Boolean simplification and integer inequalities

J. M. Wilson

Department of Management Studies, Loughborough University of Technology, Loughborough, Leicestershire LE11 3TU

A method for the minimisation of Boolean expressions is presented. The algorithm has as its basis the Quine approach, but it is geared to handling larger sets of Boolean variables, rather than small sets of variables encountered in traditional examples. These larger sets of variables occur when a Boolean minimisation process is incorporated in an Integer Programming algorithm.

(Received December 1975)

1. Introduction

Various algorithms for handling sets of integer inequalities using Boolean algebra have been developed by Granot and Hammer, (1970). Many of these algorithms require that a method of Boolean simplification such as that of Quine (1955) be used. As the Quine procedure in its usual form is not always suitable for computation, an approach is developed here which leads to a more systematic procedure.

2. Definitions

1. Let the Boolean operation of union be defined on x, y $(x, y \in (0, 1))$ as
$$x \vee y = \max(x, y) .$$
2. Let the Boolean operation of multiplication (product) be defined on x, y $(x, y \in (0, 1))$ as $xy = \min(x, y)$.
3. Let $\bar{x} = 1 - x (x \in (0, 1))$ be defined as the complement of x.
4. Let the consensus of $x\phi$ and $\bar{x}\psi$ with respect to x be defined as $\phi\psi$ where ϕ, ψ are Boolean products such that $\phi\psi$ contains no variable which is both complemented and uncomplemented.
5. A product of several Boolean variables is termed a conjunction.

3. Integer inequalities

An integer inequality of the form
$$\sum_{i=1}^{n} x_i \leqslant n - 1 \quad (1) \quad (x_i \in (0, 1); i = 1, \ldots, n)$$
may be represented by the Boolean conjunction
$$f = x_1 x_2 \ldots x_n$$
in the sense that
$$x_1 = y_1, x_2 = y_2, \ldots, x_n = y_n \quad (y_i \in (0, 1); i = 1, \ldots, n)$$
is a solution to (1) if and only if
$$f(y) = y_1 y_2 \ldots y_n = 0 .$$
It has been shown (Granot and Hammer, 1970) that for each system of inequalities
$$\sum_{i=1}^{n} a_{ij} x_i \leqslant b_j \quad (x_i \in (0, 1); i = 1, \ldots, n) \qquad (2)$$
$$j = 1, \ldots, m$$
there exists another system of form
$$\sum_{k_l \in S_l} \tilde{x}_{k_l} \leqslant |S_l| - 1 \quad (l = 1, 2, \ldots, t) \qquad (3)$$
$$S_l \subseteq \{1, 2, \ldots, n\} \; \forall \; l = 1, \ldots, t$$
(where $\tilde{x}_{k_l} = x_{k_l}$ or $\bar{x}_{k_l}$
and $|S_l|$ denotes the number of elements in the set S_l) such that the set of solutions to (2) is identical to the set of solutions to (3).

Any solution to (2) will be such that
$$f_l = \prod_{k_l \in S_l} \tilde{x}_{k_l} = 0 \quad (l = 1, 2, \ldots, t)$$
and so any solution to (2) will be such that
$$\phi = f_1 \vee f_2 \vee \ldots \vee f_t = 0 .$$
The function ϕ is called the resolvent of the system of inequalities (2) and is a union of conjunctions. It is likely that the function ϕ will possess Boolean redundancies, e.g. for the inequalities
$$x_1 + x_2 + x_3 \leqslant 2$$
$$x_1 + \bar{x}_2 + x_3 \leqslant 2$$
$$\phi = x_1 x_2 x_3 \vee x_1 \bar{x}_2 x_3 .$$
However, using the usual procedure of Quine (1955) simplification ϕ may be reduced to $\phi = x_1 x_3$, as the other conjunctions subsume (and imply) $x_1 x_3$.

4. Simplification approach

The procedure of converting a set of inequalities to a resolvent has been found to be useful in solving the integer programming problem
$$\text{maximise} \sum_{i=1}^{n} c_i x_i$$
subject to
$$\sum_{i=1}^{n} a_{ij} x_i \leqslant b_j \quad (j = 1, \ldots, m)$$
$$(x_i \in (0, 1); i = 1, \ldots, n)$$
(for further details see Granot and Hammer (1970).

