New exact algorithms for the 2-constraint satisfaction problem

Alexander Golovnev*[1] and Konstantin Kutzkov[2]

[1]New York University
[2]IT University of Copenhagen

Abstract

Many optimization problems can be phrased in terms of constraint satisfaction. In particular MAX-2-SAT and MAX-2-CSP are known to generalize many hard combinatorial problems on graphs. Algorithms solving the problem exactly have been designed but the running time is improved over trivial brute-force solutions only for very sparse instances. Despite many efforts, the only known algorithm [28] solving MAX-2-CSP over n variables in less than $O^*(2^n)$ steps uses exponential space.

Several authors have designed algorithms with running time $O^*(2^{nf(d)})$ where $f : \mathbb{R}^+ \to (0, 1)$ is a slowly growing function and d is the average variable degree of the input formula. The current best known algorithm for MAX-2-CSP [25] runs in time $O^*(2^{n(1-\frac{2}{d+1})})$ and polynomial space. In this paper we continue this line of research and design new algorithms for the MAX-2-SAT and MAX-2-CSP problems.

First, we present a general technique for obtaining new bounds on the running time of a simple algorithm for MAX-2-CSP analyzed with respect to the number of vertices from algorithms that are analyzed with respect to the number of constraints. The best known bound for the problem is improved to $O^*(2^{n(1-\frac{3}{d+1})})$ for $d \geq 3$. We further improve the bound for MAX-2-SAT, in particular for $d \geq 6$ we achieve $O^*(2^{n(1-\frac{3.677}{d+1})})$.

As a second result we present an algorithm with asymptotically better running time for the case when the input instance is not very sparse. Building on recent work of Feige and Kogan we derive an upper bound on the size of a vertex separator for graphs in terms of the average degree of the graph. We then design a simple algorithm solving MAX-2-CSP in time $O^*(2^{c_d n})$, $c_d = 1 - \frac{2\alpha \ln d}{d}$ for some $\alpha < 1$ and $d = o(n)$.

Keywords: exact exponential time algorithms, constraint satisfaction, maximum satisfiability

1 Introduction

1.1 Exponential time algorithms

A paradigm in computational complexity was to divide problems into efficiently solvable and hard problems, in more precise terms NP-hard or #P-hard. Since hard problems are unlikely to be solvable exactly in polynomial time, researchers started to design *approximation* algorithms such that one can efficiently find a solution which is reasonably good. However, for many hard problems, approximation algorithms are either inadequate or do not yield satisfactory results. For example, unless P=NP no polynomial time algorithm can approximate MAX-2-SAT with a factor better than 0.9546 [13].

Hard problems were deemed intractable and except for a few results [14, 16, 21, 26], not much work was invested in the design of exponential time algorithms improving upon the naïve solution. However, the increase in computational power and available memory led to a shift in these attitudes. The last two decades have witnessed a growing interest in design and analysis of deterministic and randomized exponential time

*Part of this work was done when the first author was at St. Petersburg University of the Russian Academy of Sciences.

algorithms. New algorithms with considerably improved running times have increased the size of efficiently computable instances of hard problems. Notable examples include solving k-COLORABILITY for graphs on n vertices in time and space $O^*(2^n)$[1] independent of k [3] and k-SATISFIABILITY on n variables in time $O^*((\frac{2(k-1)}{k})^n)$ and polynomial space [23, 24].

1.2 Problem Statement

The maximum satisfiability problem (MAX-SAT) is: given a Boolean formula in conjunctive normal form (CNF), i.e. a conjunction of literal disjunctions, find the maximum number of simultaneously satisfiable clauses of this formula. MAX-2-SAT is a restricted version of MAX-SAT, where each clause contains at most two literals.

