BIT 25 (1985), 2–23

INVITED PAPER.

EFFICIENT ALGORITHMS FOR COMBINATORIAL PROBLEMS ON GRAPHS WITH BOUNDED DECOMPOSABILITY – A SURVEY

STEFAN ARNBORG

Department of Numerical Analysis and Computing Science, The Royal Institute of Technology, S-10044 Stockholm, Sweden

1. Introduction.

In this survey I will give an overview of recent developments in methods for solving combinatorially difficult, i.e., NP-hard, problems defined on simple, labelled or unlabelled, graphs or hypergraphs. Special instances of these methods are repeatedly being invented in applications. They can be characterized as *table-based reduction methods*, because they work by successively eliminating the vertices of the problem graph, while building tables with the information about the eliminated part of the graph required to solve the problem at hand. Bertele and Brioschi [9] give a full account of the state-of-the-art in 1972 of these methods applied to non-serial optimization problems.

A preliminary taste of the family of algorithms I consider is given by the most well-known of all graph algorithms: the series-parallel reduction method for computing the resistance between two terminals of a network of resistors. It was invented by Ohm (1787–1854) and is contained in most high-school physics curricula. However, since resistance computation is not combinatorially difficult (the resistance is easily obtained, via Kirchhoff's laws, from the solution to a linear system of equations), I will illustrate the method on a different problem, that of computing the probability that a connection exists between two terminals of a series-parallel network of communication links that are unreliable and fail independently of each other, with given probabilities. This problem is NP-hard for arbitrary graphs, see Garey and Johnson [17, problem ND20], basically because all states of the set of communication links must be considered. On a series-parallel network, however, a simple reduction method will work.

The method successively eliminates non-terminal nodes which are adjacent to at most two links, by a local transformation of the network. The elimination operation can be defined as in Figure 1.1., where the quantities p_i labelling edges are the probabilities that the corresponding links work correctly. If a link of the

Received November 1, 1984.

triangle involved ((a, b, c) in Figure 1.1) is missing, the corresponding probability is set to zero in the computation of p'_1. The dashed area represents the part of the network not involved (and unchanged by the transformation).

When these transformations are applied to a connected two-terminal series-parallel network they will ultimately remove all non-terminal nodes and leave the terminals with a link between them which is labelled with a probability that is the answer to the problem. With suitable data structures, see Wald and Colbourn [2], the worst-case running time is $O(n)$ for an n-node network.

It is important to understand how the series-parallel reduction method can be interpreted as a decomposition method [1, Ch. 2]: Each link of the intermediate network can be seen as representing a part of the original network, a branch, which is connected to the remainder only through the endpoints of the intermediate link. The reliability of the intermediate link is all the information about the branch required later during the algorithm execution, and it is only by coincidence that this information is of the same kind as the input data for the problem. The node elimination operation can now be seen as an operation that

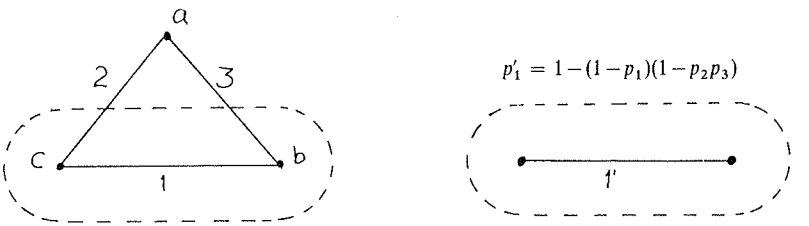

Fig. 1.1. Reduction of series-parallel communication network.

combines three branches into one and computes the information required for the new branch (Figure 1.2). A path through the new branch B'_1 is either a path through B_3 and a path through B_2, or a path through B_1, and the link sets of the branches B_1, B_2 and B_3 are disjoint, from which the computation rule for p'_1 (Figure 1.1) follows.

Fig. 1.2. Combining branches of series-parallel network.

2. Outline of paper.

This paper is intended to be useful for application specialists encountering computationally difficult problems and for computer scientists not specialized in this area. It also contains preliminary results on complement decomposable graphs. The reader is assumed to be familiar with algorithms and their analysis, see, e.g., Aho, Hopcroft and Ullman [1]. Results will be stated as theorems or as propositions depending on how new, significant and deep I consider them. No long proofs will be duplicated here, but a reference is always given. Many facts will just be sprinkled into the text with a reference if I consider them simple or well-known, or if they are supplied as background information.

In section 3, I give graph-theoretic definitions and some basic properties of graphs related to elimination processes, required for a comprehensive treatment of table-based reduction algorithms. Section 4 explains the table-based reduction method and exemplifies it with algorithms for some independent set problems. The problem of finding suitable elimination orderings, either with rewrite rule techniques or with dynamic programming methods, as well as the complexity of the optimization of elimination ordering, is treated in section 5. The basic method gives efficient algorithms for graphs which can be decomposed by a set of small separators. Graphs without small separators can still be treated for certain problems when they are clique decomposable. Results for such graphs are reviewed in section 6, as well as a new decomposition method, applicable to graphs with bounded complement decomposability. The appendix contains a full list of computational problems referred to in the text.

3. Graph-theoretic definitions and k-decomposable graphs.

A *simple loopless unlabelled graph* $G = (V, E)$ is a set V of *vertices* and a set E of *edges*, each edge being a 2-element subset of V. Throughout the paper, G, V and E, possibly with sub- or superscripts, will denote such a graph, its vertex set and its edge set. The set operation $\cup$ on graphs is defined componentwise. Vertices v and w are *adjacent iff* $\{v, w\} \in E$. Edge e is *incident* to vertex v *iff* $v \in e$. $\Gamma(v)$, the *neighborhood of* v, is the set of vertices adjacent to v. The neighborhood of a vertex set W is the set $\Gamma(W) = \cup_{w \in W} \Gamma(w)$. The *degree* of vertex v is the size of its neighborhood, $\deg(v) = |\Gamma(v)|$, and $\Delta(G) = \max_{v \in V} \deg(v)$. An *edge-labelling* of G is a function from its edges to some

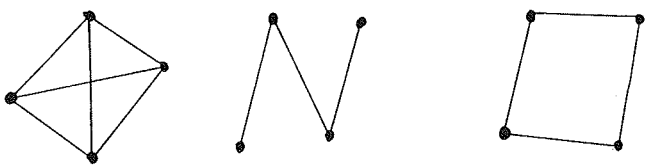

Fig. 3.1. K_4, P_4 and C_4.

domain, and a *vertex-labelling* is analogously defined. An *edge(vertex)-labelled graph* is a graph with an edge(vertex) labelling. A *complete* graph on vertex set V, $c(V)$, is a completely connected graph, i.e., if it has vertices v and w, then it has the edge $\{v, w\}$. K_n, P_n and C_n denote, respectively, the complete graph, the path and the simple cycle on n vertices, see Figure 3.1.

