JOURNAL OF ALGORITHMS 3, 225–244 (1982)

Algorithmic Extremal Problems in Combinatorial Optimization*

KARL J. LIEBERHERR

Princeton University, Department of Electrical Engineering and Computer Science, Princeton, New Jersey 08544

Received April 2, 1981; accepted October 6, 1981

An efficient approximation algorithm generator for the generalized maximum ψ-satisfiability problem is presented which produces an efficient approximation algorithm ψ-*MAXMEAN** for each finite set ψ of relations. The algorithms ψ-*MAXMEAN** are shown to be best-possible in the class of polynomial algorithms (if $P \neq NP$), in both absolute and relative terms. The algorithms are of wide applicability, because of the central position of the generalized maximum satisfiability problem among the class of combinatorial optimization problems.

1. INTRODUCTION

Many combinatorial optimization problems, especially those which are NP-equivalent, are hard to solve exactly. A wealth of heuristics (efficient approximation algorithms) have been proposed to solve these problems approximately. [1] Here a new class of heuristics is analyzed, which is best-possible in a precise sense.

These heuristics perform a "background" optimization, which is based on the following idea: Let m_k be the expected value of the "objective" function for a class M_k of random solutions. $k \in PAR$ is a "natural" parameter of the problem. Pick $k_{\max}$, such that $m_{k_{\max}} = \max_{k \in PAR} m_k$ and find a solution, so that the "objective" function is $\geq m_{k_{\max}}$. We call this method MPR (for: *M*aximize among expected values of *P*arametrized *R*andom solutions).

The MPR method will be discussed in connection with the generalized maximum satisfiability problem, an extension to the one defined in [2].

The main result of this paper establishes a beautiful link between certain mathematical and algorithmic extremal problems. Let ψ be a finite set of

*This research is supported by National Science Foundation grants MCS80-04490 and ENG76-16808.

0196-6774/82/030225-20$02.00/0

relations and let Γ be a class of ψ-formulas. (The definitions are given in Section 2.) Consider the extremal problems:

M1: Which fraction τ_Γ of the clauses in a ψ-formula $S \in \Gamma$ can always be satisfied?

A1: Which fraction τ_Γ of the clauses in a ψ-formula $S \in \Gamma$ can be satisfied in polynomial time?

The main result implies that the solutions to these two problems are identical. Furthermore, the set of ψ-formulas $S \in \Gamma$ which have an assignment satisfying at least the fraction $\tau' > \tau_\Gamma$ of the clauses (τ' *rational*), is shown to be NP-complete. Hence the fundamental constant τ_Γ (an algebraic number, in general) turns out to be a complexity class separator.

The present paper gives a connected and detailed exposition of this theory of algorithmic extremal problems, improving and considerably expanding earlier work. [3–6]

The paper is organized as follows: The definitions are summarized in Section 2, which should be used as a reference section. In Section 3 several main ideas of the paper are used to derive the MPR method. Section 4 contains the main result for the generalized maximum satisfiability problem, where the variables can assume the values 0, 1. Sections 7 and 8 generalize the main result to partitioned formulas and to the maximum satisfiability problem, where the variables can assume the values $0, 1, \ldots, d$ $(d \geq 1)$. Sections 5 and 6 contain the proof of the main result.

In Section 9 the new algorithms are compared to a class of algorithms, a few of which have appeared before in the literature. It is shown that the lower bound for the performance bound of the new algorithms is in general greater than the lower bound on the performance of the old algorithms.

2. Definitions

Generalized Maximum Satisfiability

We start with an introductory example. Let $R(x, y, z)$ be a 3-place logical relation whose truth table is $\{(1,0,0),(0,1,0),(0,0,1)\}$, i.e., $R(x, y, z)$ is true iff exactly one of its three arguments is true. Consider the problem of deciding whether an arbitrary conjunction of clauses of the form $R(x, y, z)$ is satisfiable. Following [2], this problem is called the ONE-IN-THREE SATISFIABILITY problem. For example, the formula $R(b, c, d) \wedge R(b, c, a) \wedge R(a, b, c)$ is satisfiable, because it is made true by assigning the values $0, 0, 1, 0$ to the variables a, b, c, d respectively. The one-in-three satisfiability problem is NP-complete. [2]

The similarity between this problem and the standard satisfiability problem for propositional formulas in conjunctive normal form (CNF) leads to the generalization which is the subject of this paper. Consider the problem of deciding whether a given CNF with three literals in each clause is satisfiable—a well-known NP-complete problem. Since a clause may contain any number of negated variables from 0 to 3, there are four distinct relations. They are defined by $R_0(a, b, c) =$ *a or b or c*, $R_1(a, b, c) =$ *not a or b or c*, $R_2(a, b, c) =$ *not a or not b or c*, $R_3(a, b, c) =$ *not a or not b or not c*. An input to the standard satisfiability problem is just a conjunction of clauses of the form $R_i(a, b, c)$, $i \in \{0, 1, 2, 3\}$.

This sets the stage for the following generalization. Let $\psi = \{R_1, \ldots, R_m\}$ be any finite set of logical relations. A logical relation is defined to be any subset of $\{0,1\}^r$ for some integer $r \geq 1$. The integer r is called the *rank* of the relation. Define a *ψ-formula* to be any sequence of clauses, each of the form $R_i(\zeta_1, \zeta_2, \ldots)$, where $\zeta_1, \zeta_2, \ldots$ are distinct, nonnegated variables whose number matches the rank of R_i, $i \in \{1, \ldots, m\}$. The *ψ-satisfiability problem* is the problem of deciding whether a given ψ-formula is satisfiable. The main result in [2] characterizes the complexity of the ψ-satisfiability problem for every finite set ψ of logical relations. An interesting feature of this characterization is that for any such ψ, the ψ-satisfiability problem is either polynomial-time decidable or NP-complete. The difficulty of approximating the ψ-satisfiability problem is the subject of this paper. The *MAXIMUM ψ-SATISFIABILITY* problem is defined by

Instance: a ψ-formula S.

Question: Find a (0, 1)-assignment to the variables of S which satisfies the maximum number of clauses.

