Lecture Notes in Mathematics

Edited by A. Dold, B. Eckmann and F. Takens

Subseries: Fondazione C.I.M.E., Firenze
Adviser: Roberto Conti

1403

B. Simeone (Ed.)

Combinatorial Optimization

Como 1986

Springer-Verlag

Lecture Notes in Mathematics

Edited by A. Dold, B. Eckmann and F. Takens

Subseries: Fondazione C.I.M.E., Firenze
Adviser: Roberto Conti

1403

B. Simeone (Ed.)

Combinatorial Optimization

Lectures given at the 3rd Session of the
Centro Internazionale Matematico Estivo (C.I.M.E.)
held at Como, Italy, August 25–September 2, 1986

Springer-Verlag
Berlin Heidelberg New York London Paris Tokyo Hong Kong

Editor

Bruno Simeone
Dipartimento di Statistica,
Probabilità e Statistiche Applicáte
Università di Roma "La Sapienza"
Piazzale Aldo Moro 5, 00185 Roma, Italy

Mathematics Subject Classification (1980): 90C27; 68R99

ISBN 3-540-51797-9 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-51797-9 Springer-Verlag New York Berlin Heidelberg

Printing and binding: Druckhaus Beltz, Hemsbach/Bergstr.
2146/3140-543210 – Printed on acid-free paper

PREFACE

The present volume contains the proceedings of the CIME International Summer School on "Combinatorial Optimization", which was held at Villa Olmo, Como, Italy, from August 25 to September 2, 1986.

This was the first CIME Summer School specifically devoted to this quickly developing area, although the Varenna School on "Matroid Theory and its Applications" organized by Prof. Barlotti in 1980 did already include some lectures on matroid optimization. As a matter of fact, the idea of the present School came up for the first time there.

Combinatorial Optimization has a peculiar location in the map of Applied Mathematics, being placed in an interzone in the middle of Combinatorics, Computer Science and Operations Research. From a mathematical point of view, it draws on pure combinatorics, including graphs and matroids, on Boolean algebras and switching functions, partially ordered sets, group theory, linear algebra, convex geometry and probability theory, as well as other tools. Over the past years, a substantial amount of research has been devoted to the connections between Combinatorial Optimization and theoretical Computer Science, and in particular to computational complexity and algorithmic issues. Quite often the study of combinatorial optimization problems is motivated by real-life applications, such as scheduling, assignment, location, distribution, routing, districting, design and other Operations Research applications.

Although references to actual applications were frequently given, the emphasis of the School has been on theoretical aspects of Combinatorial Optimization. The four invited Lecturers, Prof. Peter L. Hammer, Rutgers University, USA; Prof. Ellis L. Johnson, IBM Scientific Research Center, Yorktown Heights, USA; Prof. Bernhard Korte, University of Bonn, West Germany; and Prof. Eugene L. Lawler, University of California, Berkeley, USA, have given a broad account of recent results and current trends in the area. Special attention has been devoted to the study of important classes of functions (either real- or binary-valued) defined on the binary n-cube (Prof. Hammer); to polyhedral combinatorics and its connections with combinatorial duality theories and min-max identities (Prof. Johnson); to the deep link between

greedy algorithms and finite geometries such as matroids and greedoids (Prof. Korte); to the role of submodularity (a discrete analogue of convexity) and to a general decomposition theory leading to linear-time graph algorithms (Prof. Lawler).

Contributed papers were presented by D. Acketa, C. Arbib, J. Bisschop, S. Dragutin, O. Holland, M. Lucertini, S. Pallottino, G. Pirillo, W. Piotrowski, B. Simeone, and P. Winter, and many of them are collected in this volume. We have also included contributions by M. Conforti and P. Hansen, who had planned to attend the School, but at the last moment were not able to come.