MAX-2-SAT is a special case of the maximum 2-constraint satisfaction problem (MAX-2-CSP). In the MAX-2-CSP problem one is given a graph $G = (V, E)$ along with sets of "weight" functions $S_v : \{0, 1\} \to \mathbb{Z}$ for each vertex $v \in V$ and $S_e : \{0, 1\}^2 \to \mathbb{Z}$ for each edge $e \in E$. (We will assume that the weight functions can be evaluated and represented in polynomial time and space.) The goal is to find an assignment $\phi : V \to \{0, 1\}$ maximizing the sum

$$\sum_{e=(v_1,v_2)\in E} S_e(\phi(v_1), \phi(v_2)) + \sum_{v\in V} S_v(\phi(v)). \tag{1}$$

It is easy to see that by associating variables with vertices and clauses with edges, MAX-2-SAT corresponds to the case when all functions from S_e are disjunctions and each vertex $v \in V$ is assigned a weight of 0 by S_v. In the following an *instance* of MAX-2-CSP will be described by the graph G and the set of weight functions $S = S_v \cup S_e$, and an instance of MAX-2-SAT will be given only by the Boolean formula F since the weight function does not need to be explicitly defined.

MAX-SAT and MAX-2-SAT are among the most famous NP-hard optimization problems generalizing many graph problems. As already mentioned, the problem is hard to approximate with a factor better than 0.9546. Moreover, there is no known algorithm for either problem with polynomial space and running time less than $O^*(2^n)$, i.e. a running time of the form $O^*((2 - \varepsilon)^n)$ for a constant $\varepsilon > 0$. The strong exponential time hypothesis [4, 17] implies that MAX-SAT cannot be solved in time less than $O^*(2^n)$.

1.3 The Main Definitions

Let $G = (V, E)$ be an undirected graph[2]. In the following we give definitions only for MAX-2-CSP in terms of graph terminology. By associating vertices with Boolean variables and edges with 2-clauses, i.e. disjunctions of two variables or their negations, the definitions trivially extend to Boolean formulas in CNF form.

By $n(G), m(G)$ we denote, respectively, the number of vertices and the number of edges in G. The size of G is defined as $|G| = m(G) + n(G)$. By the degree $deg(x)$ of a vertex x we mean the number of edges incident to x. We say that a vertex y is the neighbor of a vertex x if there is an edge $(x, y) \in G$. By $\Delta(G)$ we denote the maximum vertex degree in G. $d(G) = 2m/n$ is the average vertex degree. We omit G if it is clear from the context.

Note that in the case of MAX-2-CSP one can assume without loss of generality that the corresponding graph does not contain multiple edges (as any two parallel edges can be replaced by their "sum"). At the same time one cannot exclude multiple edges from a MAX-2-SAT graph by the same argument (e.g., the graph of a formula $(x \vee y)(\neg x \vee y)(y \vee z)$ has two edges between x and y).

By (n, Δ)-MAX-2-CSP we denote MAX-2-CSP problems restricted to instances in which each vertex has degree at most Δ. By $Opt(G, S)$ we denote the maximal value of (1) for (G, S) over all possible assignments $\phi : V \to \{0, 1\}$. Similarly we define (n, Δ)-MAX-2-SAT for a Boolean formula F, and define $Opt(F)$ to be the maximal number of simultaneously satisfiable clauses of the formula F.

[1] The O^* notation ignores polynomial factors.

[2] Note that we allow loops and multiple edges.

Let (G, S) be an instance of MAX-2-CSP, and v be a vertex in G. By $(G, S)[v]$, or simply $G[v]$ when clear from the context, we denote the instance resulting from replacing all occurrences of v by 1 in (G, S). Similarly, for $G[\neg v]$ we replace v by 0. The definition naturally transfers to MAX-2-SAT, clauses containing a given literal l or $\neg l$ are either satisfied or shortened, i.e., 2-clauses become 1-clauses and 1-clauses are removed. Recursively solving a MAX-2-CSP problem instance by considering the cases $G[v]$ and $G[\neg v]$ for a vertex v, is referred to as *splitting* or *branching* on v.

A *vertex cover* of a graph $G = (V, E)$ is a subset of vertices $C \subseteq V$ such that for each $(u, v) \in E$, $u \in C$ or $v \in C$ holds. The complement set $V \backslash C$ is an *independent set* in G. Two sets of vertices $U \subset V, W \subset V$ are called *pairwise independent* if $U \cap W = \emptyset$ and there exist no edges $(u, w) \in E$ with $u \in U, w \in W$. We will call a set $\mathcal{S} \subseteq V$ of vertices *balanced graph separator of cardinality* k if $V \backslash \mathcal{S} = (U, W)$ such that (1) U and W are pairwise independent and (2) $|U| = \lfloor \frac{n-k}{2} \rfloor$ and $|W| = \lceil \frac{n-k}{2} \rceil$.