The *subgraph of G induced by* vertex set W is the graph $G(W) = (W, E')$, where $E' = \{e \in E : e \subset W\}$. A *partial graph* of $G = (V, E)$ is a graph $G' = (V, E')$, where $E' \subset E$. A *subgraph* of G is a partial graph of an induced subgraph of G. A *clique* of G is a (not necessarily maximal) vertex set W such that $G(W)$ is complete. The *complement* $\bar{G}$ of $G = (V, E)$ is the graph (V, E') such that $E \cap E' = \phi$ and $G \cup \bar{G}$ is complete.

The transitive reflexive closure of the adjacency relation is an equivalence relation and the blocks of the corresponding partition (equivalence classes) are the *connected components* of the graph. A graph with one connected component is *connected*, a graph with more than one component is *disconnected*. Vertex set S of G is a *separator* if $G(V-S)$ is disconnected. S is a *minimal separator* if in addition $G(V-S')$ is connected whenever S' is a proper subset of S (i.e., $S' \subset S$ and $S' \neq S$).

G is *chordal* iff it does not have an induced subgraph C_n with $n > 3$ (i.e., if every cycle has a chord, see Rose [22]). It is *k-chordal* if in addition it does not have a minimal separator of size greater than k. A graph $G = (V, E)$ is a *k-tree* [8, 23] if

(i) either G is the complete graph on k vertices,

(ii) or G has a vertex v of degree k, such that $\Gamma(v)$ is a clique of G and $G(V-\{v\})$ is a k-tree.

A vertex of a k-tree satisfying the condition in (ii) is a *simplicial vertex*. A vertex is always simplicial if it has degree k and its neighborhood is completely connected [22]. The class of k-chordal graphs is equal to the class of induced subgraphs of k-trees. Therefore, chordal graphs and k-trees are easily recognized by a procedure which successively deletes vertices with completely connected neighborhoods [22]. A graph G is *k-decomposable* if either of (i), (ii) holds:

(i) G has at most $k+1$ vertices

(ii) G has a separator S, $|S| \leqq k$, such that the components of $G(V-S)$ are $S_1, \ldots, S_n$ and all graphs $G(S_i \cup S) \cup c(S)$, $i = 1, \ldots, n$, are k-decomposable.

The graphs $G(S_i \cup S) \cup c(S)$ mentioned in the definition are obtained as shown in Figure 3.2. The component graphs are obtained by taking a "part" of the graph and filling in all edges between vertices of the separator.

An *elimination ordering* π is an ordering of the vertices of a graph. The *fill-in* caused by the ordering π, F_π, of $G = (V, E)$ is a set of edges computed as follows [22]:

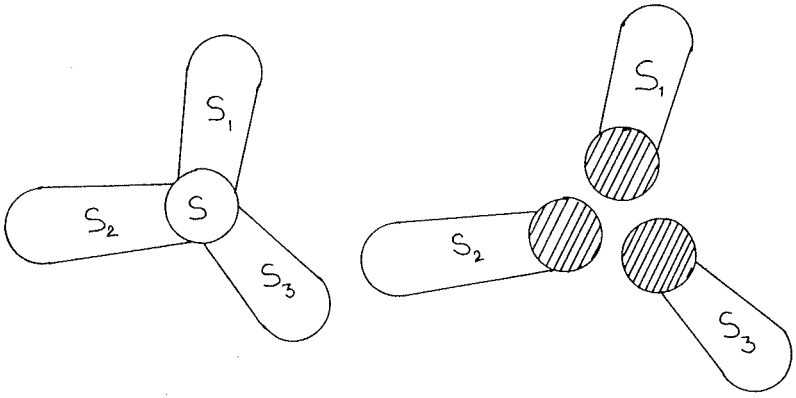

Fig. 3.2. A decomposable graph with its component graphs.

$F := \phi$;
Examine vertices in order π:
For each vertex v, find $C_\pi(v)$ which is the set of unexamined vertices adjacent to v in $G' = (V, E \cup F)$;
Add all edges in $\bar{G}'(C_\pi(v))$ to F;
F_π is the value of F when all vertices have been examined.

$G^\pi = (V, E \subset F_\pi)$ is always chordal. $C_\pi(v)$ is a separator of G and of G^π if some w is either a π-successor of v with $w \notin C_\pi(v)$, or a π-predecessor of v with $C_\pi(w) \subset C_\pi(v)$ [22]. The fill-in represents dependencies introduced during a variable (vertex) elimination process; e.g., if we have $x+z = a$ and $y-z = b$, represented by $(\{x, y, z\}, \{\{x,z\}, \{y, z\}\})$ we can eliminate z and get $x+y = a+b$, represented by $(\{x, y\}, \{\{x, y\}\})$. A dependency between x and y was introduced by elimination of z. A graph G has *dimension k with respect to π* if $\max_{v \in V}|C_\pi(v)| = k$. A graph G has *dimension k* if $\min_\pi \max_{v \in V}|C_\pi(v)| = k$, cf. [9]. A *series-parallel graph* is a graph of dimension at most 2.

A *simple, unlabelled hypergraph* $H = (V, E)$ consists of a set V of vertices and a set $E \subset \mathcal{P}(V)$ of *hyperedges* (i.e., each hyperedge is a subset of the vertices). The *clique representation* of a hypergraph $H = (V, E)$, is a simple graph $G = (V, E')$, obtained by replacing each hyperedge e by all edges of the complete graph on vertex set e, i.e.,

$$E' = \bigcup_{h \in E} \bigcup_{v,w \in h} \{v, w\},$$

or, alternatively,

$$G = \bigcup_{h \in E} c(h).$$

The clique representation is unique but several hypergraphs can have the same clique representation. All graph concepts defined above carry over to hypergraphs via the adjacency relation (the relation of having a common edge) or the clique representation.

The concepts k-tree, k-chordal, k-decomposable and dimension were introduced in various investigations rather independently of each other. They are, however, intimately related:

THEOREM 3.1: *The following conditions on a graph G are equivalent:*

(i) G is k-decomposable,

(ii) G is a partial graph of a k-chordal graph,

(iii) G is a partial graph of a k-tree,

(iv) G has dimension at most k.

PROOF: The equivalences (i) $\Leftrightarrow$ (iii) and (ii) $\Leftrightarrow$ (iii) are proved by Arnborg and Proskurowski [5] and Arnborg, Corneil and Proskurowski [4], respectively. An elimination ordering leads to a decomposition by the seperators $C_\pi(v)$, hence (iv) $\Rightarrow$ (i). (ii) $\Rightarrow$ (iv) follows from the observation that the dimension of a graph is non-decreasing as its edge-set increases, and that a chordal graph has an elimination ordering without fill-in [22]. ■

It follows from Theorem 3.1 that the class of series-parallel graphs is the class of partial graphs of 2-trees. Partial 2-trees are characterized by the forbidden homeomorphic subgraph K_4, but this simple characterization cannot be generalized to arbitrary partial k-trees, cf. [14, 29].