It is a pleasure to acknowledge the financial support of CIME, as well as the valuable assistance provided by Prof. Roberto Conti, Director, and Prof. Antonio Moro, Secretary of CIME. I am grateful to Fondazione "A. Volta" and its Director Prof. Giulio Casati for their kind hospitality: the elegant neo-classic architectures of Villa Olmo and the scenic beauty of Lake Como have created a charming atmosphere for the School; and the local staff, in particular Dr. Chiara De Santis and Mrs. Donatella Marchegiano, has efficiently handled even the tiniest logistic details. I also thank my colleagues Prof. Mario Lucertini and Prof. Stefano Pallottino for their personal help in the organization of the School. Finally, my deepest thanks to the invited Lecturers and to the other participants for their individual contributions to the School.

Bruno Simeone,
University of Rome "La Sapienza"

TABLE OF CONTENTS

Courses

Seminars

QUADRATIC FUNCTIONS OF BINARY VARIABLES

by
Peter L. Hammer
RUTCOR, Rutgers University, New Brunswick, NJ, USA
and
Bruno Simeone
RUTCOR, Rutgers University, New Brunswick, NJ, USA
and
Department of Statistics, University of Rome, Italy

Contents

1 Introduction

The present survey is devoted to quadratic functions of n binary variables. Part I (Sec. 2 to 3) deals with binary-valued functions (*boolean functions*, *truth functions*) and its main theme is the efficient solution of quadratic boolean equations; Part II (Sec. 4 to 17) deals with real-valued functions (*pseudo-boolean functions*, *set-functions*) and focuses on the maximization of such functions over the binary n-cube.

Quadratic functions of binary variables deserve attention for a variety of reasons. They naturally arise in modelling *interactions*. Consider a set of n objects, labelled $1, 2, ..., n$, each of which can be either chosen or not. Assume that for any pair (i, j) of objects a real number a_{ij}, measuring the "interaction" between i and j, is given. Also, assume that the global interaction is the sum of the interactions between all pairs of chosen objects. Let $x_i = 1$ or 0 depending on whether object i is chosen or not. Then the global interaction can be written as a quadratic function $\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_j x_j$ of the n variables $x_1, ..., x_n$.

For example, inter-city traffic [Rhys (1970)] and kinetic energy in spin-glass models [Kirkpatrick, Gelatt and Vecchi (1983)] can be represented in this way.

Quadratic functions of binary variables also naturally arise in *least-square approximation*. Assume that a weight w_i is assigned to each object $i = 1, ..., n$, and that one wants to choose a subset of objects whose total weight is as close as possible to a "target" weight t. This leads to the minimization of the quadratic function $(w_1 x_1 + \cdots + w_n x_n - t)^2$. One nice application deals with the optimal distribution of cargoes among the trips of a space shuttle in the supply support system of a lunar base [Freeman, Gogerty, Graves, and Brooks (1966)]. As another example, consider the *optimal regression* problem [Beale, Kendall, and Wall (1967)]. An endogenous variable Y is approximated by a linear function $a_1 Z_1 + \cdots + a_n Z_n$ of n exogenous variables $Z_1, ..., Z_n$. We assume that the coefficients a_j have been already estimated from a sample of m observations $(y_i, z_{i1}, ..., z_{in})$ of the variables Y, $Z_1, ..., Z_n$ via standard linear regression techniques. However, for practical reasons one often wants to choose only p variables ($p \ll n$) among $Z_1, ..., Z_n$ and express Y as a linear combination of the chosen variables with the smallest possible loss of information. Let $x_j = 1$ or 0 depending on whether variable Z_j is chosen or not. Then the problem can be formulated as the minimization of the quadratic function

$$\sum_{i=1}^{m} (y_i - a_1 z_{i1} x_1 - \cdots - a_n z_{in} x_n)^2$$

subject to the cardinality constraint $x_1 + \cdots + x_n = p$.