When handling the resolvent to solve integer programming problems frequent use of the operation of Quine simplification will be required. It has been shown by Quine that the operation of developing the consensi of a union of conjunctions may be done in a finite number of steps. However, in many of the examples considered in Boolean simplification (see for instance Quine (1955) and Zissos and Duncan (1973)) the number of Boolean variables is small. When the procedure of Quine simplification is used for a larger set of variables a more systematic approach is needed as the operations will be necessarily programmed on a computer. A systematic approach to the handling of Quine simplification can be developed because of the following result.

Theorem

For a resolvent ϕ of Boolean variables $x_1, \ldots, x_n$ the set of all conjunctions derivable by consensus operations within ϕ can be derived by performing the operation of consensus on the conjunctions separately with respect to each of the n variables $x_1, x_2, \ldots, x_n$ each taken in turn in any order. Hence, the operations may be completed in n sets of steps.

(*a*) Original Conjunctions

	x_2	$\bar{x}_2$
—	*c*	*d*
x_1	*e*	*f*
$\bar{x}_1$	*g*	*h*

(*b*) New conjunctions after consensus with respect to x_2

x_1	$\bar{x}_1$
cd	*cf, de, ef*	*ch, dg, gh*

(*c*) New conjunctions after consensus with respect to x_1

x_2	$\bar{x}_2$
ab	*ag, be, eg*	*ah, bf, fh*

(*d*) New conjunctions after consensus with respect to x_1 (after x_2) *or* x_2 (after x_1).

	x_2	$\bar{x}_2$
	cd, ab, ach, bcf, cfh, agh deg, adg, bef bde, efgh	*ag, be, eg*	*ah, bf, fh*
x_1	*cf, de, ef*	—	—
$\bar{x}_1$	*ch, dg, gh*	—	—

Fig. 1

*Proof**

1. Consider first the variables x_1, x_2.
The conjunctions involving x_1, x_2 are

$$ax_1, b\bar{x}_1, cx_2, d\bar{x}_2, ex_1x_2, fx_1\bar{x}_2, g\bar{x}_1x_2, h\bar{x}_1\bar{x}_2 \qquad (1)$$

where $a, b, \ldots, h$ are Boolean products in other variables. Now if consensus is performed on set (1) with respect to x_1 and then on the augmented set of conjunctions and consensi with respect to x_2, 23 new conjunctions are produced.
If consensus is performed on set (1) with respect to x_2 and then on the augmented set of conjunctions and consensi with respect to x_1, 23 new conjunctions are produced. (see **Fig. 1**)
These two sets of 23 consensi conjunctions are identical and so the results of the operation of consensus are independent of the order chosen for x_1, x_2. But x_1, x_2 were any variables so that the operation of consensus can be performed with respect to any two variables in turn in any order.

2. Results in permutation theory justify that an ordered set of n variables may be expressed as any other ordered set of n variables after a sequence of pairwise interchanges of variables has taken place. 1, above shows that consensus is independent of the order of any pair of variables, consensus operations may be performed in any order on n variables and produce the same set of consensi.

The simplification approach in algorithmic form is as follows: for a set S of conjunctions in Boolean variables $x_1, x_2, \ldots, x_n$ select a variable x_i and partition S into three subsets,

S_{i_1} those conjunctions which involve x_i (uncomplemented)

S_{i_2} those conjunctions which involve $\bar{x}_i$

S_{i_3} those conjunctions which do not involve x_i or $\bar{x}_i$.

The consensi between conjunctions in set S_{i_1} and set S_{i_2} are now formed. Any consensus which subsumes conjunctions in S may be removed as it is redundant and any conjunction in S which may be replaced by a conjunction which it subsumes and which involves fewer variables is replaced by the appropriate conjunction.

Let S_1 denote the modified set formed from S after any replacements have been made. There are now two sets of conjunctions—the set S_1 and the set S^*, of conjunctions derived from S. A new variable x_j not already chosen is now chosen to replace x_i; the split into three subsets is made over $S_1 \cup S^*$ into $S_{j_1}, S_{j_2}, S_{j_3}$ such that $S_{j_1} \cup S_{j_2} \cup S_{j_3} = S_1 \cup S^*$.
Consensus proceeds between S_{j_1} and S_{j_2} and the consensi formed are now included in the set S^*. Removal of redundancies takes place as before and the set S_2 is formed from S_1. The process continues for each of the n variables $x_1, \ldots, x_n$ until finally the situation is reached where the sets S_n and S^* are left (S^* being redundant to S_n).

