A universally fastest algorithm for Max 2-Sat, Max 2-CSP, and everything in between[*]

Serge Gaspers[†] Gregory B. Sorkin[‡]

Abstract

We introduce "hybrid" Max 2-CSP formulas consisting of "simple clauses", namely conjunctions and disjunctions of pairs of variables, and general 2-variable clauses, which can be any integer-valued functions of pairs of boolean variables. This allows an algorithm to use both efficient reductions specific to AND and OR clauses, and other powerful reductions that require the general CSP setting.

Parametrizing an instance by the fraction p of non-simple clauses, we give an exact (exponential-time) algorithm that is the fastest polynomial-space algorithm known for Max 2-Sat (and other $p = 0$ formulas, with arbitrary mixtures of AND and OR clauses); the only efficient algorithm for mixtures of AND, OR, and general integer-valued clauses; and tied for fastest for general Max 2-CSP ($p = 1$). Since a pure 2-Sat input instance may be transformed to a general CSP instance in the course of being solved, the algorithm's efficiency and generality go hand in hand.

Our novel analysis results in a *family* of running-time bounds, each optimized for a particular value of p. The algorithm uses new reductions introduced here, as well as recent reductions such as "clause-learning" and "2-reductions" adapted to our setting's mixture of simple and general clauses. Each reduction imposes constraints on various parameters, and the running-time bound is an "objective function" of these parameters and p. The optimal running-time bound is obtained by solving a convex nonlinear program, which can be done efficiently and with a certificate of optimality.

1 Introduction

1.1 Treatment of "hybrid" Sat–CSP formulas

We show a polynomial-space algorithm that solves general instances of integer-valued Max 2-CSP (formally defined in Section 2), but takes advantage of "simple" clauses to reduce the running time, where simple clauses are unit-weighted conjunctions and disjunctions.

Running Time	Problem	Space	Reference
$O^\star\left(2^{m/2.879}\right)$	Max 2-Sat	poly	[13]
$O^\star\left(2^{m/3.448}\right)$	Max 2-Sat	poly	[1] (implicit)
$O^\star\left(2^{m/4}\right)$	Max 2-Sat	poly	[5]
$O^\star\left(2^{m/5}\right)$	Max 2-Sat	poly	[4]
$O^\star\left(2^{m/5}\right)$	Max 2-CSP	poly	[15]
$O^\star\left(2^{m/5.263}\right)$	Max 2-CSP	poly	[16]
$O^\star\left(2^{m/5.217}\right)$	Max 2-Sat	poly	[7]
$O^\star\left(2^{m/5.5}\right)$	Max 2-Sat	poly	[9]
$O^\star\left(2^{m/5.769}\right)$	Max 2-Sat	exp	[6]
$O^\star\left(2^{m/5.769}\right)$	Max 2-CSP	exp	[17]
$O^\star\left(2^{m/5.88}\right)$	Max 2-Sat	poly	[10]
$O^\star\left(2^{m/6.215}\right)$	Max 2-Sat	poly	[14]
$O^\star\left(2^{m/5.263}\right)$	Max 2-CSP	poly	(here)
$O^\star\left(2^{m/6.321}\right)$	Max 2-Sat	poly	(here)

Table 1: A historical overview of algorithms for Max 2-Sat and Max 2-CSP

Let us give a simple example. In the instance

$$\begin{aligned}&(x_1 \vee x_2) + (x_2 \vee \overline{x_4}) + (x_2 \wedge x_3) + 3 \cdot (x_1 \vee x_3)\\ &\qquad +(2 \cdot (\overline{x_2}) - 5 \cdot x_4 + (x_2 \oplus x_4)),\end{aligned} \tag{1.1}$$

the first two clauses are unit-weighted disjunctive clauses, the third is a unit-weighted conjunction, the fourth is a disjunction with weight 3, and the last is a general integer-valued CSP clause (any integer-valued 2×2 truth table). Thus this example has 3 simple clauses (the first three) and 2 non-simple clauses.

Both Max 2-Sat and Max 2-CSP have been extensively studied from the algorithmic point of view. For variable-exponential running times, the only two known algorithms faster than 2^n for Max 2-CSP (or even Max 2-Sat) are those by Williams [19] and Koivisto [8], both with running time $O^\star\left(2^{n/1.262}\right)$. They employ beautiful ideas, but have exponential space complexity.

For clause-exponential running times, there has been a long series of improved algorithms; see Table 1. To solve Max 2-Sat, all early algorithms treated pure 2-Sat formulas. By using more powerful reductions

[*]Additional support from the Norwegian Research Council.

[†]Department of Informatics, University of Bergen, N-5020 Bergen, Norway, `serge@ii.uib.no`

[‡]Department of Mathematical Sciences, IBM T.J. Watson Research Center, Yorktown Heights NY 10598, USA, `sorkin@watson.ibm.com`

closed over Max 2-CSP but not Max 2-Sat, the Max 2-CSP generalization of Scott and Sorkin [16] led to a faster algorithm. Then, several new Max 2-Sat-specific reductions once again gave the edge to algorithms addressing Max 2-Sat particularly.

In this paper we get the best of both worlds by using both especially efficient reductions specific to Max 2-Sat (actually, we allow conjunctive as well as disjunctive clauses), and powerful CSP reductions. While it is likely that Max 2-Sat algorithms will become still faster, we believe that further improvements will continue to use this method of combination.

1.2 Results Let p be the fraction of non-simple clauses in the input instance, no matter how this fraction changes during the execution of the algorithm (in example (1.1), $p = 2/5$). The algorithm we present is the fastest known polynomial-space exact algorithm for $p = 0$ (including Max 2-Sat but also instances with arbitrary mixtures of AND and OR clauses); fastest for all $p < 0.29$ (where no other algorithm is known, short of solving the instance as a case of general Max 2-CSP); and tied with [17] for fastest for $0.29 \leq p \leq 1$ (notably for Max 2-CSP itself), though [17] solves real- as well as integer-valued CSPs. For the well-studied classes Max 2-Sat and Max 2-CSP, our algorithm has running times $O^\star\left(2^{m/6.321}\right)$ and $O^\star\left(2^{m/5.263}\right)$, respectively.

For "cubic" instances, where each variable appears in at most three 2-variable clauses, our analysis gives running-time bounds that match and generalize the best known when $p = 0$ (including Max 2-Sat); improve on the best known when $0 < p < 1/2$; and match the best known for $1/2 \leq p \leq 1$ (including Max 2-CSP).

We derive running-time bounds that are optimized to the fraction p of non-simple clauses; see Table 2. Every such bound is valid for every formula, but the bound derived for one value of p may not be the best possible for a formula with a different value.