There is a rich and fruitful interplay between the theory of quadratic functions of binary variables and the theory of graphs. As a matter of fact, with any graph $G = (V, E)$, where $V = \{1, ..., n\}$ is the vertex-set and E is the edge-set, one can naturally associate the quadratic monotone boolean function $f_G(x) = \bigvee_{(i,j) \in E} x_i x_j$, and vice-versa. For this reason, quadratic monotone boolean functions are sometimes called *graphic*. One important direction of research attempts to find connections between combinatorial properties of G and functional properties of f_G. This line of research is typified by a theorem of Chvátal and Hammer (1977) stating that a graphic function is threshold if and only if the associated graph does not contain squares, paths of length 3, or parallel edges.

Graph-theoretic methods are very useful in the solution of quadratic boolean equations. Actually, all the fastest known solution algorithms are graph-theoretic in nature (see Sec. 3). Graphs are also a very useful tool in the maximization of quadratic pseudo-boolean functions. For example, a reduction of this problem to finding a maximum weight stable set in a graph is exhibited in Sec. 14. Conversely, many graph optimization problems can be naturally formulated as quadratic $0-1$ optimization problems (see Sec. 5).

One important reason for studying quadratic boolean equations is that, unlike equations of higher order, they can be solved in polynomial – in fact, linear – time. Furthermore, the most common types of logical relations,

"Either P or Q is false"

"Either P or Q is true"

"P implies Q"

are represented by *quadratic* equations, namely

$$pq = 0,$$

$$\overline{p}\,\overline{q} = 0,$$

$$p\overline{q} = 0,$$

respectively.

The survey is structured as follows. Some fundamental concepts of the theory of boolean functions and pseudo-boolean functions are recalled in Sec. 2 and 4, respectively. Sec. 3, which is based on [Petreschi and Simeone (1985)], describes some fast graph-theoretic algorithms for the solution of quadratic boolean equations.

Sec. 5 describes many combinatorial applications of quadratic $0-1$ maximization.

The remaining sections are devoted to a detailed account of the theory of roof-duality in quadratic $0-1$ maximization. They are mainly based on [Hammer, Hansen and Simeone (1984)]; however, new developments are reported in Sections 11, 12 and 13.

The theory of roof-duality allows one to get – with a modest computational effort – upper bounds of the maximum of a quadratic function f over the binary n-cube B^n. This is achieved by considering a special class of linear overestimators of f – the so called roofs – and maximizing them (instead of f) over B^n. "Best" roofs can be generated in polynomial time via maximal flow techniques (Sec. 15). There is no guarantee that the upper bound obtained by maximizing a best roof does coincide with the quadratic optimum; however, one can check whether this is indeed the case in polynomial time by solving a quadratic boolean equation (Sec. 17).

Sec. 16 is devoted to the intriguing *persistency* phenomenon: consider any maximum point x^* of the quadratic function f and any maximum point $\tilde{x}$ of a best roof g; then for every variable x_i with a non-zero coefficient in g one has $\tilde{x}_i = x_i^*$. This phenomenon, deserves, along with many other topics related to quadratic functions of binary variables, further substantial investigation.

PART I
Quadratic boolean functions and equations

2 Boolean functions and boolean equations

It is well known that the set $B = \{0, 1\}$, endowed with the operations

$$\begin{array}{ll} x \vee y = max\{x, y\} & \text{(union)} \\ x \wedge y = min\{x, y\} = x \cdot y & \text{(intersection or product)} \\ \overline{x} = 1 - x & \text{(complementation)} \end{array} \tag{2.1}$$

is a boolean algebra.

Let $x_1, \ldots, x_n$ be boolean variables, i.e. variables taking values in B. A *literal* is either a variable x_i or its complement $\overline{x}_i$. A *term* is a finite product of distinct literals a *boolean expression* (in *disjunctive normal form*, DNF) a finite union of terms.

A *boolean function* is any mapping $f : B^n \to B$. Given any boolean expression ϕ, one can associate with each $\alpha \in B^n$ the element $\phi(\alpha) \in B$ obtained by replacing in ϕ each variable x_i by α_i and evaluating the resulting expression according to (2.1). Such mapping $\alpha \mapsto \phi(\alpha)$ is a boolean function, and conversely each boolean function is representable in this way.