Means

Let ψ be a set of relations and S a ψ-formula with n variables. $mean_{ALL}(S)$ denotes the expected number of satisfied clauses if each variable is assigned 0 or 1 at random, independently of each other and with probability $\frac{1}{2}$. $mean_k(S)$ is the average number of satisfied clauses among all assignments which set exactly k variables to 1. Let $maxmean(S) = \max_{0 \leq k \leq n} mean_k(S)$. Consider a partition of the n variables into the first n_1 variables of type 1 and the next n_2 variables of type $2 (n = n_1 + n_2)$. $mean^{n_1 n_2}_{k_1 k_2}(S)$ is the average number of satisfied clauses among all assignments, for which k_1 of the n_1 variables and k_2 of the n_2 variables are set to 1.

$$maxmean^{n_1, n_2}(S) = \max_{\substack{0 \leq k_1 \leq n_1 \\ 0 \leq k_2 \leq n_2}} mean^{n_1 n_2}_{k_1 k_2}(s).$$

The definition of $mean^{n_1 n_2 n_3 \cdots}_{k_1 k_2 k_3 \cdots}(S)$ and $maxmean^{n_1 n_2 n_3 \cdots}(S)$ is straightforward.

Renamings

The *renaming* of a *variable x with respect to value v* is a substitution of $e(x, v) = (x - v) \bmod 2$ for variable x.

Let J be an assignment for formula S. The *renaming* of *formula S with respect to J* is a substitution of $e(x, J(x))$ for all variables x in S. The resulting formula is called the renamed formula with respect to J.

Let $R(x_1,\ldots,x_n)$ be a relation and let J be an assignment for $x_1,\ldots,x_n$. The *renamed relation R with respect to J* is the relation $L(R, J)$ defined by

$$L(R, J)(e(x_1, J(x_1)),\ldots,e(x_n, J(x_n))) \leftrightarrow R(x_1,\ldots,x_n).$$

By definition

$$R(J(x_1),\ldots,J(x_n)) = L(R, J)(0,\ldots,0).$$

A set of relations ψ is said *to be closed under renaming*, if all relations, which can be generated from relations in ψ by renaming, are in ψ. A set of relations is *closed under restriction*, if all relations, which can be generated from relations in ψ by substituting constants, are in ψ.

Symmetry

Let $SA(S, J)$ be the number of satisfied clauses in formula S under assignment J. Let π_n be the full permutation group on the n variables of S. For $\sigma \in \pi_n$ let $\bar{\sigma}(S)$ be the permuted formula, which is the result of substituting $\sigma(v)$ for all variables v in S. A ψ-formula S is called *symmetric* if any permutation of the variables in the formula returns the same formula up to a permutation of the clauses. If S is a symmetric ψ-formula, then for all permutations σ in π_n and all assignments J of S: $SA(S, J) = SA(\bar{\sigma}(S), J)$ (or equivalently for all permutations σ in π_n and all assignments J of S: $SA(S, J) = SA(S, \sigma^*(J))$, where the assignment $\sigma^*(J)$ is defined by $\sigma^*(J)(v) = J(\sigma^{-1}(v))$ for all variables v in S.)

A symmetric ψ-formula which contains a relation of rank r contains at least $\binom{n}{r}$ clauses. The notion of symmetry is easily generalized, if the variables are partitioned into classes. Let Φ be a partition of the n variables of S into classes. Then S is said to be Φ*–symmetric*, if any permutation of the variables, which preserves the classes, returns the same formula up to a permutation of the clauses.

A logical relation R of rank r is said to be symmetric, if for any permutation $\sigma \in \pi_r$ of r variables: $R(\zeta_1, \zeta_2,\ldots,\zeta_r)$ iff $R(\sigma(\zeta_1), \sigma(\zeta_2),\ldots,\sigma(\zeta_r))$.

Complexity

Let ψ be a set of relations. $cl(S)$ denotes the number of clauses in a ψ-formula S. A rational number $C(0 \le C \le 1)$ is said to be a *(relative)P-optimal threshold with respect to* ψ, if (1) there is a polynomial algorithm Q_1 satisfying at least $C \cdot cl(S)$ clauses for any $S \in \{\psi\text{-}formulas\}$ and (2) the set of ψ-formulas having an assignment satisfying at least $C' \cdot cl(S)$ clauses is NP-complete for any rational $C' > C$ $(C' \le 1)$. Algorithm Q_1 is also called relative P-optimal.

A polynomial time computable function q: $\{\psi\text{-}formulas\} \to \{$rational numbers$\}$ is said to be *absolute P-optimal* with respect to ψ, if (1) there is a polynomial algorithm Q_2 satisfying at least $q(S)$ clauses for any $S \in \psi$-*formulas* and (2) The set of ψ-formulas S, which have an assignment satisfying more than $q(S)$ clauses is NP-complete.

Of course, there are many trivial absolute P-optimal functions, but we consider interesting performance bounds defined by closed form expressions.

A decision, search or optimization problem Q is *NP-equivalent*, whenever it can be shown that a polynomial algorithm exists for Q, iff P = NP (by Turing reductions). (A NP-equivalent decision problem is by definition NP-complete with respect to Turing reductions.)

Relations

$R^{\$}_{ijb}(x_1, x_2,\ldots,x_n) \Leftrightarrow \Sigma_{l=1}^{n} a_l x_l \$ b$ where $\$ \in \{\ge, \le, =\}$ and $a_l \in \{-1, 0, 1\}$ and j coefficients $a_l (1 \le l \le n)$ are ± 1 and the first i coefficients are $+1$.

Assume that the variables of ψ-formula S are partitioned into two classes. Then

$$R^{\$}_{i_1 j_1 i_2 j_2 b}(x_1, x_2,\ldots,x_n) \Leftrightarrow \sum_{l_1=1}^{n_1} a_{l_1} x_{l_1} + \sum_{l_2=1}^{n_2} a_{l_2} x_{l_2} \$ b,$$

where $n_1 + n_2 = n$ and the variables $\{x_{l_1} \mid 1 \le l_1 \le n_1\}$ are in class 1 and the variables $\{x_{l_2} \mid 1 \le l_2 \le n_2\}$ are in class 2. j_1 coefficients $a_{l_1}(1 \le l_1 \le n_1)$ are ± 1 with the first i_1 of them $= +1$, and similar, j_2 coefficients $a_{l_2}(1 \le l_2 \le n_2)$ are ± 1 with the first i_2 of them $= +1$.