1.3 Method of analysis, and hybrid Sat–CSP formulas Since a fair amount of machinery will have to be introduced before we can fully explain our analysis, let us first give a simplified overview. Our algorithm reduces an instance to one or more smaller instances, which are solved recursively to yield a solution to the original instance. We view a Max 2-CSP instance as a constraint graph $G = (V, E \cup H)$ where vertices represent variables, the set of "light" edges E represents simple clauses and the set of "heavy" edges H respresents general clauses. The reductions are usually local and change the constraint graph's structure, and a related *measure*, in a predictable way.

For example, if G has two degree-4 vertices sharing two simple clauses, a "parallel-edge" reduction replaces the two simple clauses with one general clause, changing the vertices' degrees from 4 to 3, giving a new constraint graph G'. With the measure μ including weights w_e and w_h for each simple and general clause, and weights w_3 and w_4 for each vertex of degree 3 and 4, this reduction changes an instance's measure by $\mu(G') - \mu(G) = -2w_e + w_h - 2w_4 + 2w_3$. An inductive proof of a running-time bound $O^\star\left(2^{\mu(G)}\right)$ will follow *if the measure change is non-positive.* Thus, we constrain that $-2w_e + w_h - 2w_4 + 2w_3 \leq 0$.

An algorithm requires a set of reductions covering all instances: there must always be some applicable reduction. Just as above, each reduction imposes a constraint on the weights. One reduction's constraint can weaken those of other reductions, by limiting the cases in which they are applied. For example, if we prioritize parallel-edge reduction, we may assume that other reductions act on graphs without parallel edges. Reductions producing a single instance, or any number of isomorphic instances, yield linear constraints (as in [15–17]); reductions producing distinct instances yield nonlinear, convex constraints.

If a set of weights giving a measure μ satisfies all the constraints, the analysis results in a proof of a running-time bound $O^\star\left(2^{\mu(G)}\right)$ for an input instance G. To get the best possible running-time bound subject to the constraints, we wish to minimize $\mu(G)$. To avoid looking at the full degree spectrum of G, we constrain each vertex weight w_d to be non-positive, and then ignore these terms, resulting in a (possibly pessimistic) running-time bound $O^\star\left(2^{|E|w_e+|H|w_h}\right)$.

If G is a Max 2-Sat instance, to minimize the running-time bound is simply to minimize w_e subject to the constraints: as there are no heavy edges in the input instance, it makes no difference if w_h is large. This optimization will yield a small value of w_e and a large w_h. Symmetrically, if we are treating a general Max 2-CSP instance, where all edges are heavy, we need only minimize w_h. This optimization will yield weights w_e, w_h that are larger than the Max 2-Sat value of w_e but smaller than its w_h. For a hybrid instance with some edges of each type, minimizing $|E|w_e + |H|w_h$ is equivalent to minimizing $(1-p)w_e + pw_h$, where $p = |H|/(|E|+|H|)$ is the fraction of non-simple clauses. This will result in weights w_e and w_h each lying between the extremes given by the pure 2-Sat and pure CSP cases; see Figure 1.

Thus, a new aspect of our approach is that it results in a family of nonlinear programs (NLPs), not just one: the NLPs differ in their objective functions, which are tuned to the fraction p of non-simple clauses in an input instance. The optimization done for a particular value

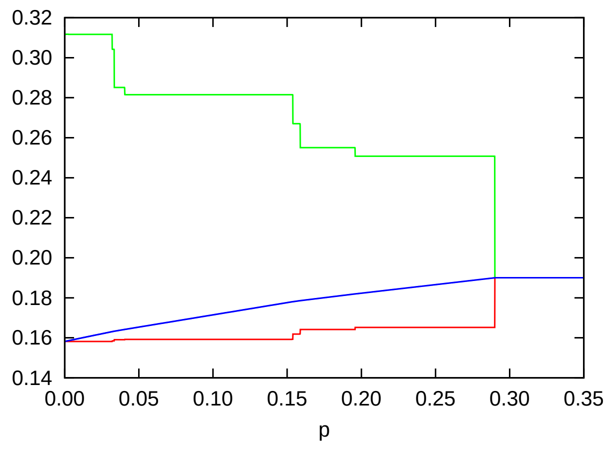

Figure 1: Plot of w_e (red), w_h (green), and the running-time exponent $(1-p)w_e+pw_h$ (blue) versus the fraction p of non–simple 2-clauses. The three values are equal (and exactly 0.19) for $p > 0.29$. Both w_e and w_h appear to be piecewise constant: the resolution of the graph is in p increments of 0.0001, and all the small changes are meaningful

of p, by construction, gives a running-time bound that is the best possible (within our methods) for an input instance with this fraction of non-simple clauses, but (because the constraints are the same in all the NLPs) that is valid for all instances; see the caption of Table 2.

1.4 Novel aspects of the analysis Our introduction of the notion of hybrids between Max 2-Sat and Max 2-CSP, discussed above, is the main distinguishing feature of the present work. It yields a more general algorithm, applicable to CSP instances not just Sat instances, and gives better performance on Max 2-Sat by allowing both efficient Sat-specific reductions and powerful reductions that go outside that class. This is surely not the final word on Max 2-Sat algorithms, but we expect new algorithms to take advantage of this hybrid approach.

A secondary point is that CSP reductions such as combining parallel edges or reducing on small cuts mean that in other cases it can be assumed that a graph has no parallel edges or small cuts. This simplifies the case analysis, counter-balancing the complications of considering two types of edges.

Our analysis uses a now-common method, but with some novel aspects. Specifically, we analyze a reduction-based algorithm with a potential-function method akin to the measures used by [11, 12], the quasi-convex analysis of [2], the "measure and conquer" approach of [3], the (dual to the) linear programming approach of [17], and much older potential-function analyses in mathematics and physics. The goal is to solve a NLP giving a set of weights which minimizes a running-time bound, while respecting constraints imposed by the reductions. The hybrid view marks the biggest change to this approach, since, as already discussed, it means that the objective function depends on the fraction of non-simple clauses, so there is a continuum of NLPs, not just one.

Also, it is common to make some assumptions about the weights, but we try to avoid this, instead only limiting the weights by the constraints necessitated by each reduction. This avoids unnecessary assumptions compromising optimality of the result, which is especially important in the hybrid realm where an assumption might be justified for Sat but not for CSP, or vice-versa. It also makes the analysis more transparent.

Our nonlinear programs are convex, allowing them to be solved quickly and with certificates of optimality. While this should be true of some previous works, we have not seen it observed before.

As is often the case with exact algorithms, regularity of an instance is important, and in our analysis we treat this with explicit weights penalizing regularity (motivated by a similar accounting for the number of 2-edges in a hypergraph in [18], and the "forced moves" in [17]). This introduces some extra bookkeeping but results in a more structured, more verifiable analysis.