A boolean expression is called

- *primitive*, if no two distinct terms involve the same variables, no matter whether complemented or not. For example, the presence of the term xyz forbids the presence of terms $\overline{x}yz$, $\overline{x}\,\overline{y}z$, $\overline{x}\,\overline{y}\,\overline{z}$, $x\,\overline{y}z$ and so on, as well as other occurrences of xyz.

- *normal*, if all terms are different and if no two terms of the form xC, $\overline{x}C$ are present. Example: $xyz \vee \overline{x}\,\overline{y}\,\overline{z} \vee x\,\overline{y} \vee y\,\overline{z}$.

- *pure*, if every term contains at least one uncomplemented variable. Example: $xyz \vee \overline{x}y \vee y\,\overline{z}$.

- *mixed*, if every term contains both complemented and uncomplemented variables. Example: $x\,\overline{y}\,\overline{z} \vee \overline{x}y \vee y\,\overline{z}$.

Let ϕ be a boolean expression. A monomial I (i.e. a finite product of distinct literals but not necessarily a term of ϕ) is an *implicant* of ϕ if $I \leq \phi$; that is, when $I = 1$ then $\phi = 1$.

The implicant I is *prime* if there is no implicant $J \neq I$ such that $I \leq J \leq \phi$. The prime implicants of a boolean expression ϕ can be found by the following *consensus method*, due to Quine (1955): Starting from the list of terms of ϕ, execute as many times as possible the following two operations.

CONSENSUS: If two terms xC and $\overline{x}D$ are present in the current list and no variable is complemented in C and uncomplemented in D or vice versa, then add to the current list the *consensus* CD of the two terms after deleting possible repeated literals.

ABSORPTION: If the current list contains two terms C, D such that each literal in D appears in C, then delete D from the list.

Quine (1955) has proved that when no further consensus or absorption operation is possible, then the final list contains all prime implicants of ϕ, and only them.

The consensus method may well take exponential time. However, when the method is applied to *quadratic* boolean expressions, only quadratic or linear terms can be generated at each step. It follows that the number of terms in the list at each step is $O(n^2)$ and this in turn implies that in the quadratic case the consensus method runs in polynomial time.

Let us now turn our attention to boolean equations. Let ϕ be a boolean expression. The boolean equation $\phi = 0$ is said to be *consistent* if there exist some $\alpha \in B^n$ such that $\phi(\alpha) = 0$. Then α is called a *solution* of the equation.

It is easy to see that a boolean equation $\phi = 0$ is consistent if and only if the constant 1 is not a prime implicant of ϕ. Hence one method for checking the consistency of $\phi = 0$ consists in applying the consensus method to ϕ and to verify whether 1 is a prime implicant or not. Many methods for solving general or specific boolean equations have been proposed in the literature [see e.g. Rudeanu (1974)].

Clearly a pure boolean equation is always consistent, since $\alpha = (0, ..., 0)$ is always a solution. This statement can be somehow reversed, as shown by Prop. (2.1) below.

Given the boolean expression $\phi(x_1, \ldots, x_n)$, the *switch* on the variable x_i is the operation which replaces each occurence of x_i in ϕ by $\overline{x}_i$ and vice versa. Similarly one defines a switch on a set S of variables.

Proposition 2.1 *A boolean equation is consistent if and only if it can be transformed into a pure one by a switch on some set S of variables.*

Proof: Given the boolean equation $\phi(x_1, \ldots, x_n) = 0$, assume that $\alpha = (\alpha_1, \ldots, \alpha_n)$ is a solution. If $S = \{x_i : \alpha_i = 1\}$, the switch on S transforms the equation into a pure one. Conversely, suppose that there is some set S of variables such that the switch on S transforms the equation into a pure one. Then $(0, ..., 0)$ is a solution of the transformed equation; hence, if one defines $\alpha_i = 1$ or 0 according as x_i belongs to S or not, the vector $\alpha = (\alpha_1, \ldots, \alpha_n)$ is a solution of the original equation. □

3 Efficient graph-theoretic algorithms for solving quadratic boolean equations

In the present section, we shall describe three fast algorithms for the solution of a quadratic boolean equation $\phi = 0$:

- The Labelling Algorithm of Even, Itai and Shamir (1976) (this paper contains only an outline of the algorithm; more detailed descriptions can be found in Gavril (1977) and Simeone (1985);
- The Switching Algorithm of Petreschi and Simeone (1980);
- The Strong Components Algorithm of Aspvall, Plass and Tarjan (1979).