4. The table-based reduction method.

Many important combinatorial problems on graphs concern the existence of certain sets. An *independent set* of a graph is a vertex set I such that no two vertices in I are adjacent. The standard versions of combinatorial problems are *decision problems*, e.g., the problem INDEPENDENT SET is: "Given G and an integer k, does G have an independent set of at least k vertices?". This problem is NP-complete [17]. In applications one is often interested in the corresponding *optimization problem*, in this case MAX IND SET: "Given graph G, what is the maximum size of its independent sets?" We may also be interested in *counting problems* such as finding the number of maximum size independent sets or independent sets larger than a prescribed number (many problems in discrete probabilities can be seen as weighted counting problems, like the network reliability problem described in the introduction), and *realization problems* with solutions actually exhibiting one or all maximum size independent sets.

We will consider table-based reduction algorithms which successively eliminate vertices of the problem graph, and at the same time build information in tables about the eliminated part of the graph, so that when vertex v is eliminated and

the elimination ordering is π, a table is created for vertex set $C_\pi(v)$. This table has contents depending only on the subgraph of the problem induced by $C_\pi(v)$ and the connected component of $G(V - C_\pi(v))$ which contains v, the *branch* $B_\pi(v)$ of vertex v under elimination ordering π. The table for $C_\pi(v)$ is created from tables associated with vertex sets containing v but no eliminated vertex, and possibly problem data in the form of a vertex or edge labelling on the subgraph $G(C_\pi(v) \cup \{v\})$.

The answer to an optimization or decision problem for a connected graph can be computed from the last table produced. Counting problems are solved by having extra information in the tables, and realization problems are solved through a scan of created tables in reverse order.

I will now describe algorithms, in the form of table update rules, for independent set problems. In the optimization problem each row of a table will contain a subset of the vertex set of the table and an integer. Row (i, n) in a table for vertex set $C_\pi(v)$ says that $G(B_\pi(v) - \Gamma(i))$ has a maximum size independent set with n vertices. Initially, tables are created for edges of G, so that edge $\{v, w\}$ has table

i	n
ϕ	0
$\{v\}$	0
$\{w\}$	0

The update operation when vertex v is eliminated is as follows, where $T(v)$ is the set of tables with vertex set containing v but no eliminated vertex:

(i) Construct intermediate tables T_{in} and T_{out} for the two cases where v is in or not in the independent set: A row (i, n), $i \subset C_\pi(v)$ of T_{in}, is produced if there is a row (i_j, n_j) for each table T_j in $T(v)$ on a vertex set e_j such that

$$i_j = (i \cap e_j) \cup \{v\}$$

$$n = 1 + \sum_j n_j.$$

Similarly, a row (i, n) of T_{out} is produced if

$$i_j = i \cap e_j$$

$$n = \sum_j n_j.$$

(ii) Merge tables T_{in} and T_{out} to table T. If row (i, n_{in}) is in T_{in} and row (i, n_{out}) in T_{out}, keep row $(i, \max(n_{in}, n_{out}))$ in T, and if index i of row (i, n) occurs only in one of the tables, keep row (i, n) in T.

When the last vertex has been eliminated, the answer to the optimization problem is the sum of the values in created tables associated with the empty vertex set. One such table is produced for every connected component of the problem graph, and each such table has only one row (ϕ, n).

The correctness of the method follows from an optimality principle: if $S \subset V$ and $G(V-S)$ has components $\{S_j\}_{j\in J}$, then the size of a maximum independent set is the maximum over $i \subset S$ of the sum over $j \in J$ of maximum independent set sizes of $G(S_j - \Gamma(i))$, plus $|i|$.

One counting problem version of MAX IND SET is the problem of finding the number of maximum independent sets. This problem can be solved by adding another number c to each row of each table, as follows:

(i) For the initial tables, $c = 1$.

(ii) For tables T_{in} and T_{out} constructed under step (i) of the optimization algorithm, c equals the product of the c_j values of rows in tables T_j, (i_j, n_j, c_j), from which the row of table T_{in} or T_{out} was constructed.

(iii) When T is obtained by merging T_{in} and T_{out}, c will be taken from the same row as n, but if $n_{in} = n_{out}$, c will be the sum of corresponding c values ($c_{in} + c_{out}$ if the rows are (i, n, c_{in}) and (i, n, c_{out})).

The answer to the counting problem will be the product of c values for tables associated with empty vertex sets.

The realization version of MAX IND SET, that of actually constructing one or all maximum independent sets, is solved by the following modification. Let T_v be the table produced when vertex v was eliminated. Add an item h to each row of each table T_v which indicates how the row was obtained: from T_{in}, T_{out} or, when the function values were equal, from both. Thus $h \in \{\text{in}, \text{out}, \text{both}\}$. A solution is obtained non-deterministically by examining the tables in the reversed elimination ordering:

I is a vertex set variable, initially empty;
for each vertex v, in reversed elimination ordering **do**
  find row (i, n, h) of T_v, such that $I \cap C_\pi(v) = i$;
  if $h =$ in **then** $I := I \cup \{v\}$ **else**
  if $h =$ both **then doordonot** $I := I \cup \{v\}$;
I is a maximum independent set.

Non-determinism is expressed by the operator **doordonot**. Depending on how it is implemented (random choice or backtrack), one or all optimal solutions are produced. The algorithm can also be used for numbering the optimal solutions.

Three types of optimization of the algorithms are quite often applicable. The first consists of eliminating a table whose associated vertex set is a subset of that

of another table, i.e., if T_1 and T_2 are associated with vertex sets e_1 and e_2, and $e_1 \subset e_2$, then a table T_2' on e_2 replaces T_1 and T_2. A row of T_2' is (i, n) if there are rows (i_1, n_1) and (i_2, n_2) in T_1 and T_2, respectively, such that

$$i_2 \cap e_1 = i_1$$

$$n = n_1 + n_2.$$

Another optimization consists in eliminating rows that are "dominated": if table T contains rows (i_1, n_1) and (i_2, n_2), $i_1 \subset i_2$ and $|i_1| + n_1 > |i_2| + n_2$, then row (i_2, n_2) can be removed because it cannot be used in an optimal solution. If only the optimization problem is wanted or if only one arbitrary realization is requested, then (i_2, n_2) can be removed also if $|i_1| + n_1 = |i_2| + n_2$ because some optimal solution does not use row (i_2, n_2).

A third technique, known as constraint propagation, has been reported by Walz [30] to eliminate all unwanted rows (i.e., rows that are not used in any solution) at an early stage in some scene labeling applications of CONJUNCTIVE QUERY. The technique consists of comparing tables with overlapping vertex sets (say T_i, T_j with vertex sets e_i and e_j) and deleting row (i_i, n_i) in T_i if there is no row (i_j, n_j) in T_j with $i_i \cap e_j = i_j \cap e_i$, cf. Freuder [15].