We introduce several new reductions, including a 2-reduction combining ideas from [9] (for the Sat case) and [17] (the CSP case), a "super 2-reduction", and a generalization of the "clause-learning" from [10].

2 Definitions

We use the value 1 to indicate Boolean "true", and 0 "false". The canonical problem Max Sat is, given a boolean formula in conjunctive normal form (CNF), to find a boolean assignment to the variables of this formula satisfying a maximum number of clauses. Max 2-Sat is Max Sat restricted to instances in which each clause contains at most 2 literals.

We consider a class more general than Max 2-Sat, namely integer-valued Max (2,2)-CSP; we abbreviate this to Max 2-CSP. An instance (G, S) of Max 2-CSP is defined by a *constraint graph* (or multigraph) $G = (V, E)$ and a set S of *score* functions. There is a *dyadic* score function $s_e\colon \{0,1\}^2 \to \mathbb{Z}$ for each edge $e \in E$, a monadic score function $s_v\colon \{0,1\} \to \mathbb{Z}$ for each vertex $v \in V$, and (for bookkeeping convenience) a single *niladic* score "function" $s_\emptyset\colon \{0,1\}^0 \to \mathbb{Z}$.

A candidate solution is a function $\phi : V \to \{0,1\}$ assigning values to the vertices, and its score is

$$s(\phi) := \sum_{uv\in E} s_{uv}(\phi(u), \phi(v)) + \sum_{v\in V} s_v(\phi(v)) + s_\emptyset.$$

An optimal solution ϕ is one which maximizes $s(\phi)$.

A hybrid instance $F = (V, E, H, S)$ is defined by its variables or vertices V, normal or *light* edges E representing conjunctive clauses and disjunctive clauses, *heavy* edges H representing arbitrary (integer-valued) clauses, and a set S of monadic functions and dyadic clauses. Its light-and-heavy-edged constraint graph is $G = (V, E, H)$, though generally we just think of the graph $(V, E \cup H)$; no confusion should arise. We write $V(F)$ and $V(G)$ for the vertex set of an instance F or equivalently that of its constraint graph G.

In a graph G, we define the *(open) neighborhood* of a vertex u as $N(u) := \{v : uv \in E \cup H\} \setminus \{u\}$ (excluding u will not matter once we simplify our graphs and make them loopless), and the closed neighborhood as $N[u] := N(u) \cup \{u\}$. Generalizing, a *set* of vertices, U, has (open) neighborhood $N(U) = \left(\bigcup_{u \in U} N(u)\right) \setminus U$, and (open) *second neighborhood* $N^2(U) = N(N(U)) \setminus U$. For a single vertex u, define $N^2(u) := N^2(\{u\})$.

We define the degree $\deg(u)$ of a vertex u to be the number of edges incident on u where loops are counted twice, and the *degree* (or *maximum degree*) of a formula F (or its constraint graph G) to be the maximum of its vertex degrees. Without loss of generality we assume that there is at most one score function for each vertex. Then, up to constant factors the space required to specify an instance F with constraint graph $G = (V, E, H)$ is the *instance size* $|F| = 1 + |V| + |E| + |H|$.

We use the symbol $\boxdot$ to end the description of a reduction rule or the analysis of a case, and $\square$ to end a proof. Some proofs and details had to be omitted in this extended abstract due to space constraints.

3 Algorithm and outline of the analysis

We will show an algorithm which, on input of a hybrid instance F, returns an optimal coloring ϕ of F's vertices in time $O^\star\left(2^{w_e|E|+w_h|H|}\right)$, which is to say in time

$$T(F) \le \text{poly}(|F|) 2^{w_e|E|+w_h|H|}. \tag{3.2}$$

3.1 Algorithm and central argument The algorithm is recursive: on input of an instance F, in time polynomial in the instance size $|F|$, F is *reduced* to a single instance F' (a *simplification*) or to several instances $F_1, \ldots, F_k$ (a *splitting*), each of smaller size; the algorithm solves the reduced instance(s) recursively; and, again in time poly$(|F|)$, the algorithm constructs an optimal solution to F from the solutions of the reduced instances.

The central argument is to establish (3.2) for *simplified* formulas of maximum degree ≤ 6. We do this now, in Lemma 3.1, with the bulk of the paper devoted to verifying the lemma's hypotheses. Lemma 3.1 gives a bound, $T(F) \le |F|^k 2^{\mu(F)}$, which is stronger if (as we will ensure) for some constant C and every simplified instance F of degree ≤ 6, the *measure* $\mu(F)$ satisfies

$$\mu(F) \le w_e|E| + w_h|H| + C. \tag{3.3}$$

Lemma 3.1. (Main Lemma) *Suppose there exists an algorithm A and constants $D, c \ge 1$, such that on input of any hybrid CSP instance F of maximum degree $\le D$, A either solves F in time at most 1, or decomposes F into instances $F_1, \ldots, F_k$ all with maximum degree $\le D$, solves these recursively, and* inverts *their solutions to solve F, using time at most $|F|^c$ for the decomposition and inversion (but not the recursive solves). Further suppose that for a given measure μ,*

$$(\forall F) \quad \mu(F) \ge 0, \tag{3.4}$$

and, for any decomposition done by algorithm A,

$$(\forall i) \quad |F_i| \le |F| - 1, \text{ and} \tag{3.5}$$

$$2^{\mu(F_1)} + \cdots + 2^{\mu(F_k)} \le 2^{\mu(F)}. \tag{3.6}$$

Then A solves any instance F of maximum degree $\le D$ in time at most $|F|^{c+1} 2^{\mu(F)}$.

We will often work with the equivalent to (3.6), that

$$\sum_{i=1}^{k} 2^{\mu(F_i)-\mu(F)} \le 1. \tag{3.6$'$}$$

3.2 Measure We will use a measure μ on simplified instances of maximum degree 6, where μ is a sum of weights associated with light edges, heavy edges, and vertices of various degrees (at most 6), and constants associated with the maximum degree d of F and whether F is regular (for regularity, not distinguishing between light and heavy edges):

$$\mu(F) := \nu(F) + \delta(F), \text{ with} \tag{3.7}$$

$$\nu(F) := |E|w_e + |H|w_h + \sum_{v \in V} w_{\deg(v)}, \text{ and} \tag{3.8}$$

$$\delta(F) := \sum_{d=4}^{6} \chi(\text{maxdeg}(G) \ge d) C_d + \sum_{d=4}^{6} \chi(G \text{ is } d\text{-regular}) R_d. \tag{3.9}$$

Here $\chi(\cdot)$ is the indicator function: 1 if its argument is true, 0 otherwise.

To satisfy condition (3.3) it is sufficient that

$$(\forall d) \quad w_d \le 0; \tag{3.10}$$

this is also necessary for large regular instances. Since we are now only considering instances of degree ≤ 6, we interpret "$\forall d$" to mean for all $d \in \{0, 1, \ldots, 6\}$.