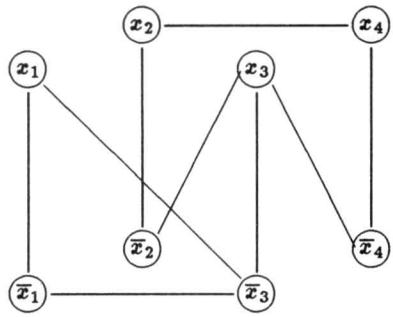

Figure 1: *The graph G associated with $\psi = x_1\overline{x}_3 \vee \overline{x}_1\overline{x}_3 \vee \overline{x}_2x_3 \vee x_2x_4 \vee x_3\overline{x}_4$.*

A common feature of the three above algorithms is their graph-theoretic nature: in the first two algorithms, the quadratic boolean expression ϕ is represented by an undirected graph (the so called "matched graph" or "clause graph"), while the third algorithm exploits a digraph model (the so called "implication graph").

Consider a quadratic boolean equation (in DNF) in n variables $x_1, \ldots, x_n$ and m terms

$$\phi \equiv T_1 \vee \ldots \vee T_n \;=\; 0, \tag{3.1}$$

where, without loss of generality, we may assume that each term is the product of exactly two literals and that the presence of the term $\xi\eta$ forbids the presence of other occurences of $\xi\eta$.

Definition. The *matched-graph* associated with ϕ is the undirected graph $G = (V, E)$, where $V = \{x_1, \ldots, x_n; \overline{x}_1, \ldots, \overline{x}_n\}$ and

$$E = \{\langle \xi, \eta \rangle \text{ for each term } \xi\eta \text{ of } \phi\} \cup \{\langle x_i, \overline{x}_i \rangle : i = 1, ..., n\}$$

The edges of G are classified into *positive*, *negative*, *mixed* or *null* ones according to whether they are of the form $\langle x_i, x_j \rangle$, $\langle \overline{x}_i, \overline{x}_j \rangle$, $\langle x_i, \overline{x}_j \rangle$, $\langle x_i, \overline{x}_i \rangle$, respectively.

Figure 1. shows the matched graph G associated with the following expression

$$\psi = x_1\overline{x}_3 \vee \overline{x}_1\overline{x}_3 \vee \overline{x}_2x_3 \vee x_2x_4 \vee x_3\overline{x}_4 \tag{3.2}$$

As shown by Theorem (3.1) below, the consistency of the quadratic boolean equation $\phi = 0$ has a nice graph-theoretic counterpart in the matched graph G associated with ϕ. We recall that, given an arbitrary graph $G' = (V', E')$, a *matching* M of G' is any set of pairwise non-incident edges; a *transversal* T of G' is any set of vertices such that every edge of G' has at least an endpoint in G'; and that G' is said to have the *König-Egerváry Property* (briefly, the KE Property) if the maximum cardinality of a matching is equal to the minimum cardinality of a transversal.