1.4 Known Results

Running time	Problem	Authors	Year
	with respect to n		
$2^{\frac{\omega n}{3}}$, exp. space	MAX-2-CSP	Williams [28]	2005
$c^n, c < 2$	MAX-SAT with constant density	Kulikov, Kutzkov [19]	2009
	With respect to n and d		
$2^{n(1-\frac{2}{d+1})}$	MAX-2-CSP	Scott, Sorkin [25]	2007
$2^{n(1-\frac{2}{\Delta})}$	MAX-CUT	Della Croce, Kaminski, Paschos [6]	2007
$2^{\frac{n}{6.7}}$	$(n, 3)$-MAX-2-SAT	Kulikov, Kutzkov [19]	2009
	With respect to m		
$2^{\frac{m}{2.465}}$	MAX-SAT	Chen, Kanj [5]	2004
$2^{\frac{m}{6.321}}$	MAX-2-SAT	Gaspers, Sorkin [10]	2012
$2^{\frac{m}{5.263}}$	MAX-2-CSP	Gaspers, Sorkin [10]	2012

Table 1: Known upper bounds for MAX-2-SAT and MAX-2-CSP

In a ground-breaking work Williams [28] presented an algorithm for MAX-2-CSP with running time $O^*(2^{\frac{\omega n}{3}})$ where ω is the matrix multiplication exponent. Currently the best known bound is $\omega < 2.3727$ [27]. Unfortunately, the algorithm runs in $\Theta(2^{\frac{2n}{3}})$ space and this makes it intractable in practice. It is still an open question whether MAX-2-SAT can be solved in less than $O^*(2^n)$ steps using polynomial memory. However, the trivial 2^n upper bound was improved for several special cases of the considered problems. Dantsin and Wolpert [7] showed that MAX-SAT for formulas with constant clause density, i.e. the ratio of clauses to variables is bounded by a constant, can be solved in faster than $O(2^n)$ time using exponential space. Kulikov and Kutzkov [19] developed an algorithm for MAX-SAT using polynomial space and with running time $O^*(2^{c_\rho n})$ for formulas with constant clause density ρ, where $c_\rho < 1$ is a constant depending on ρ. Note that unlike in [7] the running time does not admit a closed form expression.

Several results on polynomial-space algorithms for MAX-2-CSP and some of its special cases have been achieved. Fürer and Kasiviswanathan [9] designed an algorithm for MAX-2-CSP with running time $O^*(2^{n(1-\frac{1}{d-1})})$. Scott and Sorkin [25] improved this bound to $O^*(2^{n(1-\frac{2}{d+1})})$. (Note that the bound is better than the one shown by Fürer and Kasiviswanathan since $\frac{1}{d-1} \leq \frac{2}{d+1}$ for $d \geq 3$ and, as we discuss in Section 2, we can restrict our attention to the case $d \geq 3$.) For the widely studied MAX-CUT problem, a special case of MAX-2-CSP, Della Croce, Kaminski and Paschos [6] developed an algorithm with running time $O^*(2^{n(1-\frac{2}{\Delta})})$ for graphs of degree bounded by Δ. Their algorithm is better than the one by Scott and

Sorkin for nearly regular input graphs when $\Delta < d + 1$. For $(n, 3)$-MAX-2-SAT, Kojevnikov and Kulikov [18] proved an $O^*(2^{\frac{n}{6}})$ bound. This was later improved to $O^*(2^{\frac{n}{6.7}})$ by Kulikov and Kutzkov [19].

The best known upper bound in terms of the number of clauses m for MAX-SAT, $O^*(2^{\frac{m}{2.465}})$, was given by Chen and Kanj [5]. In a long sequence of works the best known upper bounds for MAX-2-SAT have been improved [15, 12, 18, 25, 19, 2, 10]. The current record holders are Gaspers and Sorkin [10] who proved that MAX-2-SAT and MAX-2-CSP can be deterministically solved in time $O(2^{\frac{m}{6.321}})$ and $O(2^{\frac{m}{5.263}})$ time respectively. Note that these bounds are better than the brute-force solution only for very sparse instances. Table 1 summarizes the best known bounds with respect to different parameters.

1.5 New Upper Bounds

In this paper, we present two simple algorithms that achieve new upper bounds on the complexity of MAX-2-SAT and MAX-2-CSP when only polynomial space is allowed.

