A. AIELLO
E. BURATTINI
A. MASSAROTTI

Reducibility as a tool to extend the power of approximation algorithms the minimization of boolean expressions

RAIRO – Informatique théorique, tome 11, nº 1 (1977), p. 75-82.
<http://www.numdam.org/item?id=ITA_1977__11_1_75_0>

R.A.I.R.O. Informatique théorique/Theoretical Computer Science
(vol. 11, n° 1, 1977, p. 75 à 82)

REDUCIBILITY AS A TOOL TO EXTEND THE POWER OF APPROXIMATION ALGORITHMS THE MINIMIZATION OF BOOLEAN EXPRESSIONS (*)

by A. AIELLO, E. BURATTINI et A. MASSAROTTI (1)
Communicated by M. NIVAT

Summary. — *According to the more recent theoretical results the exact resolutions of a large number of combinatorial problems are to be considered practically unobtainable. In this work starting from Johnson's approach to approximation algorithms we show that some good advantages can be taken from reducibility among combinatorial problems. Minimization of Boolean Expressions is demonstrated to be reducible to Set Covering and consequently an already known approximation algorithm for the latter is applied to the former. In the conclusions a discussion is open about the prospects and limits of the proposed methodology.*

1. INTRODUCTION

The increasing interest towards the development of an exhaustive theory of the complexity of combinatorial problems, that is of a large class of computational problems involving the determination of properties of graphs, integers, finite families of finite sets, boolean expressions, etc., is due to the fact that they schematize practical problems which arise in very important large areas of the social reality: economy, communications, transports, electronics, computer science, and so on.

The first effort made by several authors was trying to classify the problems or to define some hierarchy among them according to their computational complexities. Within this ambit, the more remarkable results due to Cook [1], Karp [3], Meyer and Stockmeyer [4] strongly suggest, although it is not exactly proved, that a very large number of combinatorial problems will remain intractable perpetually.

Therefore it appears sufficiently justified the research of algorithms able to give approximate solutions even because the analysis of the behaviour of these algorithms may introduce some further classification among those problems which the classification of Karp consideres all equally "hard".

(*) Received : May 1976.
(1) Laboratorio di Cibernetica, Arco Felice, Naples, Italie.

2. BASIC CONCEPTS

The concept which plays the leading rôle in the treatment of complexity is that of reducibility among combinatorial problems. The following classical definition of reducibility refers to the problems in terms of recognition of languages :

a language L is reducible to L_0 ($L \propto L_0$) if there exists a polynomial-time-bounded DTM (Deterministic Turing Machine) which converts each string w in the alphabet of L into a string w_0 in the alphabet of L_0 and w is in L if and only if w_0 is in L_0.

A very remarkable result which is relevant to the concept of "reducibility" is given by the well known theorem of Cook [1] :

If $L \in NP$ then $L \propto$ Satisfiability,
which in simple words says that the still open question about the validity of the identity P = NP comes down to ascertain wheather the Satisfiability belongs to P or not.

So the reducibility among problems permits to define a wide equivalence class (the class of NP-Complete Problems) whose elements play the same rôle as Satisfiability in Cook's theorem.

The studies made by Karp and successively those of Sahni [5] and Ullman [7] have more and more enlarged this equivalence class which constitutes the top of complexity for the NP-problems.

The "intractability" of NP, even if it appears demonstrated by a strong circumstantial evidence, is still an open problem. But besides NP-problems there exist some provably "intractable" problems in the hierarchy stated by the computational complexity. On this subject, the result of Meyer and Stockmeyer [4] related to extended regular expressions is remarkable in that it states that the running time of a program for ascertain the emptiness of languages generated by extended regular expressions increase more rapidly than something as

$$f(n) = 2^{2^{2^{\cdot^{\cdot^{\cdot^{2^{2^{2^{n}}}}}}}}}$$

for any finite number of 2's.

3. APPROXIMATE ALGORITHMS

Undoubtedly the main open problem in the theory of the computational complexity is to ascertain whether $P = NP$ is valid or not. This because the very probably negative answer would implies the absolute impossibility of solving all the problems in a wide and important class by the aid of today's computers. For this reason and since it seems that more problems are being found, like that of the extended regular expressions, whose computational complexity is such as to discourage any realistic attempt of giving computer programs for solving them, a certain interest arised about the study of approximate algorithms. An approach particularly good in this direction can be found in Johnson [2]. That work bases itself on the observation that the most combinatorial problems can be formulated as optimization problems and that an approximate algorithm for a problem is defined as a method to yield one approximate solution among those belonging to the set of permitted approximate solutions, when the input is given.