When the available optimizations have been exploited, we may still not be able to eliminate all vertices in reasonable time. We must then revert to standard techniques of finding a reasonable solution even if it is not optimal. One such technique consists in looking for locally optimal solutions, and if several locally optimal solutions are produced from starting values obtained with Monte-Carlo techniques, there is hope that the best solution is close to the global optimum. Unfortunately, it is in general not possible to analyze this approximation method in quantitative terms and although quite many applications use such methods, few empirical results have been published – except as rather partial and incomplete justifications of an application. A recent popularization of the method, using a fascinating analogy with statistical mechanics to justify the heuristics applied, was reported by Kirkpatrick, Gelatt and Vecchi [21]. For counting problems, Monte-Carlo simulation is usually a reasonable technique. Rosenthal [26] has shown that such a simulation gives better results if it is preceded by elimination of vertices of low degree.

MAX CLIQUE, the complementary problem of MAX IND SET, can somewhat surprisingly be solved by essentially the same method, although the information in tables will be rather different. Suppose K is a largest clique in G. When the first vertex eliminated in K is v, K must be contained in $C_\pi(v) \cup \{v\}$. The proper way to obtain the size of the largest clique is thus to let the table associated with $C_\pi(v)$ contain just the size of the largest clique in $G(C_\pi(v) \cup B_\pi(v))$. This is the maximum of the values of all tables associated with a vertex set containing v, and the size of the largest clique in $G(C_\pi(v) \cup \{v\})$. Counting and

realization problems are solved quite analogously to how it is done for maximum independent sets.

These table-based reduction algorithms have a wide applicability. Among reported application algorithms using the method can be mentioned NONSERIAL OPTIMIZATION [9], NETWORK RELIABILITY [24], CONJUNCTIVE QUERY [3], CNF SATISFIABILITY [13] and SYSTEM RELIABILITY [2]. Since these algorithms were independently designed, their descriptions do not always emphasize the simple elimination principle on which they are based. They concentrate more on lower-level details like application-dependent optimizations and efficient data structures (problems that are, of course, important enough). Algorithms for a number of problems have been constructed [6] for the purpose of showing the wide applicability of the method. These problems are VERTEX COVER, INDEPENDENT SET, DOMINATING SET, GRAPH K-COLORABILITY, HAMILTONIAN CIRCUIT and NETWORK RELIABILITY. Algorithms designed for series-parallel graphs which seem straightforwardly extendable to graphs with bounded dimension are reported for finding a Steiner tree [29], the resistance of an electrical network with nonlinear elements [14] and for computing several more complex measures of network reliability [26] and performance [10]. Several other examples can be found in the NP-completeness column of Johnson [20].

Generally speaking, combinatorial optimization problems arising in applications are often slightly modified versions of the "pure" problems analyzed in the scientific literature. It seems as if table-based reduction algorithms can be used for a large majority of these modified problems, and the applicability of the method depends mainly on whether or not the application tends to generate graphs and hypergraphs with small dimension. The application oriented results for table-based reduction methods clearly show that there are quite many such applications, but there are also examples where graphs are of high dimension. One such example is a square grid [9, 25]. In some applications the methods are claimed feasible even on graphs with higher dimension than suggested by a worst-case estimate. Solid support has been given to some of these claims. As an example, the Davis-Putnam procedure for CNF SATISFIABILITY was shown polynomial on the average for some reasonable instance distributions [20, June 84]. When the problem is restricted to at most two literals per clause, the fill-in will not come in full force, so the procedure becomes in fact polynomial in the worst case without a dimension restriction [17]. The principle carries over to several problems which have a natural formulation on hypergraphs and for which a very efficient elimination algorithm can be designed when the problem is restricted to hypergraphs with at most two vertices per hyperedge (i.e., essentially ordinary graphs). This technique has been explored by Aspvall [7].

One may ask if all NP-hard problems defined on graphs can be solved in linear time on k-decomposable graphs. The answer is probably no, because some problems on graphs are NP-complete even on trees, a subclass of 1-decomposable

graphs. However, the only examples I know of are BANDWIDTH, which was shown NP-complete even on trees with bounded degree [16], and a quantified version of CONJUNCTIVE QUERY [3], NP-complete on trees with only two quantifier alternations (the prefix is $\exists^*\forall\exists$). The CUTWIDTH problem is quite similar to BANDWIDTH but was recently shown by Yannakakis to have an $O(n \log n)$ algorithm for trees [33]. However, the "tables" are fairly complicated and the algorithm has not yet been extended even to series-parallel graphs. I do not know any natural problem which is polynomial on trees but NP-complete on series-parallel graphs. At present no useful characterisation exists for problems efficiently solvable with table-based reduction methods on k-decomposable graphs. Some results for series-parallel graphs were derived by Takamizawa, Nighizeki and Saito [27]:

THEOREM 4.1. *If a graph property Q is defined in terms of a finite set of forbidden, induced or homeomorphic, subgraphs, then the following problems can be solved in linear time when the problem graph $G = (V, E)$ is constrained to be series-parallel;*

(i) *Does G have property Q?*
(ii) *How large is the largest set $W \subset V$ such that $G(W)$ has property Q?*
(iii) *How large is the largest set $E' \subset E$ such that $G' = (V, E')$ has property Q?*

THEOREM 4.2. *For any graph P, the following problem (generalized matching) can be solved in linear time: Given a series-parallel graph G, what is the size of the largest family of disjoint vertex sets $\{C_i\}_{i=1}^k$ such that P is isomorphic to a partial graph of $G(C_i)$, all $i = 1, \ldots, k$?*

Another natural question is whether or not table-based reduction methods are better than alternative methods. Of course, since we do not even know if NP-hard problems have polynomial time algorithms (we only know that all or no NP-complete problems have polynomial time algorithms), it is not possible at present to get particularly strong optimality results. A preliminary, but quite interesting, result has been reported by Rosenthal [25]. It is based on the decision tree computation model, i.e. computations are modeled with a binary (or ternary) tree, where a node represents a comparison, its sons the outcomes of the comparison, and where each possible execution is represented by a path from the root to a leaf. The cost of the algorithm is the maximum length of such a path:

THEOREM 4.3. *For NON-SERIAL OPTIMIZATION restricted to the family of chordal hypergraphs, no comparison-based algorithm has lower cost than a table-based reduction method using a perfect elimination ordering.*

This result cannot necessarily be extended to those problems which are special cases of NON-SERIAL OPTIMIZATION, because in those cases the

function tables are restricted. Rosenthal also proves the optimality of table-based reduction methods on unrestricted hypergraphs, but with a restriction on the family of algorithms to "non-overlapping" comparison algorithms. Although this family seems natural, its definition is too technical for this survey and the interested reader is referred to [25].

5. Finding good vertex elimination orderings.

The cost of table-based reduction algorithms depends critically on the elimination ordering used. The cost to eliminate a vertex can usually (i.e., in all cases that have been investigated) be bounded by a function of the size of the largest separator used in the elimination process, the dimension k of the graph with respect to the elimination ordering. For an NP-hard problem, this bound is always at least exponential in k (otherwise we would have a polynomial time algorithm for an NP-hard problem, since a graph of n vertices has dimension less than n. This would be extremely unlikely and surprising). For many problems the elimination cost for vertex v also depends very much on the connectedness of $G(C_\pi(v) \cup \{v\})$. In particular, the elimination cost is often polynomial in the size of separators constrained to be cliques, a feature discussed further in section 6. However, the dimension of the graph with respect to the elimination ordering is a satisfactory objective function in most applications, and I will therefore survey some results on finding elimination orderings giving smallest dimension.