The first algorithm solves MAX-2-CSP in time $O^*(2^{n(1-\frac{3}{d+1})})$, which improves over the best known algorithm from [25] running in time $O^*(2^{n(1-\frac{2}{d+1})})$. We improve this bound for MAX-2-SAT to $O(2^{n(1-\frac{3.677}{d+1})})$ for $d \geq 6$. For $3 \leq d < 6$ our bound is at least $O(2^{n(1-\frac{c_d}{d+1})})$ with $c_d \geq 3.4042$ which grows monotonically with d. Since MAX-CUT is a special case of MAX-2-CSP, we also get an improved upper bound of $O^*(2^{n(1-\frac{3}{d+1})})$ for MAX-CUT.

Our second algorithm presents an asymptotically better bound for MAX-2-CSP for the case when the underlying constraint graph is neither very sparse nor very dense. It follows the same simple idea as the first algorithm but we make use of graph separators of bounded size whose existence we show by building on recent result by Feige and Kogan [8]. We achieve running time of $O(2^{c_d n})$ where $c_d = 1 - \frac{2\alpha \ln d}{d}$ for constant $\alpha < 1$.

A preliminary version of this work was presented at the 6th International Symposium on Parameterized and Exact Computation [11]. In the full version of the paper we improve and simplify the analysis of the first algorithm and also present the second algorithm.

2 An algorithm for MAX-2-CSP

In this section, we present a simple algorithm for MAX-2-SAT and MAX-2-CSP. To solve an instance of average degree d, the algorithm branches on a variable of maximum degree until we obtain an instance of low average degree. Then, we solve the problem by a known algorithm for MAX-2-SAT or MAX-2-CSP analyzed with respect to the number of clauses or edges, respectively.

2.1 Intuition

Before formally presenting our algorithm in detail let us give a general overview on how it works and why we are able to achieve better bounds than previously known algorithms. Let us recall how the simple algorithm by Scott and Sorkin [25] works. Building on a result by Alon, Kahn and Seymour [1] for the size of an induced forest in a graph of average degree $d \geq 2$, the authors use the fact that MAX-2-CSP is solvable in polynomial time when the underlying constraint graph G is a forest. The observation easily follows from the fact (see next section) that in MAX-2-CSP a vertex of degree at most 2 can be removed from G without splitting. This simplification rule has played a crucial role for achieving improved upper bounds for MAX-2-SAT and MAX-2-CSP with respect to the number of variables. However, the removal of variables of degree at most 2 is not the only tool used in the design of exact algorithms for MAX-2-SAT. The best currently known upper bounds are obtained through sophisticated algorithms using a plethora of simplification rules and the algorithms are analyzed with respect to a suitably chosen non-standard complexity measure. Some of these simplifications, for example clause learning [19], are specific to MAX-2-SAT and it is not clear how to use them for algorithms for MAX-2-CSP. The basic idea of the new algorithm is to make use of all these advanced techniques without explicitly using them. More concretely, we will branch on a variable of high degree and in this way we will eliminate all of its incident edges. We will show that in this way the formula

becomes sparser and sparser, and at some point it will be sparse enough such that the best known algorithms analyzed with respect to the number of edges or clauses become more efficient than the brute-force splitting algorithm. In this way we can design a simple algorithm using as a black-box a sophisticated splitting algorithm. There are better algorithms for solving MAX-2-SAT than for solving MAX-2-CSP w.r.t. m, and hence we get better bounds for MAX-2-SAT.

2.2 Tools used in the analysis

Jensen's inequality. We use the following version of Jensen's inequality:

Theorem 1 *If f is concave function, $x_1, x_2, \ldots, x_n$ are in its domain, and $\forall i, n_i > 0$, then:*

$$\frac{\sum n_i f(x_i)}{\sum n_i} \leq f(\frac{\sum n_i x_i}{\sum n_i}).$$

Corollary 1 *For any positive c, n_i, if $\alpha_i \leq 1 - \frac{c}{i+1}$ and $\sum n_i = n$, then*

$$n_3\alpha_3 + \ldots + n_\Delta\alpha_\Delta \leq n(1 - \frac{c}{d+1}),$$

where $d = \frac{\sum in_i}{n}$.