3.3 Peripheral argument

LEMMA 3.2. *Suppose that every* simplified *Max 2-CSP instance F of degree at most $D \le 6$ can be solved in time $O^\star\left(2^{\mu(F)}\right)$. Then every instance F of degree at most D can be solved in time $O^\star\left(2^{\mu(F)}\right)$. Moreover, if for $D = 6$ the corresponding μ has $w_e, w_h \ge 1/7$, then every instance F can be solved in time $O^\star\left(2^{w_e|E|+w_h|H|}\right)$.*

3.4 Optimizing the measure The task of the rest of the paper is to produce the comprehensive set of reductions hypothesized by Lemma 3.1 (to any formula there should be some reduction we can apply), and a measure μ satisfying the hypotheses. For a 2-Sat instance we would like μ to have w_e as small as possible; more generally, for a formula with $m(1-p)$ simple clauses and mp general integer-valued clauses, we wish to minimize $(1-p)w_e + pw_h$.

For each reduction, the hypothesized constraint (3.5) will be trivially satisfied, and it will be straightforward to write down a constraint ensuring (3.6′). We then solve the nonlinear program of minimizing $(1-p)w_e + pw_h$ subject to all the constraints.

Minimizing $(1-p)w_e + pw_h$ for a given set of constraints can be done with an off-the-shelf nonlinear solver, but finding a set of reductions resulting in a small value remains an art.

With the constraints established in the next sections, we will obtain our main result.

THEOREM 3.1. *Let F be an instance of integer-weighted Max 2-CSP in which each variable appears in at most $\Delta(F)$ 2-clauses, and there are $(1-p(F))m$ conjunctive and disjunctive 2-clauses, and $p(F)m$ other 2-clauses. Then, for any pair of values w_e, w_h in Table 2 (not necessarily with the table's p equal to $p(F)$), the above algorithm solves F in time $O^\star\left(2^{m\cdot((1-p(F))w_e+p(F)w_h)}\right)$.*

Which of the constraints are tight strongly depends on p and $\Delta(F)$.

3.5 The measure's form Let us explain the rather strange form of the measure. Ideally, it would be defined simply as ν, and indeed for the measure we ultimately derive, most of our simplifications and splittings satisfy the key inequality (3.6′) with ν alone in place of μ. Unfortunately, for regular instances of degrees 4–6, satisfying (3.6′) would require a larger value of w_e. Viewing (3.6′) equivalently as $\sum_{i=1}^k 2^{\mu(F_i)-\mu(F)} \le 1$, adding a cost R_d to the measure of a d-regular instance F means that if a d-regular instance F is reduced to nonregular instances F_i of degree d, each difference $\mu(F_i) - \mu(F)$ is smaller by R_d than the corresponding difference $\nu(F_i) - \nu(F)$. We will therefore want

$$(3.11) \qquad (\forall d \in \{4,5,6\}) \quad R_d \ge 0.$$

If a nonregular instance F of degree d is reduced to instances F_i one or more of which is d-regular, there will be a corresponding penalty: for each d-regular F_i, $\mu(F_i) - \mu(F)$ is $\nu(F_i) - \nu(F) + R_d$.

The case where a nonregular instance of degree d produces a regular instance F_i of degree $< d$ can be dispensed with simply by choosing C_d sufficiently large to reap whatever additional reward is needed. Our splitting rules are generally local and will never increase measure by more than a constant, so some constant C_d suffices. Also, our reductions never increase the degree of an instance, so C_d will never work against us, and there is no harm in choosing it as large as we like. Thus, we never need to consider the cases where the instance degree decreases, nor the values C_d.

The remaining cases where a nonregular instance has regular children will be considered on a case-by-case basis for each reduction. Generally, for a child to become regular means that, beyond the constraint-graph changes taken into account in the baseline case (with the child nonregular), all the vertices of degree less than d must have been removed from the instance by simplifications. Accounting for these implies a further decrease in measure that compensates for the increase by R_d.

4 Some initial constraints

Let us write $w(v)$ for the weight of a vertex v (so $w(v) = w_d$ for a vertex of degree d), and similarly $w(e)$ for the weight of an edge (w_e or w_h depending on whether e is light or heavy). Sometimes it will be helpful to think of $\nu(F)$ as

$$(4.12) \qquad \nu(F) = \sum_{v \in V}\Big(w(v) + \tfrac{1}{2}\sum_{e:\, v\in e} w(e)\Big),$$

the sum of the weights of the vertices and their incident *half edges*. We define (and thus constrain)

$$(4.13) \qquad a_d = w_d + \tfrac{1}{2} d w_e.$$

Considering d-regular Max 2-Sat instances, we ensure (3.4) by

$$(4.14) \qquad (\forall d) \quad a_d \ge 0 \text{ and}$$

$$(4.15) \qquad (\forall d \in \{4,5,6\}) \quad C_d, R_d \ge 0.$$

As heavy edges generalize light ones, $w_e \le w_h$.[1] For intuitive purposes let us reveal that we will find that

[1]For the most part we will only write down constraints that are *necessary*, typically being required for some reduction to satisfy (3.6′), but we make a few exceptions early on.

p	0			0.1			0.2			1		
$\Delta(F)$	w_e	w_h	w	w_e	w_h	w	w_e	w_h	w	w_e	w_h	w
3	0.10209	0.23127	0.10209	0.10209	0.23125	0.11501	0.10209	0.23125	0.12793	0.16667	0.16667	0.16667
4	0.14662	0.31270	0.14662	0.15023	0.26952	0.16216	0.15023	0.26951	0.17409	0.18751	0.18751	0.18751
5	0.15522	0.30876	0.15523	0.15664	0.27892	0.16887	0.15664	0.27892	0.18109	0.19001	0.19001	0.19001
≥ 6	0.15820	0.31174	0.15821	0.15925	0.28154	0.17148	0.16520	0.25074	0.18231	0.19001	0.19001	0.19001

Table 2: Values of w_e, w_h and $w := pw_h + (1-p)w_e$ according to the fraction p of heavy edges and the maximum degree $\Delta(F)$ of a formula F. For any pair (w_e, w_h) in the table, a running-time bound of $O^\star\left(2^{m\cdot((1-p)w_e+pw_h)}\right)$ is valid for every formula, regardless of its fraction $p(F)$ of non-simple clauses, but the pair obtained when the table's p equals $p(F)$ gives the best bound.

$0 = a_0 = a_1 = a_2 < a_3 < \cdots < a_6$. Typically $w_h \leq 2w_e$, but not always. This "intuition" changed several times as the paper evolved, which supports the value of making as few assumptions as possible, instead just writing down constraints implied by the reductions.