This is, by Theorem 3.1, the problem of finding the smallest k such that graph G can be completed to a k-tree, or, equivalently, a k-chordal graph. It is similar to the MIN FILL-IN problem, which asks for the smallest number of edges that must be added to G in order to make it chordal. The latter problem occurred in elimination order investigations for sparse linear systems of equations. We have:

THEOREM 5.1. (Yannakakis [32]) *MIN FILL-IN is NP-hard.*

THEOREM 5.2. (Arnborg, Corneil and Proskurowski [4]) *DIMENSION is NP-hard.*

The full proofs are not given here. As in most NP-completeness results, the main difficulty is to relate the given problem, by polynomial reducibility, to an already known NP-complete problem. Those problems will be OPTIMAL LINEAR ARRANGEMENT and CUTWIDTH. The constructions are shown in Figure 5.1. Graph $G = (V, E)$ is transformed into two graphs, G' and G'' as follows: G' has a group B_e of two vertices for each edge in G, a group B_v of $\Delta(G) - \deg_G(v)$ vertices and a singleton group A_v for each vertex v in G. The vertex of A_v is adjacent to each vertex of B_e if e is incident to v and to each vertex in $\bigcup B_v$, and the vertex groups $\cup_{v \in V} A_v$ and $\cup_{x \in E \cup V} B_x$ are cliques of G'.

G'' has B_e groups defined as for G', and two groups A_v and B_v with $\Delta(G)+1$ and $\Delta(G)-\deg(x)+1$ vertices, respectively, for each $v \in V$. Vertices x and y in G'' are adjacent iff either of (i) $x \in A_v$, $y \in B_e$, e incident to v in G; (ii) $x \in A_v$, $y \in B_v$; (iii) $x \in A_v$, $y \in A_w$; (iv) $x \in B_i$, $y \in B_j$; where i and j is any edge or vertex, is true.

The main contents of the proofs of Theorems 5.1 and 5.2 is to show that G has a linear arrangement with total edge-length at most k *iff* G' can be made chordal by addition of not more than $k+n^2(n-1)/2-2m$ edges, and that G has a linear arrangement with maximum cut at most k *iff* G'' can be completed, by addition of edges, to a k'-tree with $k' = (\Delta(G)+1)(n+1)+k-1$, where G has n vertices and m edges.

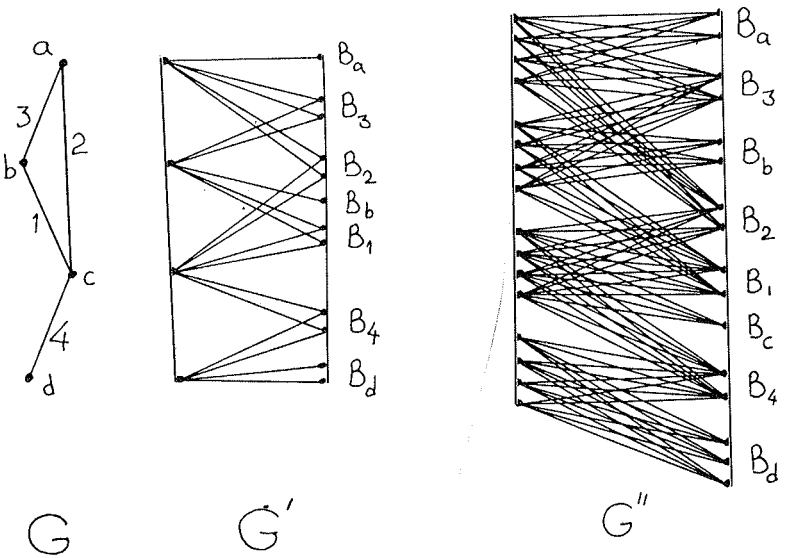

Fig. 5.1. Graph transformations for Theorems 5.1, 5.2.

Theorem 5.2 says that it is, in the worst case, infeasible to determine the dimension of a graph or to find an optimal elimination ordering. Fortunately (or perhaps unfortunately) the exponential dependence of the cost for our algorithms on the dimension of the problem graph means that we are mainly interested in finding the dimension when it is small, say up to 10 [6]. An $O(n^{k+2})$ algorithm for deciding if a graph has dimension k, and for finding an optimal elimination ordering in that case, is given in [4].

The algorithm starts by finding all separators of size at most k, and for each such separator S with associated components S_i of $G(V-S)$ the graphs $G(S_i \cup S) \cup c(S)$ are put into a table in increasing vertex set size order. The table is processed in this order and for each entry it is decided whether or not the graph can be composed from previously processed graphs decided to be k-decomposable. Since the series-parallel reduction method recognises a 2-decom-

posable graph in linear time whereas the general method would require time $O(n^4)$ for such a graph, there is room for improvement. Although the algorithm is the only, and therefore the best, polynomial algorithm for recognizing k-decomposable graphs, there are methods which will give faster execution on special graphs.

PROPOSITION 5.3. (Bertele and Brioschi [9]) *If G a clique separator S and $G(V-S)$ has the components S_i, then the dimension of G is the largest of the dimensions of the graphs $G(S \cup S_i)$.*

PROOF: If G has dimension k then every induced subgraph $G(S \cup S_i)$ has dimension at most k (see the computation rule for fill-in). If every graph $G(S \cup S_i)$ is k-decomposable, then, since $G(S \cup S_i) = G(S \cup S_i) \cup c(S)$, G is k-decomposable. The proposition then follows from Theorem 3.1. ■

This means that if S is a clique separator where $G(V-S)$ has components S_i, then an optimal elimination ordering can be found from optimal elimination orderings of graphs $G(S \cup S_i)$, i.e., Proposition 5.3 can be used to decompose the problem into smaller ones. Of course, one might expect it difficult to find clique separators, since both cliques and separators are difficult to find, unless they are small. Fortunately, a clique separator, if one exists, can be found in time $O(nm)$ for an n-vertex graph with m edges (Whitesides [31]). Proposition 5.3 also allows us to assume that a vertex of degree 0 or 1 is the first in an optimal elimination ordering. There are other cases where one can safely assume that a certain vertex can be eliminated first in an optimal elimination ordering:

PROPOSITION 5.4. (Bertele and Brioschi [9], Arnborg and Proskurowski [5]) *If $\bar{G}(\Gamma(v))$ has no two adjacent vertices of degree greater than 1 and if G is of dimension at least $deg(v)$, then v is the first vertex in some optimal elimination ordering.*

This Proposition allows us to eliminate at once vertices of degree 2. This is sufficient for finding an optimal elimination ordering for all 2-decomposable graphs, since a graph without vertices of degree at most 2 cannot be 2-decomposable (see also [29]).