Further Notation

$SA(S, J)$ is the number of satisfied clauses in formula S under assignment J. $S_{x=v} (v \in \{0, 1\})$ denotes the ψ-formula which is obtained from S after substituting v for x. Note that $S_{x=v}$ might have clauses containing relations (even of rank 0) which are always satisfied or never satisfied. $J_{ALL\,0}$ is the assignment which assigns 0 to all variables.

3. Derivation of Method

Let ψ be a finite set of logical relations. Let $\Gamma \subseteq \{\psi\text{-formulas}\}$ and assume that Γ is closed under symmetrization, i.e. $\mathrm{SYM}(\Gamma) \subseteq \Gamma$, where $\mathrm{SYM}(\Gamma)$ is the class of formulas which are obtained by symmetrizing the formulas $S \in \Gamma$ with the full permutation group on the number of variables of S. The symmetrization process is only applied to nonsymmetric formulas. Once a formula is symmetric, it is not symmetrized again. The computation of

$$\tau_\Gamma = \inf_{S \in \Gamma} \max_{\substack{\text{all assignments}\\ J \text{ for } S}} \frac{SA(S, J)}{cl(S)}$$

leads in a natural way to the MPR method.

Lemma 1. *If $SYM(\Gamma) \subseteq \Gamma$ then*

$$\tau_\Gamma = \inf_{S \in SYM(\Gamma)} \max_{\substack{\text{all assignments}\\ J \text{ for } S}} \frac{SA(S, J)}{cl(S)}.$$

Proof. Let S be a ψ-formula. Symmetrize S by using the full permutation group on the n variables of S. The resulting symmetric formula with $n! \cdot cl(S)$ clauses is called S^*. If S^* has an assignment satisfying the fraction g of the clauses, then S has an assignment satisfying at least the fraction g of the clauses. This is implied by the fact that, if the average of a set of numbers is g, then at least one number is $\geq g$. Therefore it is sufficient to minimize among the symmetric formulas in order to compute τ_Γ. □

Define $mean_k(S)$ to be the average number of satisfied clauses among all assignments which set exactly k variables to 1.

Lemma 2. *If S is a symmetric ψ-formula then*

$$\max_{\substack{\text{all assignments}\\ J \text{ for } S}} SA(S, J) = \max_{0 \leq k \leq n} mean_k(S).$$

Proof. Since S is symmetric, we have $SA(S, J) = mean_k(S)$, if J assigns 1 to exactly k variables. □

Let $\psi = \{R_1, R_2, \ldots, R_m\}$ and let $t_{R_i} (1 \leq i \leq m)$ be the number of clauses containing relation R_i in a given formula.

Lemma 3. *Let S be a ψ-formula with n variables.* $\mathrm{mean}_\lambda(S)$ *is a polynomial in k, the coefficients of which are functions of n and $t_{R_i} (1 \leq i \leq m)$ which are linear in $t_{R_i} (1 \leq i \leq m)$. The degree of the polynomial is bounded by the highest rank of a relation in ψ.*

Proof. By elementary combinatorial analysis

$$mean_k(S) = \sum_{\substack{\text{all relations}\\ R \in S}} t_R SAT_k^n(R),$$

where

$$SAT_k^n(R) = \frac{\displaystyle\sum_{s=0}^{r(R)} \frac{q_s(R)}{\binom{r(R)}{s}} \cdot \binom{k}{s} \cdot \binom{n-k}{r(R)-s}}{\binom{n}{r(R)}},$$

where t_R is the number of clauses containing relation R, $r(R)$ is the rank of relation R, $q_s(R)$ is the number of satisfying rows in the truth table of R which contain s ones. □

The above three lemmas imply that

$$\tau_\Gamma = \lim_{n \to \infty} \inf_{t_{R_i} (1 \leq i \leq m)} \max_{0 \leq k \leq n} \frac{\sum_{i=1}^{m} t_{R_i} \mathrm{SAT}_k^n(R_i)}{\sum_{i=1}^{m} t_{R_i}}.$$

This problem is considerably simpler and for many sets of relations ψ and classes Γ it can be solved explicitly. Examples are given in [5, 12, 13].

Note that the MPR method is naturally implied. The parameter k is the number of variables which are set to one.

4. Main Result

It turns out, that for the generalized maximum satisfiability problem the MPR method is best possible in a sense which is made precise by the following theorem.

Theorem 1. *Let $maxmean(S) = \max_{0 \leq k \leq n} mean_k(S)$ and assume that Γ is not "trivial", i.e. the maximum ψ-satisfiability problem for ψ-formulas in Γ is NP-equivalent.*

1.1 *If $SYM(\Gamma) \subseteq \Gamma$ then there is a polynomial algorithm* MAXMEAN, *which satisfies the fraction*

$$\tau_\Gamma = \inf_{S \in \Gamma} \max_{\substack{\text{all assignments}\\ J \text{ of } S}} \frac{SA(S, J)}{cl(S)}$$

of the clauses for a given ψ-formula $S \in \Gamma$.

1.2 *If (a) $SYM(\Gamma) \subseteq \Gamma$, and (b) the ψ-satisfiability problem for the formulas in Γ is NP-complete, and (c) Γ is closed under concatenation of formulas with disjoint variables, then: For any rational $\tau' > \tau_\Gamma$ the set of ψ-formulas in Γ which have an assignment satisfying the fraction τ' of the clauses is NP-complete.*

1.3 *Algorithm* MAXMEAN *satisfies at least $maxmean(S)$ clauses for $S \in \Gamma$.*

1.4 *If Γ is closed under renaming, then the problem of finding an assignment that satisfies $> maxmean(S)$ clauses for $S \in \Gamma$, is NP-equivalent.*

1.5 *The polynomial algorithm* MAXMEAN *guaranteed by* 1.1 *can be computed in polynomial time.*

5. Algorithms

First we prepare some tools for deriving the efficient algorithm MAXMEAN. The basic idea is to derive a recurrence relation for $mean_k(S)$.

Lemma 4. *If $k \neq 0$, then*

$$mean_k(S) = \frac{k}{n} mean_{k-1}(S_{x=1}) + \frac{n-k}{n} mean_k(S_{x=0}).$$
$$mean_0(S) = mean_0(S_{x=0}).$$

$S_{x=\gamma}(\gamma = 0, 1)$ denotes the ψ-formula which is obtained from S after substituting γ for x.