Theorem 3.1 : *The quadratic boolean equation $\phi = 0$ is consistent iff the matched graph G associated with ϕ has the König-Egerváry Property.*

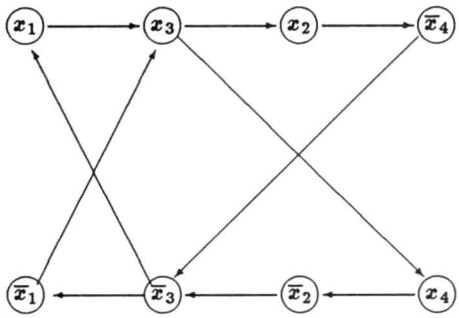

Figure 2: *The implication graph of ψ, as given by (3.2).*

Proof: See Simeone (1985). □

Gavril (1977) has described an $O(m)$ algorithm for recognizing the König-Egerváry Property in a graph with m edges, assuming that a maximum cardinality matching is at hand. One such matching can be found in $O(n^{2.5})$ time, e.g. via the implementation of Edmonds' Blossom Algorithm described in Micali and Vazirani (1980).

In view of Theorem (3.1), one can check the consistency of the equation $\phi = 0$ by executing Gavril's algorithm on the matched graph G associated with ϕ. (Note that the null edges form a maximum cardinality matching of G). The resulting procedure turns out to be essentially Even, Itai and Shamir's method.

Definition. The *implication graph* associated with ϕ is the digraph $D(V, A)$, where V is defined as above and

$$A = \{(\xi, \overline{\eta}), (\overline{\xi}, \eta) \text{ for each term } \xi\eta \text{ in } \phi\}.$$

The digraph D is isomorphic to the digraph $\tilde{D}$ obtained from D by reversing the orientation of every edge and complementing the name of every variable.

The implication graph of ψ (as given by (3.2)) is shown in Figure 2.

We are ready to describe the three above mentioned algorithms.

3.1 Labelling Algorithm

The idea of the algorithm is to *guess* the value of an arbitrary literal ξ in some solution and to *deduce* the possible consequences of this guess on other variables appearing in the expression. Since ξ can take either the value 0 or the value 1, the algorithm analyzes in parallel the consequences of these two alternative guesses on ξ. One keeps track of these consequences by a "red" labelling (corresponding to the guess $\xi = 1$) and by a "green" labelling (corresponding to the guess $\xi = 0$). Initially all terms are declared to be "red-unexplored" and "green- unexplored"; then one selects on arbitrary literal ξ and assigns

Step	Red-explored terms	Green-explored terms	Red labels	Green labels
0	/	/	$x_1 = 1;\ \overline{x}_1 = 0$ (guess)	$x_1 = 0;\ \overline{x}_1 = 1$ (guess)
1	$x_1\overline{x}_3$		$\overline{x}_3 = 0;\ x_3 = 1$	
2		$\overline{x}_1\overline{x}_3$		$\overline{x}_3 = 0;\ x_3 = 1$
3	$\overline{x}_2x_3$		$\overline{x}_2 = 0;\ x_2 = 1$	
4		$x_3\overline{x}_4$		$\overline{x}_4 = 0;\ x_4 = 1$
5	$x_3\overline{x}_4$		$\overline{x}_4 = 0;\ x_4 = 1$	
6		$\overline{x}_2x_3$		$\overline{x}_2 = 0;\ x_2 = 1$
7	x_2x_4		$x_4 = 0;\ \overline{x}_4 = 1$ conflict	
8		x_2x_4	conflict	$x_4 = 0;\ \overline{x}_4 = 1$

Table 1: *Steps for the Labelling algorithm for ψ as given by (3.2)*

to it the red label 1 and the green label 0. Then $\overline{\xi}$ must receive the red label 0 and the green label 1.

Then the two labellings are extended to as many literals as possible by alternately performing for the red labelling and for the green one the following *STEP*.

STEP: Take an arbitrary unscanned term $\eta\zeta$ such that η has the label 1, and assign to ζ and to $\overline{\zeta}$ the labels 0 and 1, respectively, making sure that ζ did not previously get the label 1. Declare the term $\eta\zeta$ scanned. (Of course, terms like "label", "unscanned", "scanned" involved in *STEP* are relative to the color currently under consideration).