For the purposes of integer programming the set S_n will suffice as a representation of the conjunctions as the Boolean information is now in a concise form; however, criteria such as those of Zissos and Duncan (1973) can be used to shorten the size of the final representation. This option was included in one computer program and made no appreciable reduction in time used.

Notes

1. Various rules for which variable x_i to select can be adopted. A rule which was found convenient to program was to choose that x_i which would give rise to the smallest number of potential conjunctions as it was known that all consensi would be generated eventually, i.e. choose i such that $|S_{i_1}| \times |S_{i_2}|$ is minimised.
2. That no further conjunctions could be produced after the n sets of steps have been completed can be deduced from the Theorem. For if the order of variable selection had been such that

$$x_1, x_2, \ldots, x_r, \ldots, x_n, x_r$$

were to be used expecting to generate new conjunctions at step $(n + 1)$, then this order could be altered to

$$x_1, x_2, \ldots, x_r, x_r, \ldots, x_n$$

by a sequence of pairwise interchanges and so would produce an equivalent set of consensi. But this order is clearly equivalent to

$$x_1, x_2, \ldots, x_r, \ldots, x_n \ .$$

Thus as the Theorem justifies pairwise interchanges of order in the variable selected for consensus, then no more new consensi can arise after the n steps of the process have been completed.
3. The fact that the process of consensus ends after n sets of steps is more convenient to program than an approach which is only completed once it can show that no further new consensi can be generated. In the Quine approach the end is reached when no further new consensi can be found. This condition is awkward to handle in a computer program.

5. Results

In a typical (0-1) integer programming problem of 40 constraints and 20 variables the problem is solved in about 100 seconds of CPU time on an ICL 1900 machine. The process involves around ten complete uses of the simplification routine. The program was written in FORTRAN and is such that up to 1,000 consensi can be generated and held in core storage before deletion has to be used to make space available. However, deletion usually takes place before as many as 1,000 consensi have been generated. The constraints give rise to around 200 initial conjunctions in the resolvent.

6. Conclusions

An approach has been developed for systematically using Quine simplification to handle sets of n variables. The approach

*A fuller version of this proof is contained in (Wilson (1975).

has similarities to the McCluskey version of Quine's method (McCluskey, 1962) but is more geared to general types of Boolean expressions. McCluskey's approach is more dependent on many similar Boolean expressions occurring in the set of Boolean expressions to be simplified. This allows drastic simplification to take place early on, but in the integer programming problems it was found that the type of Boolean expressions arising were not amenable to this treatment. Hence a completely systematic approach of the type described was necessary.

References

GRANOT, F. and HAMMER, P. L. (1970). On the Use of Boolean Functions in Bivalent Programming, *Methods of Operations Research*, Vol. 12, pp. 154-184

QUINE, W. V. (1955). A Way to Simplify Truth Functions, *American Mathematical Monthly*, Vol. 62, pp. 627-631

ZISSOS, D. and DUNCAN, F. G. (1973). Boolean minimisation, *The Computer Journal*, Vol. 16, No. 2, pp. 174-179

WILSON, J. M. (1975). Boolean Methods of (0-1) Integer Programming, D. Phil. thesis, University of Sussex

MCCLUSKEY, E. J. (1962). *A Survey of Switching Circuit Theory*, McGraw-Hill

Book reviews

FORTRAN IV in Chemistry, by G. Beech, 1975; 303 pages. (*John Wiley*, £8·75)

Dr. Beech's book is a useful addition to the growing literature on the use of computers in undergraduate science courses. In his introductory chapter he states that users of the book should be familiar with FORTRAN IV and he makes no attempt to give the potted version of the language that wastes so much space in similar works. The reviewer agrees with his statement that many programs published in similar books are so simple that they would be better suited to a desk calculator. Dr. Beech's examples go significantly beyond this but he still limits them to the capacity of a relatively small computer and throughout implies the use of teletype display. There can be few institutions therefore, that could not benefit directly from the examples given, although sophisticates with access to large computers and to graphical display will be somewhat impatient with them. A good feature of the book is the full theoretical background given to all the examples cited.