Proof. Consider the symmetrized formula S^* corresponding to S, which is obtained by using the full permutation group. This formula contains $n!$ copies of S, which are divided into $k \cdot (n-1)!$ copies of $S_{x=1}$ and $(n-k)(n-1)!$ copies of $S_{x=0}$. Since $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ and $\binom{n-1}{k-1} / \binom{n}{k} = k/n$ and $\binom{n-1}{k} / \binom{n}{k} = (n-k)/n$,

$$n!mean_k(S) = k \cdot (n-1)!mean_{k-1}(S_{x=1}) + (n-k)(n-1)!mean_k(S_{x=0}).$$

This implies the above recurrence relation. □

Consider the following algorithm MEAN:

Input: ψ-formula S, integer $k (0 \le k \le n)$.

Output: Assignment which satisfies at least $mean_k(S)$ clauses. The number of variables to which 1 is assigned might be $< k$.

for all variables x in S **do**
  if $mean_{k-1}(S_{x=1}) > mean_k(S_{x=0})$
  then $x := 1$; $k := k-1$; $S := S_{x=1}$
  else $x := 0$; $S := S_{x=0}$
  ($mean_{-1}(S)$ is defined to be zero)

The proof of correctness is implied by the above recurrence relation. The algorithm requires, that $mean_k(S)$ is expressed for ψ^1-formulas, where ψ^1 is the set ψ closed under restriction.

Consider the following algorithm MAXMEAN:

Input: ψ-formula S.

Output: Assignment which satisfies at least $maxmean(S)$ clauses.

1. Find $k (0 \le k \le n)$, so that $maxmean(S) = mean_k(S)$ by a linear search.
2. Apply algorithm MEAN to S and k to find an assignment satisfying at least $maxmean(S)$ clauses.

This algorithm is certainly polynomial in the length $|S|$ of S. It can be made more efficient, if $mean_k(S)$ is of small degree compared to the number of variables in S. Step 1 will be executed faster, if the following method is used. (1) Determine the derivative p of $mean_k(S)$ and approximate all roots $r_1, r_2, \ldots$ of p sufficiently. (2) Determine the maximum value of $mean_k(S)$ for

$$k = 0, \lfloor r_1 \rfloor, \lceil r_1 \rceil, \lfloor r_2 \rfloor, \lceil r_2 \rceil, \ldots, n.$$

If the Galois group of p is not solvable, we can use Sturm's theorem combined with binary search. Sturm's theorem yields a subroutine which returns the number of zeros in a given interval.

Sturm's Theorem

Let p be a real polynomial, p not identical to zero, and let $p_0, p_1, \ldots, p_m$ be the sequence of polynomials generated by the Euclidean algorithm started with $p_0 = p$, $p_1 = p'$:

$$p_0(x) = q_1(x)p_1(x) - p_2(x),$$
$$p_1(x) = q_2(x)p_2(x) - p_3(x),$$
$$p_{k-1}(x) = q_k(x)p_k(x) - p_{k+1}(x),$$
$$p_{m-1}(x) = q_m(x)p_m(x).$$

Then for any real interval $[\alpha, \beta]$, such that $p(\alpha) \cdot p(\beta) \neq 0$, p has exactly $\vee(\alpha) - \vee(\beta)$ distinct zeros in $[\alpha, \beta]$, where $\vee(x)$ denotes the number of sign changes in the sequence $\{p_i(x), 0 \le i \le m\}$.

Proof. See, e.g., [7, p. 449]. It is well known that the regular Euclidean algorithm might lead to exponential coefficient growth. However the work of Cayley [8] and Collins [9] shows that this exponential growth can be avoided. Heindel [10] shows that the real zeros of a polynomial can be found in polynomial time in -lg(ϵ) and in the size of the polynomial, where ϵ is the allowed maximal error. Akritas [14] proposes faster algorithms.

The generation of MAXMEAN from the truth tables of the relations in ψ is implied by the proof of Lemma 3. Of course, this generation can be done in polynomial time.

6. NP-Equivalence of Improving

The key to the proof of part 1.4 of the theorem is this question: Given a formula S, how difficult is it to find a renaming R, so that

$$maxmean(R(S)) = mean_0(R(S))?$$

A *renaming* of a *variable* x is a substitution of $1 - x$ for x. A *renaming* R of a *formula* is a renaming of some of its variables. The above problem is solved efficiently by algorithm MAXMEAN*:

Input: ψ-formula S.

Output: (a) Renaming R of S, so that (1) $maxmean(R(S)) = mean_0(R(S))$, (2) $maxmean(R(S)) \geq maxmean(S)$. (b) Interpretation J satisfying $SA(S, J) = maxmean(R(S))$.

loop

(1) Apply algorithm MAXMEAN to S, which returns an assignment J satisfying at least $maxmean(S)$ clauses. (2) If assignment J is no improvement over the previous assignment then exit. (3) Rename S, so that the assignment $J_{All\ 0}$, which assigns 0 to all variables, corresponds to assignment J.

end

The renaming R is the composition of all renamings performed in the loop. J is the interpretation which corresponds to $J_{ALL\ 0}$ after renaming all variables which are renamed by R. This loop might be executed several times, since the polynomials $mean_k(S)$ and $mean_k(R_1(S))$ might be very different for a given renaming R_1.

To prove Theorem 1.4, assume that there is polynomial algorithm Ω_1, which returns an assignment satisfying more than $maxmean(S)$ clauses (if such an assignment exists). Then the following algorithm RED_1 is polynomial for the maximum ψ-satisfiability problem for Γ.

Algorithm RED_1

loop

1. Apply MAXMEAN* to S; it returns a renaming R, so that $maxmean(R(S)) = SA(R(S), J_{ALL\ 0})$; Let $S: = R(S)$;
2. Apply Ω_1: **if** the assignment $\Omega_1(S)$ does not satisfy more than $maxmean(S)$ clauses **then** exit;
3. Rename S, so that the assignment $J_{ALL\ 0}$ corresponds to the assignment $\Omega_1(S)$.

end

This loop is executed at most $cl(S)$ times.

The proof of Theorem 1.2 is a straightforward generalization of the NP-completeness proof for the 2-satisfiability problem given in [5].