It is very rewarding to regard the elimination process of a graph as a rewriting process, a concept thoroughly studied in computer algebra (see, e.g., Huet and Oppen [19]). A rewrite rule in this context deletes a vertex and its incident edges and fills in the fill edges in $\Gamma(v)$. The rules for degree 0, 1 and 2 are given in Figure 5.2 (i)–(iii). Proposition 5.4 also implies that the rule of Figure 5.2 (iv), the "triangle rule", is safe in the sense that application of the rule to a minimum degree vertex does not increase the dimension of the graph. The following Propositions give additional "safe rules":

PROPOSITION 5.5. (Arnborg and Proskurowski [5]) *If there are vertices* v_1, v_2 *such that* $S = \Gamma(v_1) = \Gamma(v_2)$, $\bar{G}(S)$ *has no vertex of degree greater than* 2 *and* G *is of dimension at least* $|S|$, *then* v_1 *and* v_2 *(in any order) start some optimal elimination ordering.*

For vertices of degree 3, this proposition gives the "buddy rule", Figure 5.2 (v).

PROPOSITION 5.6. (Arnborg and Proskurowski [5]) *If* G *has vertices* $v_1, \ldots, v_k$, $u_0, \ldots, u_k$, *such that* $\Gamma(v_i) = \{u_j | j \neq i\}$ *and* G *is of dimension at least* k, *then vertices* $v_1, \ldots, v_k$ *(in any order) start some optimal elimination ordering.*

For $k = 3$ this proposition gives the "cube rule", Figure 5.2 (vi). We have now a complete set of rewrite rules which will transform every 3-decomposable graph to the empty graph, at the same time giving an optimal elimination ordering:

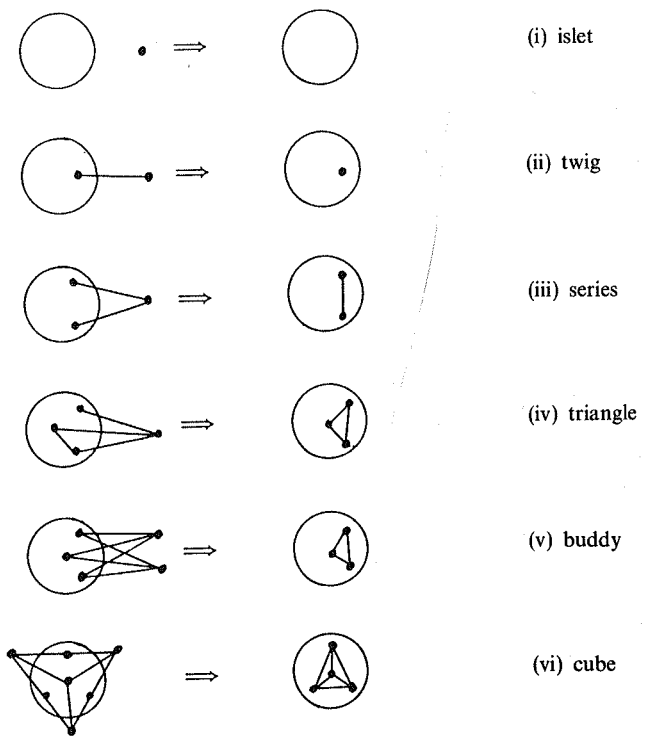

Fig. 5.2. Safe and complete rewrite rules for 3-decomposable graphs.

THEOREM 5.7. (Arnborg and Proskurowski [5]) *Rewrite rule sets from Figure* 5.2 *correspond to graph classes as follows*:

(*i*) – 0-*decomposable*
(*i*)–(*ii*) – 1-*decomposable*
(*i*)–(*iii*) – 2-*decomposable*
(*i*)–(*vi*) – 3-*decomposable*

in the sense that repeated application of rewrite rules from a set reduces a graph G to the empty graph iff G is a member of the corresponding class.

Theorem 5.7 indicates an algorithm for recognizing 3-decomposable graphs which operates in time $O(n^4)$ as opposed to $O(n^5)$ for the table-based method [4]. However, graphs which require the cube rule are extremely unusual, so in practice it operates in time less than $O(n^3)$ for randomly generated 3-decomposable graphs. An extension of theorem 5.7 to arbitrary k-decomposable graphs and even to 4-decomposable graphs would be valuable, but it seems difficult to establish the completeness of a set of rewrite rules in these cases. Monte-Carlo simulations have shown the set of "safe" rules described in [5] efficient in the sense that it is sufficient for recognizing "most" k-decomposable graphs for $k \leqq 7$.

6. Applications to dense graphs.

A k-decomposable graph with n vertices cannot, by Theorem 3.1, have more than nk edges, since it can be eliminated by removal of its vertices so that no more than k edges are removed with any vertex. Thus, k-decomposable graphs must be sparse if they have significantly more than k vertices. I will describe two cases where table-based reduction methods are applicable to dense graphs. The first case is where the problem graph contains many clique separators. As an example, consider the tables produced for the problem MAX IND SET in section 4. The index i of a table row in the table produced for $C_\pi(v)$ when v is eliminated is an independent set of $G(C_\pi(v))$, so if $C_\pi(v)$ is a clique of size m, the table can only have $m+1$ rows. Likewise, for MAX CLIQUE, it is easy to find the largest clique of $G(C_\pi(v) \cup \{v\})$ if $C_\pi(v)$ is a clique, in time polynomial in its size. Thus, for a graph and elimination ordering where all sets $C_\pi(v)$ are cliques, i.e., a chordal graph and a perfect elimination ordering, the algorithm will run in polynomial time for these two problems, as well as for the problems GRAPH K-COLORABILITY and CLIQUE COVER (Gavril [18]). Tarjan [28] has designed a decomposition algorithm which decomposes a graph by clique separators into atoms which are not clique separable. He notes that for the four problems above and for the MIN FILL-IN problem, the solution on the whole graph can be constructed in polynomial time from solutions to similar problems on the atoms, and that approximate solutions on the atoms can be used to construct approximate solutions on the whole graph. The technique is apparently not applicable to all problems. Particularly, MIN DOM SET has been shown NP-hard even when constrained to chordal graphs [11], [20, Jan. 1984]. Two key properties of a problem which make it feasible for clique-separable graphs seem to be: (i) the problem asks for a subset or labeling with a property such that there are only polynomially many on a complete graph; (ii) the problem has a "hereditary" property. I will only define this property with an example and a

nonexample: if I is an independent set of G, then $I \cap W$ is an independent set of $G(W)$, but if D is a dominating set of G then $D \cap W$ is not necessarily a dominating set of $G(W)$.