If a conflict arises, say, for the red labelling (i.e. a literal which was peviously red-labelled 1 is forced to get the red label 0 or vice versa), the red labelling stops and the red labels are erased. If, at a later stage, a conflict occurs also for the green labelling, the algorithm stops and the equation has no solution. It may happen that a labelling, say the red one, "gets stuck": no conflict has occurred, but there are still literals having no red label. This is possible only when, for each red-unexplored term, the literals appearing in that term are either red-unlabelled or have red label 1. If this situation occurs then red labels are taken for granted and both the red- and the green-labellings are restarted on the reduced expression involving only the red-unlabelled literals.

The algorithm can be shown to run in $O(m)$ time (see Gavril (1977)). As an example, the following Table 1 summarizes the algoritm steps when the input expression is ψ as given by (3.2).

The expression ψ is not satisfiable because both labellings end up in a conflict.

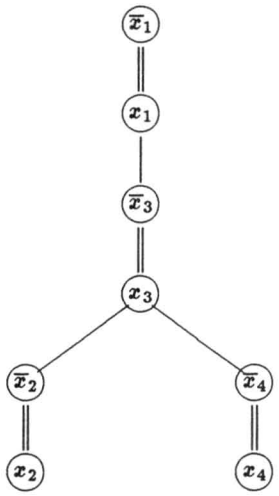

Figure 3: *The alternating tree rooted at $\overline{x}_1$ for the graph of Figure 1.*

3.2 Switching algorithm

On the basis of Prop. (2.1), the algoritm tries to transform the given expression ϕ into a pure one, if possible, through a sequence of switches. Before describing the algorithm, it is convenient to introduce some preliminary definitions. (we use the tree-terminology of Aho, Hopcroft and Ullman (1974)).

Definition: A variable x in ϕ is said to be *forced* to the value α ($\alpha = 0$ or 1) if either the equation $\phi = 0$ is inconsistent or x has the value α in all solutions.

Definition: An *alternating tree rooted at* $\overline{x}_i$ is any subgraph of G (the matched graph of ϕ) which is a tree $T(\overline{x}_i)$ rooted at $\overline{x}_i$ and has the following properties:

1. $\overline{x}_i$ is the root;
2. if $\overline{x}_j$ is a vertex of $T(\overline{x}_i)$, then its father in $T(\overline{x}_i)$ is $\overline{x}_j$.
3. if $\overline{x}_j$ is a vertex of $T(\overline{x}_i)$, then its father is a vertex x_r of $T(\overline{x}_i)$, such that $\langle x_r, \overline{x}_j \rangle$ is a mixed edge of G;
4. if x_j is a vertex of $T(\overline{x}_i)$ and $\langle x_j, \overline{x}_r \rangle$ is a mixed edge of G, then $\overline{x}_r$ is a vertex of $T(\overline{x}_i)$.

For the matched graph of Figure 1., an alternating tree rooted at $\overline{x}_1$ is shown in Figure 3.

Definition: The *join* of two vertices ξ and η of $T(\overline{x}_i)$ is their common ancestor which is farthest from the root $\overline{x}_i$.

Let us now briefly describe the switching algorithm. Again the algorithm works on the matched graph G. An endpoint $\overline{x}_i$ of a negative edge $\overline{x}_i\overline{x}_r$ is selected and the alternating tree $T(\overline{x}_i)$ is grown. As soon as a new vertex x_h of $T(\overline{x}_i)$ is generated, one checks whether there is in G any positive edge x_hx_k linking x_h to a previously generated vertex x_k of $T(\overline{x}_i)$. If this is the case, the variable x_j corresponding to the join of x_h and x_k must be forced to 0.

As a consequence, other variables are forced in cascade according to the following rules:

(i) if ξ is forced to 0, then $\overline{\xi}$ is forced to 1;

(ii) if ξ is forced to 1 and $\langle \xi, \eta \rangle$ is an edge of G, then η is forced to 0.