Proof: It is easy to see that for any $c > 0$, $f(x) = 1 - \frac{c}{x+1}$ is concave function for $x \geq 3$. By Theorem 1,

$$n_3\alpha_3 + \ldots + n_\Delta\alpha_\Delta \leq (\sum n_i) \cdot f(\frac{\sum n_i i}{\sum n_i}) = nf(d) = n(1 - \frac{c}{d+1}).$$

□

Removing variables of degree 2.

Lemma 1 *Let F be an instance of MAX-2-SAT or MAX-2-CSP containing a vertex u of degree at most 2. Then F can be transformed in polynomial time into a formula F' s.t.*

1. *$deg_{F'}(u) = 0$,*
2. *for all v, $deg_{F'}(v) \leq deg_F(v)$,*
3. *$Opt(F)$ can be computed from $Opt(F')$ in polynomial time.*

This lemma is proved for MAX-2-SAT in [18, Lemma 3.1], and for MAX-2-CSP in [10, Section 5.9]. It allows us to assume that a simplified formula only contains variables of degree at least 3.

Frequently co-occurring variables. The following lemma, specific to the MAX-2-SAT problem, has been proved in [18].

Lemma 2 *Let F be a 2-CNF formula and x and y two variables occurring in it, such that x occurs in at most one clause without y. Then we can obtain in polynomial time a formula F' such that $deg_{F'}(x) = 0$, for all variables $y \in F \backslash \{x\}$, $deg_{F'}(y) \leq deg_F(y)$ and $Opt(F)$ can be computed from $Opt(F')$ in polynomial time.*

Non-standard complexity measures. Let us consider the MAX-2-SAT problem for a 2-CNF formula F over m 2-clauses and n variables. The standard complexity measure *number of clauses* can be written as

$$m(F) = \sum_{i=1} \frac{in_i}{2}$$

where n_i denotes the number of variables of degree i.

Kojevnikov and Kulikov [18] observed that the following alternative measure for MAX-2-SAT:

$$\mu(F) = \sum_{i=1} \alpha_i n_i$$

for $\alpha_i \leq i/2$, is more suitable for the analysis of the running time of exact splitting algorithms. Due to various simplification rules, variables of low degree allow the elimination of more 2-clauses. Hence the variables of low degree contribute more to a better running time than variables of higher degree. For the new MAX-2-SAT algorithm we will also use a similar measure so that we can obtain an improved bound on the running time of the algorithm.

2.3 Meta-Algorithm

Let us first consider the case when the constraint graph is of bounded degree. We present a generic algorithm for (n, Δ)-MAX-2-CSP. The algorithm branches on a vertex of maximum degree Δ until it gets a graph of maximum degree $\Delta - 1$. It then calls a known algorithm for $(n, \Delta - 1)$-MAX-2-CSP. Although the presented algorithm works for both MAX-2-SAT and MAX-2-CSP, we use MAX-2-CSP notation for simplicity.

Denote by n_i the number of vertices of degree i for $i \in \{3, \ldots, \Delta\}$. Consider a problem instance of (n, Δ)-MAX-2-CSP given as a graph $G = (V, E)$ and weight functions $S = S_v \cup S_e$. We use the following complexity measure:

$$\mu(G) = \alpha_3 n_3 + \ldots + \alpha_\Delta n_\Delta,$$

where α_i denotes the weight of a vertex of degree i. The values of α_i will be determined later. We would like to find α_i such that for any problem instance (G, S) we can show that the algorithm has running time bounded by $poly(|G|) \cdot 2^{\mu(G)}$.

Assume that an algorithm A solves $(n, \Delta - 1)$-MAX-2-CSP in time $2^{\alpha_3 n_3 + \ldots + \alpha_{\Delta-1} n_{\Delta-1}}$. Consider the following algorithm for (n, Δ)-MAX-2-CSP.

Algorithm 1 MetaAlg — a template for the new algorithms.

Parameter: Algorithm A for $(n, \Delta - 1)$-MAX-2-CSP.
Input: (G, S) – instance of MAX-2-CSP.
Output: $Opt(G, S)$.

1: remove all vertices of degree < 3 (using Lemma 1).
2: **if** G contains no edges **then**
3: **return** the result.
4: **if** the maximum vertex degree of G is less than Δ **then**
5: **return** $A(G, S)$.
6: choose a vertex x of maximum degree Δ.
7: **return** $\max(\text{MetaAlg}(A, G[x]), \text{MetaAlg}(A, G[\neg x]))$.