The other case of problems solvable on dense families of graphs is based on complementary problem pairs. Problems P and $\bar{P}$ are complementary if the solution to P on graph G directly gives the solution to $\bar{P}$ on the complementary graph $\bar{G}$. Such pairs are MAX IND SET – MAX CLIQUE and MIN GRAPH COLORABILITY – MIN CLIQUE COVER. There are surprisingly many problems where it is easy to find a complementary problem which, even if it looks quite artificial, can be solved efficiently with the table-based reduction method (important exceptions are most problems where an edge-labelling is part of the problem instance). All such problem pairs are, of course, easy on graphs which are complements of graphs with small dimension. One may ask if there is also a reasonable definition of a class of graphs such that a problem on a graph in the class can ve solved by solving the original problem on some parts of the graph and the complementary problem on complements of other parts of the graph. I will propose such a family, which can be considered a generalization of co-graphs or complement reducible graphs (Corneil, Lerchs and Stewart Burlingham [12]):

G is a co-graph *iff* either of (i)–(iii):

(i) G is a single vertex
(ii) G is disconnected and all components of G are co-graphs
(iii) $\bar{G}$ is disconnected and all components of $\bar{G}$ are co-graphs.

Problem P is easy for co-graphs if both P and $\bar{P}$ are such that the answer for a disconnected graph is easy, given the answers on the components. Since a graph and its complement cannot both be disconnected, the decomposition defined by (i)–(iii) is unique. Linear time algorithms for the two problem pairs above and also for finding the scattering number (a generalization of HAMILTONIAN CIRCUIT) when constrained to co-graphs are described in [12], where it is also shown that co-graphs are characterized by the forbidden subgraph P_4.

The main difficulty in generalising the co-graph class is the proper treatment of fill in the joints between complemented and non-complemented parts of the graph.

DEFINTIONS: A *shaded* graph is an edge-labelled graph with labels grey and black. An unshaded graph is represented by the shaded graph obtained by painting its edges black. The *complement* of a shaded graph is obtained by changing black edges to nonedges, nonedges to black, and keeping grey edges unchanged.

By *shading vertex set* S in a shaded graph G we mean the operation of changing all edges and non-edges in $G(S)$ to grey edges, obtaining graph $s(G, S)$.

A shaded graph G is *complement k-decomposable* or *co-k iff* either of (i)–(iii):

(i) G has $k+1$ or fewer vertices,
(ii) G has separator S (with respect to black and grey edges) such that $G(V-S)$ has components S_i, and all graphs $s(G(S \cup S_i), S)$ are complement k-decomposable,
(iii) $\bar{G}$ is complement k-decomposable.

An *s-subgraph* of a shaded graph G is an induced subgraph of G where possibly some grey edges have been deleted or painted black. We note that the class of co-k graphs is closed under complementation and taking s-subgraphs. The recognition problem for co-k graphs seems more complicated than that for k-decomposable graphs, and it is not obvious that the problem is in P for fixed but arbitrary k. The following simple analogue of Proposition 5.3 gives polynomial-time algorithms for recognizing and decomposing co-0 and co-1 graphs:

PROPOSITION 6.1: *If a shaded graph G has a shaded separator S, with associated components S_i, then G is complement k-decomposable iff all the graphs $G(S \cup S_i)$ are complement k-decomposable.*

PROOF: ($\rightarrow$): if G is co-k, then its s-subgraphs $G(S \cup S_i)$ are co-k.
($\leftarrow$): if all the graphs $G(S \cup S_i)$ are co-k, then, since $G(S \cup S_i) = s(G(s \cup S_i), S)$, G satisfies item (ii) of the definition of co-k. ■

7. Conclusions and open problems.

Table-based reduction methods can be used to solve efficiently most combinatorial problems defined on graphs and hypergraphs of bounded dimension, and many problems on clique separable graphs and complement decomposable graphs. The method is thus quite general. It compares in generality with methods of generating the convex hull of the domain of feasible sets and solving the optimization with linear programming (in which case NP-completeness of the combinatorial problem reflects itself in an exponential worst-case bound on the size of the LP problem in terms of the size of the original problem). There seem to be connections between these methods which deserve to be investigated. Another area that should be investigated is the recognition problem of complement k-decomposable graphs, and a characterization of problems solvable on such graphs.

A most promising line of research, however, is an investigation of the parallelism of table-based reduction methods, and design of a circuit architecture exploiting such parallelism and programmable for a large set of table constructions for various problems. It should be observed that, given an embedding of the

problem graph in a k-tree, all simplicial vertices can be processed in parallel, and the table construction for one elimination has also a great deal of parallelism in operations like maximization and summing. A processor based on such an architecture and realized in standard technology would be orders of magnitude better in cost/performance than a supercomputer, for a significant portion of optimization problems arising in applications. It could be attached to a personal workstation, attached as a resource in a high-speed local network, or embedded in an on-line control system.

Appendix.

For completeness, this appendix defines all computational problems mentioned in the paper. For problems where no reference is given, a complexity result for a similar problem can be found from [17].

MAX CLIQUE: "Given G, what is its maximum clique size?"

A *dominating set* of G is a vertex set D such that every vertex in $V-D$ is adjacent to a vertex in D.

MIN DOM SET: "Given G, what is its minimum dominating set size?"

A *k-coloring* of G is a vertex-labelling of G with domain $\{1, 2, \ldots, k\}$, such that no two adjacent vertices have the same label.

MIN GRAPH COLORABILITY: "Given G, what is the smallest k such that G has a k-coloring?"

MIN CLIQUE COVER: "Given $G = (V, E)$, what is the size of the smallest set of cliques in G whose union is V?"

NETWORK RELIABILITY: "Given G, where each edge e is labelled with a probability $p(e)$ between 0 and 1, what is the probability that the corresponding network is connected?"

This is a weighted counting problem. The probability is the sum over all subsets x of E defining a connected partial graph (V, x) of G, of $\prod_{e \in x} p(e) \prod_{e \in E-x}(1-p(e))$. A more general version of this problem asks for the probability that a designated vertex set W is contained in one connected component. In section 1 we solved this problem for $|W| = 2$ on a series-parallel graph with terminals W.

A *system event description* is a set of independent *atomic events*, each assigned a probability, and a set of *composite events*, each defined by a logical function of

its operands, each operand being an atomic or composite event, and finally a subset of the composite events, the *system events*. The dependency relation must be acyclic.

SYSTEM RELIABILITY [2]: "Given a system event description, what are the probabilities of the system events?"

An important family of problems not normally seen as problems on graphs can be modelled on hypergraphs and solved by vertex elimination when the clique representation of the hypergraph has low dimension:

CNF SATISFIABILITY: "Given an edge labelled hypergraph H, where an edge label is a set of subsets of the edge vertices, let a logical formula correspond to H as follows: There is a variable for each vertex, a clause for each member c of an edge label of edge e, such that the clause contains those variables corresponding to vertex set e and with variables corresponding to vertices in c complemented. Is the formula corresponding to H satisfiable?"

CONJUNCTIVE QUERY: "Given an edge labelled hypergraph H where edge labels are tables of locally admissible assignments of values to the vertices of the edge, is there an assignment of values to vertices consistent with all hyperedge tables?"

This problem is central for query evaluation in databases (where the tables represent relations of the data base and the hypergraph represents an existential conjunctive query). Variants often occur in AI applications.

NON-SERIAL OPTIMIZATION [9]: "Given an edge labelled hypergraph H, where edge labels are tables giving, for each locally admissible assignment of values to vertices of the edge, a function value. What is the smallest value for the sum of function values for a globally admissible assignment of values to vertices in V?"