Generally an approximate algorithm will use a certain number of heuristic criteria for choosing which way, among those appearing to each step, seems more likely to lead to the best approximate solution. Sometime it can happen that more ways leading to solutions of different "goodness" can appear equivalent for the heuristic in hand at certain steps during the computation. Therefore, an approximate algorithm must provide some casual mechanism of choice to resort to in the parity cases and this implies that more than one solution may be "choosable" when iterating the computation for the same input. Such solutions can have different "measures" so that an "absolute performance" of an approximate algorithm can be defined either as the average of these measures (average-case analysis) or as the measure of the worst solutions (worst-case analysis). In the following we use the latter kind of measure.

Before we proceed any further we need to recall briefly some parameters already introduced by Johnson which will be adopted by us from now on.

Given an optimization problem p.

u_p^* is the *optimal measure* relative to the input u:

$$u_p^* = BEST\ \{\ m_p(x) : x \in SOL_p(u)\ \},$$

where BEST means MAX in maximization problems and MIN in minimization problems, $\mathrm{SOL}_p(u)$ is the finite set of "approximate solutions" relative to the problem p and the input u, and $m_p(x)$ is a "measure" defined for all possible approximate solutions;

x is an *optimal solution* if and only if

$$x \in SOL_p(u) \qquad and \qquad m_p(x) = u_p^*.$$

$A_p(u)$ is the *performance* of the algorithm A_p for the input u:

$$A_p(u) = WORST\ \{\ m_p(x) : x \in SOL_p(u) \quad and \quad x \text{ is choosable by } A_p \text{ on the input } u\ \}\ ,$$

where WORST is the opposite of BEST.

$r_p(A_p, u)$ is the measure of the *relative* worst case behaviour

$$r(A_p, u) = \begin{cases} u_p^*/A_p(u) & \textit{if } BEST=MAX, \\ A_p(u)/u_p^* & \textit{if } BEST=MIN. \end{cases}$$

$R(A_p)(n)$ is the *overall* worst case behaviour of A_p:

$$R(A_p)(n) = MAX\,\{\, r(A_p, u) : u \in INPUT_p \textit{ and } |u| \le n \,\},$$

where the *problem size* $|u|$ is the number of symbols required to describe u in some standard notation, and INPUT_p is the set of the possible inputs to the problem p.

4. MINIMIZATION OF BOOLEAN EXPRESSIONS (MBE)

The problem called MBE is that of finding the least number of implicants whose disjunction represents a boolean expression equivalent to that assigned. In this paper we will suppose, although a complete generalization can be done, that boolean functions are assigned in their normal disjunctive forms or, equivalently, by giving their "on-set"'s, which are subsets of the hypercube $\{0, 1\}^n$.

Thereafter we will use lower case letters to indicate boolean functions and the corresponding capital's to indicate their on-sets. So if we call f the assigned boolean function, $F \equiv \{P_1, \ldots, P_z\}$ will indicate its on-set; here each P_i is an ordered n-tuple of 0's and 1's. If g_f is an implicant of f, we will have, for the definition of implicants, $G_f \subseteq F$. Any implicant can be represented by an ordered n-tuple of 1's, 0's and -1's where 1's stay for not complemented variables, -1's for those complemented and 0's for those "don't care". MBE can be schematized as follow:

1. $INPUT_{MBE} = \{\, F : F$ *is a finite set* $\{P_1, \ldots, P_z\}$ *of ordered n-tuples of* 0'*s and* 1'*s* $\}$;

2. $SOL_{MBE}(F) = \left\{ X \subseteq \{\, g_f : g_f \textit{ implies } f \,\} : \bigvee_{g_f \in X} g_f \leftrightarrow f \right\}$;

3. $m_{MBE}(X) = |X|$;

4. $BEST = MIN$.

It is immediate to observe that it is sufficient to give an algorithm which, on the same input of MBE, yields as an output the following table of the implicants:

$$T = \{\, (g_f, G_f) : g_f \to f \textit{ and } G_f \equiv \{P_{i_1}, \ldots, P_{i}\} \textit{ is the on-set of } g_f \,\}$$

in a polynomial-bounded running time, to reduce MBE to the following pure Set Covering problem:

1. $INPUT_{SC} = \{ E \equiv \{ S_i \}_{i \ I} : \forall i, \exists (g_f, G_f) \in T$ *such that* $S_i = G_f \}$;
2. $SOL_{SC}(E) = \left\{ E' \subseteq E : \bigcup_{S \in E'} S = \bigcup_{S \in E} S \right\}$;
3. $m_{SC}(E') = |E'|$;
4. $BEST = MIN$.