The chapter on numerical methods gives an adequate survey of techniques such as simple statistics, solution of simultaneous equations, curve fitting, eigenvalue problems and numerical differentiation and integration, most of which are used in later chapters. A chapter follows in which are presented several introductory examples allied to a course in practical chemistry. These should find universal acceptance. The heart of the book is contained in chapters entitled 'Tutorial and dry-lab applications' and 'Theoretical chemistry' in which Dr. Beech and a colleague present seventeen useful programs, with attendant theory, covering topics such as lattice energies, spectrum deconvolution, transition metal spectra, Huckel molecular orbital calculations, etc. The reviewer has written his own version of several of these programs; if he had not he would use Dr. Beech's and save considerable time and effort. All must find some place in a modern chemistry course.

The penultimate chapter is the least satisfactory. In introducing the topic of data acquisition and processing, it is geared solely to the medium of paper tape. The author's justification for this is somewhat ingenious.

The book will be bought and read by those committed to the subject and they will be disappointed it did not go further. The uncommitted who, along with their students, would benefit most from the book, will not get past the title.

J. E. PARKIN (London)

Graph Theory: An Algorithmic Approach, by Nicos Christophides, 1975; 400 pages. (*Academic Press*, £12·50)

There are at least twenty textbooks on graph theory in print. What does a new one offer? Well, the subject is so large that no one book can deal with all of it, so we are treated to a new combination of topics. In this case they seem to have been chosen for their practical applicability. Most welcome are chapters on the location of medians and centres; suitably absent is discussion of enumeration and planarity.

There is an elegant departure from the overworked style of 'Theory-followed-by-applications'. Each new subject is introduced with an informal discussion which gives a hint of the likely applications. This is followed by a brief formal statement of the problem together with appropriate definitions. More detailed discussion of applications occurs only by way of examples.

The algorithms are expressed informally and they are described rather than defined. Proofs are given for some, examples of application for others. I like this method of explanation because it gets you to the core of the ideas quickly. There is detail enough to enable you to write an actual computer program, but not so much that the main theme is obscured.

Each chapter ends with a reasonable set of problems and a bibliography which is thorough but (mercifully) not exhaustive. The book is ideally suited to the engineer or computer scientist who is already solving problems of graph theory and seeks a deeper theoretical knowledge, a range of suggestions and an easy path into the literature. The pure mathematician would find it less satisfactory. Contents include: basic definitions, reachability, set covering, colouring, medians and centres, spanning trees, shortest paths, circuits (Euler graphs and Hamiltonians), network flows, matchings and assignment.

G. WYVILL (Bradford)

Information Retrieval and the Computer, by C. D. Price, 1977; 206 pages. (*MacDonald and Jane's Computer Monographs* No. 26, £5·95)

The author's lectures on document retrieval systems for courses at the University of Lancaster are here presented as a textbook. Though the courses were given to students of computer studies, only a basic knowledge of computer science is assumed so the book is also of potential interest as a text for courses in information science. The main sections of the text are The retrieval process, Documents and their classification, Indexes, Automatic classification, Abstracts and extracts and Automated document retrieval systems.

Once he has established his level of discourse, the author maintains it uniformly to the end. The text provides a general descriptive overview of the field, as seen by someone standing back from the active research fronts. The result is a balanced, readable survey supported by sensible judgements about the several issues still unresolved and by well chosen suggestions for further reading on topics which invite closer scrutiny.

In summing up the results of research on the document retrieval process, the author suggests that great effort has been expended on experimental work only 'to arrive at a surprisingly meagre set of firm conclusions' (page 196). He is right. But some are coming to the conclusion that the general validity of long standing basic concepts such as *relevance* and *precision* must now be questioned. Within the context of work on the well known text collections they may retain their accepted meaning but their application to read operating systems becomes more and more dubious. By the time that this text reaches its deserved second edition, I believe it will be this last section of the book which will demand most of the revision.

B. C. BROOKES (London)