Surprisingly many combinatorial problems can be defined as instances of CONJUNCTIVE QUERY and NON-SERIAL OPTIMIZATION. As an example, MAX IND SET is a case of NON-SERIAL OPTIMIZATION where the hypergraph is an ordinary graph and the function table for its edge $\{v, w\}$ is

0	1	$-1/\deg(w)$
1	0	$-1/\deg(v)$
1	1	0

where vertex value 0 indicates membership and 1 non-membership in the

independent set. This construction also gives an NP-hardness proof for NON-SERIAL OPTIMIZATION.

Another important family of problems are arrangement problems, where an ordering of the vertices of the problem graph is sought. For a linear ordering of the vertices of G, let the *length* of an edge be the difference between the index of its last and first vertex.

HAMILTONIAN CIRCUIT: "Given G, is there an ordering of its vertices such that each vertex is adjacent to its successor and the last vertex is adjacent to the first?"

If we edge-label G with distances and ask for a shortest Hamiltonian circuit we get a constrained traveling salesman problem.

OPTIMAL LINEAR ARRANGEMENT: "Given G and an integer k, is there an ordering giving total edge-length at most k?"

BANDWIDTH: "Given G and an integer k, is there an ordering of vertices of G such that the largest edge-length is at most k?"

CUTWIDTH: "Given G and an integer k, is there an ordering such that, for no vertex v, more than k edges join a predecessor of v and v or one of its successors?"

The complexity of the following two problems was settled in section 5:

MIN FILL-IN: "Given G, what is the minimum value of $|F_\pi|$?"

DIMENSION: "Given G, what is its dimension?"

Acknowledgements.

I would like to thank Andrzej Proskurowski for bringing my attention to this interesting problem area and for valuable suggestions on the organisation of this paper. Erik Tidén, Arnon Rosenthal and Björn Lisper have made significant suggestions for improving and correcting the manuscript.

REFERENCES

1. A. V. Aho, J. E. Hopcroft and J. D. Ullman, *Design and Analysis of Computer Algorithms*, Reading, Mass. Addison-Wesley, 1974.
2. S. Arnborg, *Reduced state enumeration – another algorithm for reliability evalutation*, IEEE Trans. Reliability R-27 (1978), 101–105.
3. S. Arnborg, *On the complexity of multivariable query evaluation*, FOA Rapport C 20292-D8, National Defence Research Institute, Stockholm, Sweden (1979).

4. S. Arnborg, D. G. Corneil and A. Proskurowski, *Complexity of finding embeddings in a k-tree*, TRITA-NA 8407, Royal Institute of Technology, Sweden (1984).
5. S. Arnborg and A. Proskurowski, *Characterization and recognition of partial k-trees*, TRITA-NA 8402, Royal Institute of Technology, Sweden (1984).
6. S. Arnborg and A. Proskurowski, *Linear time algorithms for NP-hard problems on graphs embedded in k-trees*, TRITA-NA-8404, The Royal Institute of Technology (1984).
7. B. Aspvall, *Efficient algorithms for certain satisfiability and linear programming problems*, PhD Thesis, STAN-CS-80-822, Stanford University, 1980.
8. L. W. Beineke and R. E. Pippert, *Properties and characterizations of k-trees*, Mathematika 18 (1971), 141–151.
9. U. Bertele and F. Brioschi, *Nonserial Dynamic Programming*, Academic Press, New York, 1972.
10. C. J. Colbourn and A. Proskurowski, *Concurrent transmissions in broadcast networks*, in Proc. 11th Int'l Coll. on Automata, Languages, and Programming, Antwerp, Springer Verlag, Berlin (1984), 128–136.
11. D. G. Cornell and J. M. Keil, *A dynamic programming approach to the dominating set problem on k-trees*, Dept. of Computer Science, University of Toronto, Technical report (1983).
12. D. G. Corneil, H. Lerchs and L. Stewart Burlingham, *Complement reducible graphs*, Discrete Appl. Math. 3 (1981), 163–174.
13. M. Davis and H. Putnam, *A computing procedure for quantification theory*, J. ACM 7 (1960), 201–215.
14. R. J. Duffin, *Topology of series-parallel networks*, J. Math. Anal. Appl. 10 (1965), 303–318.
15. E. C. Freuder, *Synthesizing constraint expressions*, C. ACM 21 (1978) , 958–966.
16. M. R. Garey, R. L. Graham, D. S. Johnson and D. E. Knuth, *Complexity results for bandwidth minimization*, SIAM J. Appl. Math. 34 (1978), 835–859.
17. M. R. Garey and D. S. Johnson, *Computers and Intractability*, W. H. Freeman and Company, San Francisco (1979).
18. F. Gavril, *Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of chordal graph*, SIAM J. Comput. 1 (1972), 180–187.
19. G. Huet and D. Oppen, *Equations and rewrite rules: a survey*, in Formal Languages: Perspective and Open Problems (R. Book, Ed.), Academic Press, New York, 1980.
20. D. S. Johnson, *The NP-Completeness column: an ongoing guide*, J. of Algorithms 1–4 (1981–4).
21. S. Kirkpatrick, C. D. Gelatt, Jr. and M. P. Vecchi, *Optimization by simulated annealing*, SCIENCE 220 (1983), 671–680.
22. D. Rose, *Triangulated graphs and the elimination process*, J. Math. Anal. Appl. 32, (1970), 597–609.
23. D. Rose, *On simple characterizations of k-trees*, Discrete Math. 7 (1974), 317–322.
24. A. Rosenthal, *Computing the reliability of a complex network*, SIAM J. Appl. Math. 32 (1977) 384–393.
25. A. Rosenthal, *Dynamic programming is optimal for nonserial optimization problems*, SIAM J. Comput. 11 (1982), 47–59.
26. A. Rosenthal, *Series-parallel reduction for difficult measures of network reliability*, NETWORKS 11 (1981), 323–334.
27. K. Takamizawa, T. Nishizeki and N. Saito, *Linear-time computability of combinatorial problems on series-parallel graphs*, J. ACM 29 (1982), 623–641.
28. R. E. Tarjan, *Decomposition by clique separators*, Discrete Math., to appear.
29. J. A. Wald and C. J. Colbourn, *Steiner trees, partial 2-trees, and minimum IFI networks*, Networks 13 (1983), 159–167.
30. D. L. Walz, *Generating semantic descriptions from drawings of scenes with shadows*. AI-TR-271, A.I. Lab., M.I.T., Cambridge, Mass., 1972.
31. S. H. Whitesides, *An algorithm for finding clique cutsets*, Inf. Proc. Letters 12 (1981), 31–32.
32. M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM J. Alg. and Discr. Methods 2 (1981), 77–79.
33. M. Yannakakis, *A polynomial algorithm for the MIN CUT LINEAR ARRANGEMENT of Trees*, Proc. 24th Annual Symp. on Foundations of Computer Science, IEEE, 1983, 274–281.