We give a polynomial transformation Δ, which transforms a ψ-formula $S \in \Gamma$ to a ψ-formula $\Delta(S) \in \Gamma$, so that S is satisfiable, iff $\Delta(S)$ has an assignment satisfying at least the fraction $c' > \tau$ of the clauses ($c' = p/q$ rational). The definition of τ and the general expression for $mean_k(S)$ guarantee the existence of a symmetric formula S_ω, for which only the fraction $c_2 < c'$ can be satisfied.

Let S_ω contain m_1 clauses, of which only m_2 can be satisfied.[1] Let S be a satisfiable ψ-formula containing m clauses. $\Delta(S)$ consists of z_1 copies of S and z_2 copies of S_ω, so that the following conditions hold: If $f(r_1, z_1, z_2) = (r \cdot z_1 + m_2 z_2)/(m \cdot z_1 + m_1 z_2)$ then (1) $f(m - 1, z_1, z_2) < c' = p/q$, (2) $f(m, z_1, z_2) = c' = p/q$.

It is straightforward to check that both conditions hold, if $z_1 = m_1 p - m_2 q$ and $z_2 = m(q - p)$ and $c'm < m - 1$. The latter inequality may be assumed without loss of generality.

Note that for this reduction it might be important, that a formula can contain multiple clauses, since the formula S_ω might necessarily contain multiple clauses.

7. Generalization to Partitioned Formulas

Theorem 1 allows a natural generalization. Instead of considering ψ-formulas S we look at partitioned ψ-formulas, which have an additional structure given by a partition of the variables. For simplicity of notation we only consider partitions into two sets; the generalization to several sets (a constant number) is straightforward. There are at least two motivations to

[1] S_ω can be found in finite time, since the formulas in Γ are countable.

study partitioned formulas: (I) It is possible that

$$\inf_{S\in\Gamma} \max_{\substack{\text{all assignments}\\ J \text{ of } S}} \frac{SA(S,J)}{cl(S)} > \inf_{S\in SYM(\Gamma)} \max_{\substack{\text{all assignments}\\ J \text{ of } S}} \frac{SA(S,J)}{cl(S)}$$

if not $SYM(\Gamma) \subseteq \Gamma$. (II) The approximation behavior of the maximum ψ-satisfiability problem is open. This sets the stage for the generalization of theorem 1 to partitioned formulas.

Terminology: $mean_k(S)$ is replaced by $mean^{n_1n_2}_{k_1k_2}(S)$, where $n_1 + n_2$ is the number of variables in S. $mean^{n_1n_2}_{k_1k_2}(S)$ is the average number of satisfied clauses among all assignments which set k_1 of the first n_1 variables and k_2 of the next n_2 variables to 1. Let Γ_p be a set of partitioned ψ-formulas. $SYM_p(\Gamma_p)$ denotes the class of "partially" symmetrized formulas in Γ_p. The symmetrization is done with permutations preserving the partition of the variables of a given formula.

If Γ_p is a set of partitioned ψ-formulas, we denote with Γ the corresponding set of ψ-formulas without the partition.

Define

$$maxmean^{n_1n_2}(S) = \max_{\substack{0\le k_1\le n\\ 0\le k_2 \le n}} mean^{n_1n_2}_{k_1k_2}(S).$$

THEOREM 2. *Assume that Γ_p is not "trivial", i.e. the maximum ψ-satisfiability problem for ψ-formulas in Γ_p is NP-equivalent.*

2.1 *If $SYM_p(\Gamma_p) \subseteq \Gamma_p$, then there is a polynomial algorithm $MAXMEAN^{n_1n_2}$, which satisfies the fraction*

$$\tau_{\Gamma_p} = \inf_{S\in\Gamma_p} \max_{\substack{\text{all assignments}\\ J \text{ of } S}} \frac{SA(S,J)}{cl(S)}$$

of the clauses for a partitioned ψ-formula $S \in \Gamma_p$.

2.2 *If (a) $SYM_p(\Gamma_p) \subseteq \Gamma_p$, and (b) the ψ-satisfiability problem for the formulas in Γ_p is NP-complete, and if (c) Γ is closed under concatenation of formulas with disjoint variables, then: The set of ψ-formulas in Γ_p which have an assignment satisfying the fraction τ' of the clauses is NP-complete for any rational $\tau' > \tau_{\Gamma_p}$.*

2.3 *Algorithm $MAXMEAN^{n_1n_2}$ satisfies at least $maxmean^{n_1n_2}(S)$ clauses for $S \in \Gamma_p$.*

2.4 *If Γ is closed under renaming, then the problem of finding an assignment that satisfies $>$ $maxmean^{n_1n_2}(S)$ clauses for $S \in \Gamma_p$, is NP-equivalent.*

2.5 *The polynomial algorithm $MAXMEAN^{n_1n_2}$ guaranteed by 2.1 can be computed in polynomial time.*

The proof of Theorem 2 is similar to the proof of Theorem 1. However, there are some interesting differences, which are pointed out in the following.

1. Algorithm MEAN (for partitioned formulas)

Input: ψ-formula S with a partition of its $n_1 + n_2$ variables into 2 classes, the first n_1 are in class 1, the next n_2 in class 2. Integers k_1, k_2 $(0 \le k_1 \le n_1$, $0 \le k_2 \le n_2)$

Output: Assignment which satisfies at least $mean^{n_1n_2}_{k_1k_2}(S)$ clauses.

for all variables x in S in class 1 **do**
  if $mean^{n_1-1.n_2}_{k_1-1.k_2}(S_{x=1}) > mean^{n_1-1.n_2}_{k_1.k_2}(S_{x=0})$
  then $x := 1$; $k_1 := k_1 - 1$; $S := S_{x=1}$
  else $x := 0$; $S := S_{x=0}$
for all variables y in S in class 2 **do**
  if $mean^{n_2-1}_{k_2-1}(S_{y=1}) > mean^{n_2-1}_{k_2}(S_{y=0})$
  then y$:= 1$; $k_2 := k_2 - 1$; $S := S_{y=1}$
  else $y := 0$; $S := S_{y=0}$
($mean^{n_1.n_2}_{-1.k_2}(S)$ and $mean^{n_1.n_2}_{k_1.-1}(S)$ are defined to be zero)