If during this process a conflict occurs (that is, some variableis forced to both 0 and 1), then the algorithm stops pointing out that the equation is inconsistent. Otherwise one obtains a reduced equation involving fewer variables and a new cycle begins. If the construction of $T(\overline{x}_i)$ has been completed and no positive edge between two vertices of $T(\overline{x}_i)$ has been detected, then a switch is performed on all the variables corresponding to vertices of $T(\overline{x}_i)$. In this way one obtains an equivalent expression and a new cycle begins. The procedure is iterated until either a pure equation is obtained or all variables are forced. In both cases a solution of the original equation $\phi = 0$ can be found by inspecting the list of the forced variables and the list of the switched ones.

As an example, given ψ as in (3.2), the alternating tree $T(\overline{x}_1)$ of Figure 3. is grown.

Since x_2 and x_4 are linked by a positive edge, their join x_3 is forced to 0 and hence $\overline{x}_3$ is forced to 1. In turn, $\overline{x}_3 = 1$ forces $x_1 = 0$ and $\overline{x}_1 = 1$. On the other hand, because of the edge $\langle \overline{x}_1, \overline{x}_3 \rangle$, $\overline{x}_1$ must also be forced to 0 and thus a conflict arises. Hence the equation $\psi = 0$ is inconsistent.

For a formal description of the algorithm and for a correctness proof, the reader is referred to Petreschi and Simeone (1980), where a worst-case $O(mn)$ bound on the time-complexity of the algorithm is also given.

3.3 Strong components algorithm

This algorithm is based on the following key result.

Theorem 3.2 (Aspvall, Plass and Tarjan (1979)) . *The equation $\phi = 0$ is consistent iff in the implication graph D no vertex x_i is in the same strong component as its complement $\overline{x}_i$ (i.e. no circuit of D contains both x_i and $\overline{x}_i$).* □

The algorithm works on D and preliminarly finds the strong components of D in reverse topological order (see Tarjan (1972)). The isomorphism between D and $\tilde{D}$ implies that for every strong component SC of D there exist a "mirror" component $\tilde{SC}$, the *complement* of SC, induced by the complements of the vertices in SC. Hence Theorem (3.2) can be restated as follows: "ϕ is satisfiable iff in D no strong component coincides with its complement".

The general step of the algorithm consists in processing the strong components of D in the following way.

For each strong component SC, one of the following cases must occur:

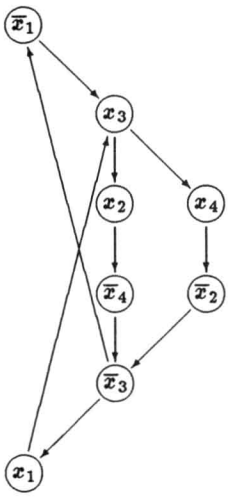

Figure 4: *The spanning arborescence generated by the algorithm of Tarjan (1972).*

(a) SC is already labelled. The algoritm processes another strong component.

(b) $SC = \tilde{SC}$. The algorithm stops. In view of Theorem (3.2) the equation $\psi = 0$ is not consistent.

(c) SC is unlabelled. The algorithm assigns the label 1 to SC and the label 0 to $\tilde{SC}$.

SC_1, is said to be a *predecessor* of SC_2 (and SC_2 a successor of SC_1,) if there exists an edge from some vertex of SC_1, to some vertex of SC_2. It is easy to show that every component labelled 1 has only components with label 1 as successors and every component labelled 0 has only components with label 0 as predecessors. Thus, if we assign to each vertex ξ the label of the component containing ξ_1 we get a solution to $\phi = 0$.

As an example, we consider again ψ and the associated implication graph D of Figure 2.

The algorithm of Tarjan (1972) for finding the strong components produces the arborescence of Figure 4: the digraph D turns out to be strongly connected. For every i, the vertices x_i and $\overline{x}_i$ belong to the same strong component and thus, by Theorem (3.2), the equation $\psi = 0$ is inconsistent.

The complexity of the Strong Components Algorithm is $O(m)$. For further details, see Aspvall,Plass and Tarjan (1979).

A randomized algorithm with expected $O(n)$ time has been described by Hansen, Jaumard and Minoux (1984).