5 Simplification rules and their weight constraints

We use a number of simplification rules (reductions of F to a single simpler instance F_1 or F'). We have already ensured constraint (3.4) by (4.14) and (4.15). Constraint (3.5) of Lemma 3.1 will be trivially satisfied by all our simplifications, so our focus is on ensuring that each reduction satisfies (3.6′).

Some of the simplification rules are standard, the CSP 1-reductions are taken from [17], the CSP 2-reductions combine ideas from [17] and [9], and a "super 2-reduction" is introduced here. For vertices of degree 5 we use a splitting reduction taken from [10] that we generalize to hybrid instances. Considering δ later, here we show that each simplification from F to F' satisfies

$$\nu(F') \leq \nu(F). \tag{5.16}$$

5.1 Combine parallel edges Two parallel edges (light or heavy) with endpoints x and y may be collapsed into a single heavy edge [17]. If one of the endpoints, say x, of the two parallel edges has degree 2, collapse the parallel edges and immediately apply a 1-reduction (see reduction 5.6) on x (of degree 1), which removes x from the constraint graph. To ensure (5.16) we constrain

$$(\forall d \geq 2) \quad -a_2 - a_d + a_{d-2} \leq 0: \tag{5.17}$$

the left hand side is $\nu(F') - \nu(F)$ thought of as subtracting a vertex of degree 2 and replacing a vertex of degree d by one of degree $d-2$. For the case that x and y have degree $d \geq 3$, we constrain

$$(\forall d \geq 3) \quad -2a_d + 2a_{d-1} - w_e + w_h \leq 0: \tag{5.18}$$

thought of as replacing two vertices of degree d by two vertices of degree $d-1$ and replacing a light by a heavy edge. If $\deg(x) \neq \deg(y)$, the resulting constraint is a half–half mixture of (5.18) with $d = \deg(x)$ and another with $d = \deg(y)$, and is thus redundant. ⊡

5.2 Remove loops If the instance includes any edge $xx \in E \cup H$, the nominally dyadic score function $s_{xx}(\phi(x), \phi(x))$ may be replaced by a (or incorporated into an existing) monadic score function $s_x(\phi(x))$. This imposes the constraints

$$(\forall d \geq 2) \quad -a_d + a_{d-2} \leq 0. \tag{5.19}$$

Note that we may ignore constraint (5.17) now as it is weaker than (5.19) by (4.14). ⊡

We may from now on assume the constraint graph is simple.

5.3 0-reduction If v is a vertex of degree 0, reduce the instance F to F' by deleting v and its monadic score function s_v. Constraint (3.5) is satisfied, since $|F'| = |F| - 1$. Constraint (5.16) is satisfied iff $-w_0 \leq 0$. On the other hand, $w_d \leq 0$ (inequality (3.10)), implying that $w_0 = 0$, and thus

$$a_0 = 0. \tag{5.20}$$

We will henceforth ignore vertices of degree 0 completely. ⊡

5.4 Delete a small component For a constant C (whose value we will fix later (reduction 7.1)), if the constraint graph G of F has components G' and G'' with $1 \leq |V(G'')| < C$, then F may be reduced to F' with constraint graph G'. The reduction and its correctness are obvious, noting that F'' may be solved in constant time. Since $\nu(F') - \nu(F) = -\sum_{v \in V(G)} a_{\deg(v)}$, it is immediate from (4.14) that (5.16) is satisfied. ⊡

5.5 Delete a decomposable edge If a dyadic score function $s_{xy}(\phi(x), \phi(y))$ can be expressed as a sum of monadic scores, $s'_x(\phi(x)) + s'_y(\phi(y))$, then delete the edge and add s'_x to the original s_x, and s'_y to s_y. The constraint imposed is that

$$(5.21) \qquad (\forall d \geq 1) \quad -a_d + a_{d-1} \leq 0.$$

Note that we may ignore constraint (5.19) now as it is weaker than (5.21). $\boxdot$

Two remarks. First, together with (5.20), (5.21) means that

$$(5.22) \qquad 0 = a_0 \leq a_1 \leq \cdots \leq a_6.$$

Second, if an edge is not decomposable, the assignment of either endpoint has a (nonzero) bearing on the optimal assignment of the other, as we make precise in Remark 1. We will exploit this in Lemma 6.1, which shows how "super 2-reduction" opportunities (reduction 6.1) are created.

Remark 1. *Let* $\text{bias}_y(i) := s_{xy}(i,1) - s_{xy}(i,0)$, *the "preference" of* s_{xy} *for setting* $\phi(y) = 1$ *over* $\phi(y) = 0$ *when* x *is assigned* $\phi(x) = i$. *Then* s_{xy} *is decomposable iff* $\text{bias}_y(0) = \text{bias}_y(1)$.

5.6 1-reduction This reduction comes from [17], and works regardless of the weight of the incident edge. Let y be a vertex of degree 1, with neighbor x. We use the fact that the optimal assignment of y is an easily-computable function of the assignment of x, and thus y and its attendant score functions $s_y(\phi(y))$ and $s_{xy}(\phi(x), \phi(y))$ can be incorporated into $s_x(\phi(x))$.

Since the reduction deletes the vertex of degree 1 and its incident edge (light, in the worst case), and decreases the degree of the adjacent vertex, (5.16) is ensured by (5.22). $\boxdot$

5.7 1-cut Let x be a cut vertex isolating a set of vertices A, $2 \leq |A| \leq 10$. (The 1-cut reduction extends the 1-reduction, thought of as the case $|A| = 1$.) Informally, for each of $\phi(x) = 0,1$ we may determine the optimal assignments of the vertices in A and the corresponding optimal score; adding this score to the original monadic score s_x gives an equivalent instance F' on variables $V \setminus A$.

This simplification imposes no new constraint on the weights. Vertices in A are deleted and x has its degree reduced; by (4.14) and (5.21), neither increases the measure ν. $\boxdot$

5.8 2-reduction Let y be a vertex of degree 2 with neighbors x and z. Then y may be contracted out of the instance: the old edges xy, yz, and (if any) xz are replaced by an edge xz which in general is heavy, but is light if there was no existing edge xz and at least one of xy and yz was light.

