAP'88*, Springer-Verlag Lecture Notes in Computer Science 299 (1988), 217-244;

[50] C. Lauteman, Efficient algorithms on context-free graph languages, *Proceedings of ICALP'88*, Springer-Verlag Lecture Notes in Computer Science 317 (1988), 362-378;

[51] T. Lengauer and E. Wanke, Efficient analysis of graph properties on context-free graph languages, *Proceedings of ICALP'88*, Springer-Verlag Lecture Notes in Computer Science 317 (1988), 379-393;

[52] A. Lingas and A. Proskurowski, Fast parallel algorithms for the subgraph homeomorphism and the subgraph isomorphism problems for classes of planar graphs, *UO-CIS-TR-87-12*, University of Oregon, to appear in *Theoretical Computer Science* (1989);

[53] A. Lingas and M.M. Syslo, A polynomial-time algorithm for subgraph isomorphism of two-connected series-parallel graphs, *Proceedings of ICALP'88*, Springer-Verlag Lecture Notes in Computer Science 317 (1988), 394-409;

[54] G.S. Lueker and K.S. Booth, A linear time algorithm for deciding interval graphs isomorphism, *Journal of the ACM 26* (1979), 183-195;

[55] S. Mahajan and J.G. Peters, Algorithms for regular properties in recursive graphs, *Proceedings of 25 Annual Allerton Conference in Communications, Control, and Computing* (1987), 14-23;

[56] J. Matousek and R. Thomas, On the complexity of finding iso- and other morphisms for partial k-trees, manuscript, (May 1988);

[57] J. Matousek and R. Thomas, Algorithms finding tree decompositions of graphs, manuscript (May 1988);

[58] G.L.Miller and J.H.Reif, Parallel Tree Contraction and its Applications, *Proceedings of the 26th FoCS* (1985), 478-489;

[59] B. Monien and I.H. Sudborough, Bandwidth constrained NP-complete problems, *Proceedings of the 13th STOC* (1981), 207-217;

[60] B. Monien, I.H. Sudborough, and M. Wiegers, Complexity results for graphs with treewidth O(log n), manuscript (1989);

[61] J.H. Muller and J. Spinrad, Incremental modular decomposition, *Journal of the ACM 36* (1989), 1-19;

[62] R.H. Möhring and F.J. Radermacher, Substitution decomposition for discrete structures and connections with combinatorial optimization, *Annals of Discrete Mathematics 19* (1984), 257-356;

[63] J. Naor, M. Naor, and A.A. Schäffer, Fast parallel algorithms for chordal graphs, *SIAM Journal on Computing* (1989);

[64] T.Politof, Δ–Y Reducible Graphs, Concordia University, Montreal, manuscript (1985);

[65] A.Proskurowski, Centers of 2-Trees, *Proceedings of the 2nd Combinatorial Conference France–Canada, Annals of Discrete Mathematics 9* (1980), 1-5;

[66] A.Proskurowski, Separating subgraphs in k-trees: cables and caterpillars, *Discrete Mathematics 49* (1984), 275-285;

[67] N. Robertson and P.D. Seymour, Graph Minors, (series of 23 papers in varying stages of editorial process, 1983-1988);

[68] D.J. Rose, On simple characterization of k-trees, *Discrete Mathematics 7* (1974), 317-322;

[69] A. Rosenthal and J.A. Pino, A generalized algorithm for centrality problems on trees, *Journal of the ACM 36* (1989), 349-361;

[70] G. Rozenberg and E. Welzl, Boundary NLC grammars: basic definitions, normal forms, and complexity, *Information and Control 69* (1986), 136-167;

[71] P. Scheffler, Linear time algorithms for NP-complete problems for partial k-trees, *R-MATH-03/87* (1987);

[72] P. Scheffler and D. Seese, Tree-width and polynomial-time solvable graph problems, manuscript (1986);

[73] D. Seese, The structure of the models of decidable monadic theories of graphs, *Journal of Pure and Applied Logic* (198x), ;

[74] D. Seese, Tree-partite graphs and the complexity of algorithms, *Proceedings of FCT-85*, Springer-Verlag Lecture Notes in Computer Science 199 (1985), 412-421;

[75] Y. Shibata, On the tree representation of chordal graphs, *Journal of Graph Theory 12* (1988), 421-428;

[76] A.O. Slisenko, Context-free grammars as a tool for describing polynomial subclasses of hard problems, *Information Processing Letters 14* (1982), 52-56;

[77] M.M. Syslo, NP-complete problems on some tree-structured graphs: a review, *Proceedings of WG-83*, Trauer Verlag (1984), 342-353;

[78] M.M. Syslo, A graph-theoretic approach to the jump number problem, in *Graphs and Orders*, I.Rival, Ed., Reidel Dodrecht (1985), 185-215.

[79] K. Takamizawa, T. Nishizeki, and N. Saito, Linear-time Computability of Combinatorial Problems on Series-parallel Graphs, *Journal of the ACM 29* (1982), 623-641;

[80] R.E. Tarjan, Decomposition by clique separators, *Discrete Mathematics 55* (1985), 221-232;

[81] J.W. Thatcher and J.B. Wright, Generalized finite automata theory with an application to a decision problem in second-order logic, *Mathematical Systems Theory 2* (1968), 57-81;

[82] T.V. Wimer, Linear algorithms on k-terminal graphs, PhD. Dissertation, Clemson University (August 1988);

[83] T.V. Wimer and S.T. Hedetniemi, k-terminal recursive families of graphs, *Proceedings of 25th Anual Conference on Graph Theory*, Utilitas Mathematica, Winnipeg, *Congressus Numerantium 63* (1988), 161-176;

[84] T.V. Wimer, S.T. Hedetniemi, and R. Laskar, A methodology for constructing linear graph algorithms, Clemson University, *TR-85-SEP-11*, (September 1985);

[85] P. Winter, Steiner problem in networks: a survey, *Networks 17* (1987), 129-167;