2. Computation of $mean^{n_1n_2}_{k_1k_2}(S)$ from the truth tables.

$$mean^{n_1n_2}_{k_1k_2}(S) = \sum_{\substack{\text{all relations}\\ R\in S}} t_R SAT^{n_1n_2}_{k_1k_2}(R),$$

where

$$SAT^{n_1n_2}_{k_1k_2}(R) = \frac{\displaystyle\sum_{s_1=0}^{r_1(R)} \sum_{s_2=0}^{r_2(R)} \frac{q_{s_1,s_2}}{\prod_{j=1}^{2}\binom{r_j(R)}{s_j}} \prod_{j=1}^{2}\binom{k_j}{s_j}\binom{n_j-k_j}{r_j(R)-s_j}}{\displaystyle\prod_{j=1}^{2}\binom{n_j}{r_j(R)}},$$

where $r_j(R)$ is the rank of relation R in class j, $q_{s_1,s_2}(R)$ is the number of satisfied rows in the partitioned truth table of R which contain exactly s_1 ones in class 1 and s_2 ones in class 2, n_j is the number of variables in class j, k_j is the number of variables in class j which are set to 1.

These formulas imply that algorithm $MAXMEAN^{n_1n_2}$ can be generated in polynomial time.

3. The determination of the optimal k_1, k_2 with classical methods is in this case more complicated. Of course, the optimal pair could be determined

by searching all relevant pairs, but if the degree of $mean^{n_1 n_2}_{k_1 k_2}(S)$ is small compared to the number of variables in S, then there are considerably faster methods.

Partitioning in the limit

By partitioning the variables of a given formula into smaller and smaller sets, we get better and better approximations and finally the optimal assignment. Of course, the running time increases as the partitions get finer and finer. A parallel implementation of $\log_2(n)$ algorithms (n = number of variables) of type $MAXMEAN^{n_1 n_2, \ldots, n_t^*}$ seems to exploit in an optimal way the fact, that already the solution found with a coarse partition might be optimal.

Let S be a ψ-formula with $n = 2^q$ variables. Let the n variables be partitioned into $t = 2^r$ sets of equal size $n/2^r (r \le \log_2 n = q)$. The complexity to find $maxmean^{n_1 n_2 \cdots n_t}(S)$ is bounded by $(n/t + 1)^t \cdot 0(|S|^c) < e^n \cdot 0(|S|^c)$ for some constant c. This bound even holds, if it is allowed to check all possible values $mean^{n_1 n_2 \cdots n_t}_{k_1 k_2 \cdots k_t}(S)$. The algorithms $MAXMEAN^*$, $MAXMEAN^{n_1 n_2^*}$, $MAXMEAN^{n_1 n_2 n_3 n_4^*}, \ldots$ can be expected to find better and better approximations and $MAXMEAN^{n_1 n_2 \cdots n_n^*}$ finds the maximal assignment, since $n_i = 1$ for $0 \le i \le n$.

It would be interesting to know for the sequence of $\log(n) = q$ algorithms $MAXMEAN^*$, $MAXMEAN^{n_1 n_2^*}, \ldots, MAXMEAN^{n_1 \cdots n_q^*}$, which one usually finds the optimum first. For symmetric formulas obviously the first algorithm terminates first.

8. Generalization to Maximum ψ_d-Satisfiability

Let $\psi_d = \{R_1, \ldots, R_m\}$ be any finite set of relations on subsets of $\{0, 1, \ldots, d\}^*$. A relation R of rank $r(R)$ is an ordered subset of $\{0, 1, \ldots, d\}^r$. A ψ_d-formula is a sequence of clauses each of the form $R_i(x_1, x_2, \ldots)$, where $x_1, x_2, \ldots$ are variables whose number matches the rank of R_i, $\{i \in 1, \ldots, m\}$. The maximum ψ_d-satisfiability problem is defined by:

Instance: A ψ_d-formula S.

Question: Find an assignment to the $(d+1)$-valued variables of S that satisfies the maximum number of clauses.

Theorem 1 directly carries over to the maximum ψ_d-satisfiability problem.

THEOREM 3. *For all constants d and for any finite set ψ_d of relations on $\{0, 1, \ldots, d\}^*$ and for any nontrivial set Γ of ψ_d-formulas the following holds*

3.1 *If $SYM(\Gamma) \subseteq \Gamma$, then there is a polynomial algorithm* MAXMEAN *that satisfies the fraction*

$$\tau_\Gamma = \inf_{S \in \Gamma} \max_{\substack{\text{all assignments} \\ J \text{ of } S}} \frac{SA(S, J)}{cl(S)}$$

of the clauses for a given ψ_d-formula $S \in \Gamma$.

3.2 *If (a) $SYM(\Gamma) \subseteq \Gamma$, and (b) the ψ_d-satisfiability problem for the formulas in Γ is NP-complete, and if (c) Γ is closed under concatenation of formulas with disjoint variables, then for any rational $\tau' > \tau_\Gamma$ the set of ψ_d-formulas in Γ which have an assignment satisfying the fraction τ' of the clauses is NP-complete.*

3.3 *Algorithm* MAXMEAN *satisfies at least maxmean(S) clauses for $S \in \Gamma$.*

3.4 *If Γ is closed under renaming, then the problem of finding an assignment that satisfies $>$ maxmean(S) clauses for $S \in \Gamma$, is NP-equivalent.*

3.5 *The polynomial algorithm* MAXMEAN *guaranteed by* 1.1 *can be computed in polynomial time.*

Again the proof of Theorem 3 is similar to the proof of Theorem 1. The interesting differences are pointed out in the following. For simplicity of notation we assume $d = 3$. The generalization is straightforward.

1. Algorithm MEAN

Input: ψ_3-formula S with n variables, integers $k_0, k_1, k_2 (k_0 + k_1 + k_2 = n)$.