If there is an edge xz then $\deg(x), \deg(y) \geq 3$ and we use the general Max 2-CSP 2-reduction from [17]. Arguing as in the 1-reduction above, here the optimal assignment of y depends only on the assignments of x and z, and thus we may incorporate all the score terms involving y into $s_{xz}(\phi(x), \phi(z))$. The effect is that y is deleted, three edges (in the worst case all light) are replaced by one heavy edge, and the degrees of x and z decrease by one. Thus, we constrain

$$(5.23) \quad (\forall d \geq 3) \quad -a_2 - w_e + w_h - 2a_d + 2a_{d-1} \leq 0.$$

If xy *or* yz *is heavy, then* $\nu(F') - \nu(F) \leq -w_h + w_e$, *and we will capitalize on this later.*

Finally, we consider the case where there was no edge xz. If xy and yz are both heavy, then as in the first case we apply the general Max 2-CSP reduction to replace them with a heavy edge xz, giving $\nu(F') - \nu(F) \leq -w_h + w_e$. Otherwise, at least one of xy and yz is light, and it can be shown that the resulting edge xz is light. If both xy and yz are light, $\nu(F') - \nu(F) \leq -a_2 \leq 0$, while (once again) if one of xy and yz is heavy, $\nu(F') - \nu(F) \leq -w_h + w_e$. $\boxdot$

5.9 2-cut Let $\{x, y\}$ be a 2-cut isolating a set of vertices A, $2 \leq |A| \leq 10$. (The 2-cut reduction extends the 2-reduction, thought of as the case $|A| = 1$.) Similarly to the 1-cut, for each of the four cases $\phi : \{x,y\} \to 0,1$ we may determine the optimal assignments of the vertices in A and the corresponding optimal score; adding this score function to the original dyadic score s_{xy} gives an equivalent instance F' on variables $V \setminus A$.

In general, $\nu' - \nu$ may be equated with the weight change from deleting the original edge xy if any (guaranteed by (5.21) not to increase the measure), deleting all vertices in A (a change of $-\sum_{v \in A} a_{\deg(v)}$), replacing one half-edge from each of x and y into A with a single heavy edge between x and y (not affecting their degrees, and thus a change of $-w_e + w_h$), then removing any half-edges remaining from other edges in $\{x, y\} \times A$ Thus we can assure $\nu' - \nu \leq 0$ by $-2a_3 - w_e + w_h \leq 0$, which is already imposed by (4.14) and (5.18). $\boxdot$

6 Some useful tools

The property of disjunction and conjunction on which we rely to treat them more efficiently (besides having range $\{0, 1\}$) is that they are monotone in each variable. Obviously exclusive-or is not monotone, and it seems that it cannot be accommodated by our methods.

6.1 Super 2-reduction Suppose that y is of degree 2 and that its optimal color $C \in \{0, 1\}$ is independent of the colorings of its neighbors x and z, i.e.,

$$(\forall D, E) \quad s_y(C) + s_{yx}(C, D) + s_{yz}(C, E)$$
$$(6.24) \qquad = \max_{C' \in \{0,1\}} s_y(C') + s_{yx}(C', D) + s_{yz}(C', E).$$

In that case, $s_y(\phi(y))$ can be replaced by $s_y(C)$ and incorporated into the niladic score, $s_{xy}(\phi(x), \phi(y))$ can be replaced by a monadic score $s'_x(\phi(x)) := s_{xy}(\phi(x), C)$ and combined with the existing s_x. The same holds for s_{yz}, resulting in an instance with y deleted. $\boxdot$

A super 2-reduction is better than a usual one since y is deleted, not just contracted.

We will commonly *split* on a vertex u, setting $\phi(u) = 0$ and $\phi(u) = 1$ to obtain instances F_0 and F_1, and solving both.

LEMMA 6.1. *After splitting a simplified instance F on a vertex u incident to a vertex y of degree 3 whose other two incident edges xy and yz are both light, in at least one of the reduced instances F_0 or F_1, y is subject to a super 2-reduction.*

Proof. In the clauses represented by the light edges xy and yz, let $b \in \{-2, 0, 2\}$ be the number of occurrences of y minus the number of occurrences of $\bar{y}$. Following the fixing of u to 0 or 1 and its elimination, let $\text{bias}_y := s_y(1) - s_y(0)$. Given that F was simplified, the edge uy was not decomposable, so by Remark 1 the value of bias_y in F_0 is unequal to its value in F_1. First consider the case $b = 0$. If $\text{bias}_y \geq 1$, the advantage from bias_y for setting $\phi(y) = 1$ rather than 0 is at least equal to the potential loss (at most 1) from the one negative occurrence of y in xy and yz, so the assignment $\phi(y) = 1$ is always optimal. Symmetrically, if $\text{bias}_y \leq -1$ we may set $\phi(y) = 0$. The only case where we cannot assign y is when $\text{bias}_y = 0 = -b/2$. Next consider $b = 2$. (The case $b = -2$ is symmetric.) If $\text{bias}_y \geq 0$ we can fix $\phi(y) = 1$, while if $\text{bias}_y \leq -2$ we can fix $\phi(y) = 0$. The only case where we cannot assign y is when $\text{bias}_y = -1 = -b/2$. Thus, we may optimally assign y independent of the assignments of x and z unless $\text{bias}_y = -b/2$. Since bias_y has different values in F_0 and F_1, in at least one case $\text{bias}_y \neq -b/2$ and we may super 2-reduce on y.

Lemma 6.1 relies on bias_y taking integral values, and is the sole reason our algorithm works with integer-valued CSP clauses but not real-valued ones.

6.2 Splitting on vertices of degree 5 Kulikov and Kutzkov [10] introduced a clever splitting on vertices of degree 5. The basic idea is the same one that went into our 2-reductions: in some circumstances an optimal assignment of a variable is predetermined. In addition to generalizing from degree 3 to degree 5 (from which the generalization to every degree is obvious), [10] also applies the idea somewhat differently.

The presentation in [10] is specific to 2-Sat. Reading their result, it seems unbelievable that it also applies to Max 2-CSP as long as the vertex being reduced upon has only light edges (even if its neighbors have heavy edges), but in fact the proof carries over unchanged.

LEMMA 6.2. (CLAUSE LEARNING) *In a Max 2-CSP instance F, let u be a variable of degree 5, with light edges only, and neighbors $v_1, \dots, v_5$. Then there exist "preferred" colors C_u for u and C_i for each neighbor v_i such that a valid splitting of F is into three instances: F_1 with $\phi(u) = C_u$; F_2 with $\phi(u) \neq C_u$, $\phi(v_1) = C_1$; and F_3 with $\phi(u) \neq C_u$, $\phi(v_1) \neq C_1$, and $\phi(v_i) = C_i$ ($\forall i \in \{2, 3, 4, 5\}$).*

7 Splitting reductions and preference order

Recall that if we have a nonempty simplified instance F, we will apply a splitting reduction to produce smaller instances $F_1, \dots, F_k$, simplify each of them, and argue that $\sum_{i=1}^{k} 2^{\mu(F_i) - \mu(F)} \leq 1$ (inequality (3.6′)). We apply splitting reductions in a prescribed order of preference, starting with division into components.