Since E is a finite family of finite set, the optimization problem which it represents can be solved approximately but satisfactorily by means of the following heuristic Johnson's algorithm AJ:

```
1. Set $SUB = \emptyset$; $UNCOV = \bigcup_{S \in E'} S$; $N = |E|$; $SET(i) = S_i$, $1 \leq i \leq N$;
2. If $UNCOV = \emptyset$ , halt and return $SUB$;
3. Choose $j \leq N$ such that $|SET(j)|$ is maximized;
4. Set $SUB = SUB \cup S_j$; $UNCOV = UNCOV - SET(j)$;
   $SET(i) = SET(i) - SET(j)$, $1 \leq i \leq N$;
5. Go to 2.
```

Let us observe that, according to Johnson, in the worst case the above algorithm requires running time t_J proportional to $N \cdot z \cdot \log N \cdot z$ where N is the number of implicants and from here on $z = |F|$ is the number of points belonging to the on-set F.

It is very simple to get an approximate solution X for MBE out of the output SUB of the algorithm, the table T of the implicants and the observation that

$$X = \{ g_f : (g_f, G_f) \in T \text{ and } \exists S_i \in SUB \text{ such that } G_f = S_i \}$$

is an approximate solution of MBE.

To complete the discussion about the reducibility of MBE to Set Covering we propose the following algorithm AR which gives the table of the implicants T on the input $\text{INPUT}_{\text{MBE}}$ in a running time proportional to z^3.

```
1. Set $VP_i = P_i$, $1 \leq i \leq z$; $S = \lfloor \log_2 z \rfloor$;
$V1 = (\overbrace{1, \ldots, 1}^{n})$; $R = 0$; $VLG(r) = (\overbrace{0, \ldots, 0}^{n})$, $1 \leq r \leq z^2$;
$CG(r, k) = 0$, $1 \leq r \leq z^2$ and $1 \leq k \leq z$;
2. If all the pairs $(i, j)$ were exhausted halt and return the pairs
$[VLG(r); CG(r, 1), \ldots, CG(r, z)]$ for $r = 1, \ldots, R$;
3. Choose a new pair $(i, j) \in \{ 1, \ldots, z \}^2$ (such that $\|VP_i \oplus VP_j\|$ is less
than $S$);
```

4. *Generate all the points VP belonging to the term represented by* $VLG = VP_i + VP_j - V1$*;*

5. *If there exists at least a point VP generated at* 4 *such that* $VP \neq VP_i$ *for each i, go to* 2*;*

6. *Set* $R = R + 1$*;* $VLG(R) = VLG$ *and for each* $k = 1, \ldots, z$ *set* $CG(R, k) = 1$ *if and only if a point VP equal to* VP_k *was generated at* 4*;*

7. *Go to* 2.

Above the prefix V indicates vectors; VLG(r) is the corrispondent of g_f in the previous symbolism and containes the previously described ternary form (ordered n-tuples of 0's, 1's and $-$ 1's) of an implicant of f; the z-tuples $CG(r, 1), \ldots, CG(r, z)$ tell us the on-set G^r_f of g^r_f; $VP_i \oplus VP_j$ is a vector whose coordinates are the sums modulus 2 of the coordinates of P_i and P_j and $\|V\|$ is the arithmetic sum of the coordinates of V. So $\|VP_i \oplus VP_j\|$ represents the Hamming distance between P_i and P_j. Iñ the point 4 of the above algorithm the operations are the usual vectorial ones.

At the point 3, consider the pairs (i, j) ordered as follow:

$$(1, 1), (1, 2), \ldots, (1, N), (2, 1), \ldots, (i, N), (i + 1, 1), \ldots, (N, N),$$

For what concerns the condition between brackets, although it evidently improoves the algorithm, its effects cannot be shown by the worst-case analysis we are adopting. From other side in this approach we have omitted every trick which should have not affected the worst-case performance of the algorithm. Furthermore in the practical application it may be necessary some supplementary instructions, omitted here, which, for example, avoide the generation of the same implicants more times as well as take in account of the prime implicants which contain only one point.

It is very immediate to observe that for a given $z = |F|$ the number of implicants N, produced by the above algorithm, is upperly bounded by the number z^2 of choosable pairs at point 3. At point 4, standing the bound on the distance between VP_i and VP_j, can be generated at most $2^s = 2^{\lfloor \log_2 z \rfloor} \leq z$ points VP. Therefore, taking into account also of point 5, the running time t_R of the given algorithm is upperly bounded by something proportional to z^3.