Output: Assignment of 0, 1 or 2 to the variables of S, so that at least $mean^n_{k_0 k_1 k_2}(S)$ clauses are satisfied.

for all variables x in S **do**
begin
let $j \in \{0, 1, 2\}$ be the number of the element in the list $(mean^{n-1}_{k_0-1, k_1, k_2}(S_{x=0}), mean^{n-1}_{k_0, k_1-1, k_2}(S_{x=1}), mean^{n-1}_{k_0, k_1, k_2-1}(S_{x=2}))$ which is maximal in this list.
$x := j$; $S := S_{x=j}$;
end

The correctness of this algorithm is based on the recurrence relation

$$\begin{aligned} mean^n_{k_0, k_1, k_2}(S) = {} & \frac{k_0}{n} mean^{n-1}_{k_0-1, k_1, k_2}(S_{x=0}) \\ & + \frac{k_1}{n} mean^{n-1}_{k_0, k_1-1, k_2}(S_{x=1}) \\ & + \frac{k_2}{n} mean^{n-1}_{k_0, k_1, k_2-1}(S_{x=2}). \end{aligned}$$

2. Computation of $mean^n_{k_0k_1k_2}$ from the truth tables ($d = 2$)

$$mean^n_{k_0k_1k_2}(S) = \sum_{\substack{\text{all relations}\\ R\in S}} t_R SAT^n_{k_0k_1k_2}(R),$$

where

$$SAT^n_{k_0k_1k_2}(R) = \frac{\displaystyle\sum_{0\le s_0+s_1\le r(R)} \frac{q_{s_0s_1}(R)}{\binom{r(R)}{s_0}\cdot\binom{r(R)-s_0}{s_1}}\cdot\binom{k_0}{s_0}\cdot\binom{k_1}{s_1}\cdot\binom{n-k_0-k_1}{r(R)-s_0-s_1}}{\binom{n}{r(R)}},$$

where k_ν is the number of variables set to $\nu(\nu = 0 \cdots d-1, k_d = n - k_0 - k_1 - \cdots - k_{d-1})$, $q_{s_0s_1\cdots s_{d-1}}(R)$ is the number of rows in the truth table of R which contain exactly s_ν times number ν ($\nu = 0 \ldots d-1, s_d = r(R) - s_0 - s_1 - \cdots - s_{d-1}$).

These formulas imply that algorithm MAXMEAN can be generated in polynomial time (for constant d).

3. The computation of the optimal k_0, k_1, k_2 for a given formula S can be done with methods similar to those used for partitioned formulas. The brute force approach to compute the optimal $k_0, k_1, k_2, \ldots, k_d$ gets very expensive as d grows. Let $p(n, k)$ be the number of partitions of n into exactly k summands. Note that $p(n, k) \ge (1/k!)\binom{n-1}{k-1}$ and that the number of evaluations of $mean_{k_0k_1,\ldots,k_d}(S)$ is $\ge p(n, k)$, if the brute force approach is used to compute $maxmean(S)$.

In some special cases (e.g. for approximate graph coloring) the optimal $k_0, k_1, \ldots, k_d$ can be computed analytically.

Another way to avoid this problem is to translate the ψ_d-satisfiability problem to a ψ_1-satisfiability problem. This is very easy, if $d + 1$ is a power of 2. However such a translation looses a part of the structure and it might be that maxmean (translated ψ_1-formula) $<$ maxmean(original ψ_d-formula). However, $mean_{ALL}$(translated ψ_1-formula) $= mean_{ALL}$(original ψ_d-formula).

4. The generalization of MAXMEAN* to the maximum ψ_d-satisfiability problem requires a generalization of renamings. Let ψ_d be a finite set of relations, i.e. a finite collection of subsets of $\{0, 1, \ldots, d\}^*$. A *renaming* of a variable of value ν is a substitution of $e_d(x, \nu) = (x - \nu) mod(d + 1)$ for variable x. If $d > 1$, then linear inequalities are not mapped into linear inequalities under such renamings.

9. Comparisons and Summary

Let $mean_{ALL}(S)$ be the average number of satisfied clauses among all $(d + 1)^n$ assignments of a ψ_d-formula S with n variables. How can we efficiently find an assignment satisfying at least $mean_{ALL}(S)$ clauses for a given ψ_d-formula S? One could try a randomized algorithm which generates random assignments, but it is not clear how many assignments are needed until we get one which satisfies at least $mean_{ALL}(S)$ clauses. The following algorithm MEANALL is deterministic and fast. For special sets ψ_d of relations algorithm MEANALL has already appeared in the literature (e.g., in [4] for the satisfiability problem or in [11] for the graph coloring problem).

Algorithm MEANALL
Input: A ψ_d-formula S containing n variables.
Output: An assignment j which satisfies $mean_{ALL}(S)$ clauses.
for all variables x in S **do**
begin
Compute $j_{\max}$ such that
$$\max_{0\le j\le d} mean_{ALL}(S_{x=j}) = mean_{ALL}(S_{x=j_{\max}});$$
$x := j_{\max}$; $S := S_{x=j\max}$
end

Remarks: (1) The correctness of this algorithm is based on the recurrence relation

$$mean_{ALL}(S) = \frac{1}{d+1}(mean_{ALL}(S_{x=0}) + \cdots + mean_{ALL}(S_{x=d})).$$

(2) For a ψ_d-formula S:

$$mean_{ALL}(S) = \sum_{\substack{\text{all relations}\\ R\in S}} t_R m_R (d+1)^{-r(R)},$$

where m_R is the number of satisfying rows in the truth table of R. (3) For any renaming R algorithm MEANALL shows the same behavior on S and $R(S)$.

Comparison of MEANALL/MAXMEAN

We can expect that most of the time MAXMEAN finds a better assignment than MEANALL. For some ψ, $mean_{ALL}(S)$ is not an absolute P-optimal performance bound in the sense that $\lceil mean_{ALL}(S)\rceil + 1$ clauses can be satisfied in polynomial time. What can be easily proven is that

$maxmean(S) \geq mean_{ALL}(S)$. This follows from

$$mean_{ALL}(S) = \sum_{k_0+k_1+\cdots+k_d=n} \frac{n!}{k_0!k_1!\cdots k_d!} mean_{k_0k_1,\ldots,k_d}(S)$$

Equality holds, iff $mean_{k_0k_1\cdots k_d}(S)$ is independent of $k_0, k_1, \ldots, k_d$. The formulas for which $mean_{k_0k_1\cdots k_d}(S)$ is independent of $k_0, k_1 \cdots k_d$ are rare. Hence algorithm MAXMEAN usually guarantees more than algorithm MEANALL, if the performance bound is expressed as a rational number ($maxmean(S)$ and $mean_{ALL}(S)$). Of course $\lceil maxmean(S) \rceil = \lceil mean_{ALL}(S) \rceil$ is possible, although $maxmean(S) > mean_{ALL}(S)$. For some sets Γ of ψ_d-formulas algorithm MEANALL satisfies at least the fraction

$$\tau = \inf_{S \in \Gamma} \max_{\substack{\text{all assignments} \\ J \text{ of } S}} \frac{SA(S, J)}{cl(S)}$$

of the clauses. This happens e.g. with the approximate graph coloring problem with $d + 1$ colors. In this case $\tau = mean_{ALL}cl(5) = d/(d + 1)$.