7.1 Split large components If the constraint graph G of F has components G_1 and G_2 with at least C vertices each (C is the same constant as in the simplification rule (5.4)), decompose F into the corresponding instances F_1 and F_2.

Note that $\nu(F_1) + \nu(F_2) = \nu(F)$, and $\nu(F_i) \geq Ca_3$ since F_i has at least C vertices, all degrees are at least 3, and the a_i are nondecreasing. Thus $\nu(F_1) \leq \nu(F) - Ca_3$. Also, $\delta(F_1) - \delta(F)$ is constant-bounded. Assuming that $a_3 > 0$ (following from the splitting on degree 3 vertices), then for C sufficiently large,

$$\mu(F_1) - \mu(F) = \nu(F_1) - \nu(F) + \delta(F_1) - \delta(F)$$
$$\leq -Ca_3 + \sum\nolimits_{d=4}^{6} (R_d + C_d) \leq -1.$$

The same is of course true for F_2, giving $2^{\mu(F_1) - \mu(F)} + 2^{\mu(F_2) - \mu(F)} \leq 1$ as required. $\boxdot$

If F's constraint graph is connected, the splitting we apply depends on the degree of F. Separate case analyses are needed for degrees 3-6, and space does not allow presenting all of them. We show the treatment of degree 5 since it uses the full range of techniques.

8 Instances of degree 5

In this section we present the splitting rules for vertices of degree 5. Note that the simplification rules imply that F has minimum degree 3. For any $d \geq 3$, we define $h_d := \min_{3 \leq i \leq d} \{a_i - a_{i-1}\}$. This is the minimum possible decrease of measure resulting from a *good degree-reduction*, that is the decrease of the degree of a vertex of degree i with $3 \le i \le d$ by the deletion of a half-edge. Such deletions always occur with the same sign in our NLP — the larger h_d, the weaker each constraint is — and therefore the above definition can be expressed in our mathematical program by inequalities

$$(8.25) \qquad (\forall 3 \le i \le d) \quad h_d \le a_i - a_{i-1}.$$

As an overview of this section, if there is a 3-cut isolating a set S with 6–10 vertices, at least one having degree 5, the algorithm splits on a vertex in the cut. Otherwise, the algorithm chooses a vertex u of degree 5 with — if possible — at least one neighbor of degree at most 4, and splits on u either by setting u to 0 and 1 or using the clause-learning splitting of Lemma 6.2.

8.1 3-cut If there is a 3-cut $C = \{x_1, x_2, x_3\}$ isolating a set S of vertices such that $6 \le |S| \le 10$ and S contains a vertex of degree 5 then splitting on x_1 leaves constraint graphs where $\{x_2, x_3\}$ form a 2-cut. Thus $S \cup \{x_1\}$ are removed from both resulting instances $(a_5 + 6a_3)$, a neighbor of x_1 outside $S \cup C$ has its degree reduced (h_5), a heavy edge x_2x_3 may appear but at least 2 half-edges incident on x_2 and x_3 disappear $(-w_h + w_e)$, and the resulting instances may become 5-regular $(-R_5)$. So, the measure decreases in both instances by at least $a_5 + 6a_3 + h_5 - w_h + w_e - R_5$. Constraint (3.6′) of Lemma 3.1 is thus assured if $2 \cdot 2^{-a_5 - 6a_3 - h_5 + w_h - w_e + R_5} \le 2^0 = 1$. We will henceforth express such constraints by saying that the case has *splitting number* at most

$$(8.26) \qquad \big(a_5 + 6a_3 + h_5 - w_h + w_e - R_5,\; a_5 + 6a_3 + h_5 - w_h + w_e - R_5\big).$$

From now on we may assume that each degree-5 variable u has $|N^2(u)| \ge 4$. $\boxdot$

8.2 5-regular If every vertex has degree 5, first consider the case in which F_0 and F_1 are 5-regular. Since splitting on u decreases the degree of each vertex in $N(u)$, and none of our reduction rules increases the degree of a vertex, every vertex in $N(u)$ must have been removed from F_0 and F_1 by simplification rules. This gives a splitting number of at most

$$(8.27) \qquad (6a_5, 6a_5).$$

If neither F_0 nor F_1 is 5-regular (R_5), then u is removed (a_5) and the degree of its neighbors decreases $(5h_5)$. Thus, the splitting number is at most

$$(8.28) \qquad (a_5 + 5h_5 + R_5, a_5 + 5h_5 + R_5).$$

If one of F_0 and F_1 is 5-regular, the resulting constraint is no stronger than (8.27) or (8.28). $\boxdot$

Otherwise, let u be a degree-5 vertex with a minimum number of degree-5 neighbors, p_i be the number of degree-i neighbors of u ($p_5 < 5$), and $H := \chi(u$ is incident to a heavy edge$)$. Depending on the values of H and p_i we will use either clause-learning splitting or normal 2-way splitting.

8.3 5-nonregular, 2-way splitting If $H = 1$ or $p_3 \ge 1$ or $p_5 \le 2$, we use the standard splitting, setting u to 0 and to 1 and simplifying to obtain F_0 and F_1. If F_i is not regular, the measure decrease is at least $\Delta_5^{\overline{r}} := a_5 + \sum_{i=3}^{5} p_i h_i + H(w_h - w_e)$, and if F_i is 5-regular, it is at least $\Delta_5^r := a_5 + \sum_{i=3}^{5} p_i a_i + H(w_h - w_e) - R_5$. If ≥ 1 branch is regular the splitting number is at most

$$(8.29) \qquad \big(\Delta_5^r,\ \Delta_5^r\big) \text{ or } \big(\Delta_5^r,\ \Delta_5^{\overline{r}}\big).$$

If both branches are nonregular, we use that any degree-3 neighbor of u either has a heavy edge not incident to u (additional measure decrease of $w_h - w_e$), or in at least one branch may be super 2-reduced (additional $2h_5$). At the start of the first super 2-reduction, every vertex has degree 2 or more. Each of the two "legs" of the super 2-reduction propagates through a (possibly empty) chain of degree-2 vertices before terminating either in a good degree reduction or by meeting a vertex that was reduced to degree 1 by the other leg. In the latter case all the vertices involved had degree 2, thus were neighbors of u originally of degree 3; also, there must have been at least three of them to form a cycle, and the remaining 2 or fewer vertices in $N(u)$ contradict the assumption that F was simplified. Thus, the splitting number is at most

$$\big(\Delta_5^{\overline{r}} + \chi(p_3 \ge 1)2h_5,\ \Delta_5^{\overline{r}}\big) \text{ or}$$
$$\big(\Delta_5^{\overline{r}} + \chi(p_3 \ge 1)(w_h - w_e),\ \Delta_5^{\overline{r}} + \chi(p_3 \ge 1)(w_h - w_e)\big).$$

$\boxdot$

8.4 5-nonregular, clause learning If $H = 0$ and $p_3 = 0$ and $p_5 \in \{3, 4\}$, let v be a degree-5 neighbor of u with a minimum number of degree-5 neighbors in $N^2 := N^2(u)$. The clause learning splitting (see Lemma 6.2) sets u in the 1st branch, u and v in the 2nd branch, and all of $N[u]$ in the 3rd branch. In each of the branches, the resulting instance could become 5-regular or not.