Now we can give the running time t_{MBE} of the global process, composed by the algorithm of reduction AR and that of Johnson's AJ, and the "goodness" R_{MBE} of the solutions as functions of the size $|\mathrm{INPUT}_{\mathrm{MBE}}| = z$:

$$t_{MBE} = t_R + t_j = O(z^3) + O(z^3 \log z^3) = O(z^3 \log z),$$

this because $t_J = O(N \,.\, z \,.\, \log N \,.\, z)$ *and* $N = O(z^2)$.

For what concerns R_{MBE} it is evident that it cannot grow faster than R_{SC}, that is :

$$R_{MBE}(z) \leq R_{SC}(z \,.\, N) = O(\log |z| \,.\, |N|) = O(\log z),$$

where we use Johnson's result according to which $R_{SC}(n) = O(\log n)$.

5. CONCLUSIONS

The main aim of this paper is to emphasize the fundamental role which reducibility plays in the study of combinatorial problems not only when we consider these in their classical forms but still when we try an approach to the analysis of the approximate algorithms.

Before we gave an example in which for an *NP*-problem an approximate solving process was elaborated by combining a well-known approximate algorithm, created for a quite different problem in the class *NP*-complete, and an algorithm for transforming appropriately the inputs of the given problem. The "goodness" of the solutions given by the global computational process and the "goodness" of the solutions given by the pre-existent algorithm are of the same order. We think that such a metodology may be extended to other cases too. For example, since the Clique is reductible to the Node Cover and since each of them can be transformed into the other by a simple complementation of the graph they refer to, it should not be too difficult to find a satisfactory approximate algorithm for the former, starting from that given by Johnson for the latter.

Nevertheless we must observe that such a methodology has not a quite general applicability. Really it happen that, even if two problems are reducible each other, when we describe them in terms of optimization problems, the two sets of approximate solutions are incompatible. This is the case when for example we consider Graph Coloring and Exact Cover. In fact, Graph Coloring is reducible to Exact Cover but if we apply the approximate algorithm elaborated for the latter to the former, we might have solutions of the Graph Coloring such that, although the number of the colors would be minimized, the guarantee of having for no pair of adjacent nodes the same color could be lost. This because to approximate Exact Cover implies to release the constrain prescribing the disjunction among the sets of the covering given by the approximate solutions.

From there it follows evidently that further deepenings and extensions of this methodology cannot leave a preliminary careful re-examination of the primitive problems out of consideration, as well as the practical consequence implied by manipulations of abstract formulations must be investigated. The consequence of such kind of analysis may yield remarkable upsetting in the hierarchy of computational complexity of combinatorial problems. For example, in the case of Travelling Salesman the suppression of the vinculum which prevents the Salesman from passing more than once through the same node, permits us to find (for the modified problem) a polynomial-time-bounded algorithm (based essentially on the reduction to the Shortest Path i. e. to a *P*-problem) which sometimes gives solutions which are better than those optimal ones relative to the original problem. On the contrary, the suppression of any condition on the length of the itinerary does not change at all the "intractability" of that problem when the previous vinculum is held. This because, after all, neglecting the length of the arcs we obtain the still "intractable" problem of Hamiltonian Cycle. As regards this, it was shown by Sahni *et al.* [6] that really the existence of an approximate algorithm in this case would implies the validity of $P = NP$.

In conclusion we say that further efforts can surely be done in the field of approximate algorithms for NP-problems starting from the available known algorithms and analyzing the mechanisms which in the specific cases permit the reducibility among problems.

REFERENCES

1. S. Cook, *The Complexity of Theorem Proving Procedures*, Proc. 3rd Annual ACM Symposium on Theory of Computing, 1971, pp. 151-158. p. 38-49.
2. D. S. Johnson, *Approximation Algorithms for Combinatorial Problems*, Conference Record of Fifth ACM Symposium on Theory of Computing, 1973, p.
3. R. M. Karp, *Reducibility Among Combinatorial Problems*, In Complexity of Computer Computations, R. E. Miller and J. W. Thatcher Eds and Plenum Press, New York, 1972, pp. 85-104.
4. A. R. Meyer and L. Stockmeyer, *Non Elementary Word Problems in Automata and Logic*, Proc. AMS Symposium on Complexity of Computation, 1973, pp. 38-49.
5. S. Sahni, *Computational Related Problems*, S.I.A.M. J. Comp. **4**, 1974, pp. 62-279.
6. S. Sahni and T. Gonzalès, *P-Complete Problems and Approximate Solutions*, Computer Science Dept., University of Minnesota, 1974.
7. J. D. Ullman, *Polynomial Complete Scheduling Problems*, Proc. 4th Symposium on Operating System Principles, 1973, pp. 96-101.