As a rule of thumb we have

$$\frac{mean_{ALL}}{cl(S)} = \inf_{S \in \Gamma} \max_{\substack{\text{all assignments} \\ J \text{ of } S}} \frac{SA(S, J)}{cl(S)}$$

if the maximum of $mean_{k_0k_1\cdots k_d}(S)$ is achieved, if all $k_j (0 \leq j \leq d)$ have about the same size.

The comparison of MEANALL and MAXMEAN can be carried over to compare MAXMEAN and $MAXMEAN^{n_1n_2}$.

Let $maxmean^{n_1n_2}_{L_1}(S) = \max_{0 \leq L_2 \leq L_1} mean^{n_1n_2}_{L_2, L_1 - L_2}$. Let $n = n_1 + n_2$. Since

$$mean^{n}_{L_1}(S) = \frac{1}{\binom{n}{L_1}} \sum_{L_2=0}^{L_1} \binom{n_1}{L_2}\binom{n_2}{L_1 - L_2} mean^{n_1n_2}_{L_2, L_1-L_2}(S)$$

we get $maxmean^{n_1n_2}_{L_1}(S) \geq mean^{n}_{L_1}(S)$ and equality holds if $mean^{n_1n_2}_{L_2, L_1-L_2}(S)$ does not depend on L_2. The above relationship holds, since

$$\sum_{L_2=0}^{L_1} \binom{n_1}{L_2}\binom{n_2}{L_1 - L_2} = \binom{n_1 + n_2}{L_1}$$

(Vandermonde's identity). Since

$$maxmean^{n_1n_2}(S) \geq maxmean^{n_1n_2}_{L_1}(S) \geq mean^{n}_{L_1}$$

we can expect that for a random renaming R algorithm $MAXMEAN^{n_1n_2}$ applied to $R(S)$ will find a better assignment than algorithm MAXMEAN applied to $R(S)$.

Algorithm MAXMEAN* has an interesting interpretation, namely, it is a polynomial algorithm for a relaxation of generalized maximum satisfiability.

The maximum ψ_d-satisfiability problem can be formulated in the following way: Given a ψ_d-formula S, find a renaming of the variables of S, so that the assignment that assigns 0 to all variables, is optimal.

Algorithm MAXMEAN* solves the following relaxation in polynomial time: Given a formula S, find a renaming, so that the assignment which assigns 0 to all variables, is optimal for the worst-case formula among all formulas similar to the renamed one. By the worst-case formula we mean the formula in which the minimal number of clauses can be satisfied. Two formulas are *similar*, if they contain for each relation R the same fraction of clauses containing R. The worst-case formulas are symmetric.

Conclusion and Open Problems

A ψ-formula S is said to be *k-satisfiable* if any k clauses can be satisfied. The *ψ-k-extremal problem* consists of: Find a polynomial algorithm which satisfies at least the fraction

$$\tau_{\psi, k} = \inf_{\substack{\text{all } k\text{-satisfiable} \\ \psi\text{-formulas } S}} \max_{\substack{\text{all assignments} \\ J \text{ of } S}} \frac{SA(S, J)}{cl(S)}$$

of the clauses in a k-satisfiable ψ-formula. The solution of the ψ-k-extremal problem for any k and ψ would give a striking insight into the structure of ψ-satisfiability problems.

In this paper the ψ-1-extremal problem is solved for any ψ since the set of 1-satisfiable ψ-formulas is closed under symmetrization. [13] contains partial results regarding the solution of the ψ-2-extremal problem.

Acknowledgments

Many thanks to P. van Emde Boas of the University of Amsterdam for pointing out a mistake in one of the formulas in section 7. I would like to thank my wife Dr. R. Lieberherr for stylistic improvements of the paper and my secretary S. Mairs for preparing it on the computer.

References

1. M. R. Garey and D. S. Johnson, Approximation algorithms for combinatorial problems: An annotated bibliography, pp. 41–52 in (J. F. Traub, ed.): *Algorithms and Complexity: New Directions and Recent Results*, Academic Press, New York, 1976.
2. T. Schaefer, The complexity of satisfiability problems, *Proc. 10th Annual ACM Symposium on Theory of Computing*, pp. 216–226 (1978).
3. P. Erdös and D. J. Kleitman, On coloring graphs to maximize the proportion of multicolored k-edges, *J. Combinat. Theory*. **5**. 1968, 164–169.
4. D. S. Johnson, Approximation algorithms for combinatorial problems, *J. Comput. Syst. Sci*. **9**, 1974, 256–278.
5. K. Lieberherr and E. Specker, Complexity of partial satisfaction, *J. ACM* **28**, 1981, 411–421.
6. K. J. Lieberherr, P-optimal heuristics, *Theoret. Comput. Sci*. **10**, 1980, 123–131.
7. P. Henrici, *Applied and Computational Complex Analysis*, vol. 1, Wiley, New York 1974.
8. A. Cayley, Note sur les fonctions de M. Sturm, *J. Math. Pures Appl*. **11**, 1846, 297–299.
9. G. E. Collins, Computer algebra of polynomials and rational functions, *Am. Math. Mon*. **80**, 1973, 725–755.
10. L. E. Heindel, Integer arithmetic algorithms for polynomial real zero determination, *J. Assoc. Comput. Mach*., **18**, 1971, 533–548.
11. S. Sahni and T. Gonzales, P-complete approximation problems, *J. ACM* **23**, 1976, 555–565.
12. K. Lieberherr and E. Specker, Complexity of partial satisfaction II, *Technical Rep*. 293, Dep. of EECS, Princeton University, 1982.
13. M. A. Huang, K. J. Lieberherr, Towards an ε-approximation scheme for generalized satisfiability, *Technical Rep*. 292, Dep. of EECS, Princeton University, 1981.
14. A. G. Akritas, The fastest exact algorithms for the isolation of the real roots of a polynomial equation, *Computing* **24**, 1980, 299–313.