In the *1st branch*, the measure of the instance decreases by at least

$$\Delta_{51} := \min \begin{cases} a_5 + \sum_{i=4}^{5} p_i h_i & \text{(5-nonregular case)} \\ a_5 + \sum_{i=4}^{5} p_i a_i - R_5 & \text{(5-regular case).} \end{cases}$$

In the analysis of the 2nd and 3rd branch we distinguish the cases where v has ≤ 1 degree-5 neighbor in N^2, and where v (thus every degree-5 neighbor of u) has ≥ 2 degree-5 neighbors in N^2.

In the *2nd branch*, if v has ≤ 1 neighbor of degree 5 in N^2, the measure decreases by at least

$$\Delta_{52}^1 := \min \begin{cases} a_5 + \sum_{i=4}^5 p_i h_i + a_4 + 3h_4 + h_5, \\ a_5 + \sum_{i=4}^5 p_i a_i - R_5. \end{cases}$$

The degree reductions $3h_4 + h_5$ do not appear in the regular case as they may pertain to the same vertices as the deletions $\sum p_i a_i$. If v has ≥ 2 degree-5 neighbors in N^2, μ decreases by at least

$$\Delta_{52}^2 := \min \begin{cases} a_5 + \sum_{i=4}^5 p_i h_i + a_4 + 4h_5, \\ a_5 + \sum_{i=4}^5 p_i a_i + 2a_5 - R_5. \end{cases}$$

It is possible to show that in the *3rd branch*, the measure decreases by at least Δ_{53}^1 if v has ≥ 2 degree-5 neighbors in N^2 and by at least Δ_{53}^2 otherwise, where

$$\Delta_{53}^1 := \min \begin{cases} a_5 + \sum_{i=4}^5 p_i a_i + 4h_5 + \chi(p_5 = 4)h_5, \\ a_5 + \sum_{i=4}^5 p_i a_i + 4a_3 - R_5, \end{cases}$$

$$\Delta_{53}^2 := \min \begin{cases} a_5 + \sum_{i=4}^5 p_i a_i + 6h_5 + g_{p_4}, \\ a_5 + \sum_{i=4}^5 p_i a_i + 2a_5 + 2a_3 - R_5. \end{cases}$$

and $g_{p_4} := \chi(p_4 = 1) \cdot \min\{2h_5, -h_5 + h_4 + h_3\}$. Finally, the splitting number of this case is at most

$$(\Delta_{51}, \Delta_{52}^1, \Delta_{53}^1) \text{ or } (\Delta_{51}, \Delta_{52}^2, \Delta_{53}^2). \tag{8.30}$$

$\boxdot$

Acknowledgment The authors are very grateful to Alex Scott for initiating this project and contributing some of the first ideas.

References

[1] N. Bansal and V. Raman, *Upper bounds for MaxSat: Further improved*, Proc. ISAAC 1999, LNCS 1741, Springer, pp. 247–258.

[2] D. Eppstein, *Quasiconvex analysis of multivariate recurrence equations for backtracking algorithms*, ACM Trans. Algorithms **2** (2006), no. 4, 492–509.

[3] F. V. Fomin, F. Grandoni, and D. Kratsch, *Measure and conquer: Domination – a case study*, Proc. ICALP 2005, LNCS 3580, Springer, pp. 191–203.

[4] J. Gramm, E. A. Hirsch, R. Niedermeier, and P. Rossmanith, *Worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT*, Discrete Appl. Math. **130** (2003), no. 2, 139–155.

[5] E. A. Hirsch, *A new algorithm for MAX-2-SAT*, Proc. STACS 2000, LNCS 1770, Springer, pp. 65–73.

[6] J. Kneis, D. Mölle, S. Richter, and P. Rossmanith, *Algorithms based on the treewidth of sparse graphs*, Proc. WG 2005, LNCS 3787, Springer, pp. 385–396.

[7] J. Kneis and P. Rossmanith, *A new satisfiability algorithm with applications to Max-Cut*, Tech. Report AIB-2005-08, Department of Computer Science, RWTH Aachen, 2005.

[8] M. Koivisto, *Optimal 2-constraint satisfaction via sum-product algorithms*, Inf. Proc. Letters **98** (2006), no. 1, 24–28.

[9] A. Kojevnikov and A. S. Kulikov, *A new approach to proving upper bounds for MAX-2-SAT*, Proc. SODA 2006, ACM, pp. 11–17.

[10] A. S. Kulikov and K. Kutzkov, *New bounds for MAX-SAT by clause learning*, Proc. CSR 2007, LNCS 4649, Springer, pp. 194–204.

[11] O. Kullmann, *New methods for 3-SAT decision and worst-case analysis*, Theoret. Comput. Sci. **223** (1999), no. 1-2, 1–72.

[12] ———, *Worst-case analysis, 3-SAT decision and lower bounds: Approaches for improved SAT algorithms*, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 35, Amer. Math. Soc., 1997, pp. 261–313.

[13] R. Niedermeier and P. Rossmanith, *New upper bounds for maximum satisfiability*, J. Algorithms **36** (2000), no. 1, 63–88.

[14] D. Raible and H. Fernau, *A new upper bound for Max-2-SAT: A graph-theoretic approach*, Tech. Report cs:DS/0803.3531v2, arxiv.org, 2008, see http://arxiv.org/abs/cs.DM/0803.3531.

[15] A. D. Scott and G. B. Sorkin, *Faster algorithms for MAX CUT and MAX CSP, with polynomial expected time for sparse instances*, Proc. RANDOM 2003, LNCS 2764, Springer, pp. 382–395.

[16] ———, *A faster exponential-time algorithm for Max 2-Sat, Max Cut, and Max k-Cut*, Tech. Report RC23456 (W0412-001), IBM Research Report, 2004, see http://domino.research.ibm.com/library/cyberdig.nsf.

[17] ———, *Linear-programming design and analysis of fast algorithms for Max 2-CSP*, Discrete Optim. **4** (2007), no. 3-4, 260–287.

[18] M. Wahlström, *Exact algorithms for finding minimum transversals in rank-3 hypergraphs*, J. Algorithms **51** (2004), no. 2, 107–121.

[19] R. Williams, *A new algorithm for optimal 2-constraint satisfaction and its implications*, Theoret. Comput. Sci. **348** (2005), no. 2-3, 357–365.