Lemma 3 *Let $\Delta > 3$, $\alpha_i < 1$, for all i. If* A *solves $(n, \Delta - 1)$-MAX-2-CSP in time $2^{\alpha_3 n_3 + \ldots + \alpha_{\Delta-1} n_{\Delta-1}}$ and*

$$\delta = \min(\alpha_\Delta - \alpha_{\Delta-1}, \alpha_{\Delta-1} - \alpha_{\Delta-2}, \ldots, \alpha_4 - \alpha_3, \alpha_3) \geq \frac{1 - \alpha_\Delta}{\Delta}, \tag{2}$$

then the running time of the algorithm MetaAlg *for (n, Δ)-MAX-2-CSP is $2^{\alpha_3 n_3 + \ldots + \alpha_\Delta n_\Delta}$.*

Proof: Denote by $T(n_3, \ldots, n_\Delta)$ the running time of the algorithm on a graph that has n_i vertices of degree i, for all $3 \leq i \leq \Delta$. If there are no vertices of degree Δ (i.e., $n_\Delta = 0$), then MetaAlg just calls A. Then, clearly,

$$T(n_3, \ldots, n_\Delta) \leq 2^{\alpha_3 n_3 + \ldots + \alpha_{\Delta-1} n_{\Delta-1}} = 2^{\alpha_3 n_3 + \ldots + \alpha_\Delta n_\Delta}.$$

Now assume that there exists a vertex x of degree Δ. Then MetaAlg branches on a vertex of degree Δ at step 6. We show that in both branches $G[x]$ and $G[\neg x]$, μ is reduced by at least 1.

Indeed, the measure decreases by α_Δ, because the algorithm branches on a vertex of degree Δ. The degree of each neighbor of x is reduced, so μ is decreased by at least δ (as δ is the minimal amount by which μ is decreased when the degree of a vertex is reduced). This causes a complexity decrease of $\Delta \cdot \delta$. Lemma 1 guarantees that removing variables of degree 2 does not increase μ. It follows from (2) that $\Delta \cdot \delta + \alpha_\Delta \geq 1$. Therefore μ decreases by at least 1. Then

$$T(n_3, \ldots, n_\Delta) \leq 2 \cdot 2^{\alpha_3 n_3 + \ldots + \alpha_\Delta n_\Delta - 1} + poly(|G|) \leq 2^{\alpha_3 n_3 + \ldots + \alpha_\Delta n_\Delta} + poly(|G|).$$

Thus, the running time of the algorithm MetaAlg is $O^*(2^{\alpha_3 n_3 + \ldots + \alpha_\Delta n_\Delta})$. □

We will use the above "template" for two new algorithms for MAX-2-SAT and MAX-2-CSP. The only difference will be the algorithm A used as a subroutine. For the running time analysis we will apply Jensen's inequality as shown in Corollary 1 to estimate the running time in terms of the average degree.

2.4 An algorithm for MAX-2-CSP

The fact that vertices of degree at most 2 can be removed without splitting implies that $(n, 3)$-MAX-2-CSP can be solved in $O^*(2^{n/4})$ time. Indeed, when branching on a vertex of degree 3 we can remove all its neighbors in both branches (so, the number of vertices is decreased by at least 4). Denote by CubicAlg the above sketched algorithm.

Algorithm 2 Max2CSPAlg — solving exactly the MAX-2-CSP problem over n vertices.

Input: (G, S) – instance of MAX-2-CSP.
Output: $Opt(G, S)$.

```
1: while there is a vertex v of degree less than 3 do
2:    remove v (using Lemma 1).
3: if G contains no 2-clauses then
4:    evaluate the scoring S_v functions for each vertex.
5:    return the result.
6: choose a vertex x of maximum degree Δ.
7: if d(x) = 3 then
8:    return CubicAlg(G, S).
9: else
10:   return max(Max2CSPAlg(G[x], S), Max2CSPAlg(G[¬x], S)).
```

Theorem 2 *There exists a deterministic polynomial space algorithm solving MAX-2-CSP in time* $O^*(2^{n(1-\frac{3}{d+1})})$.

Proof: In the following we will show that Max2CSPAlg solves MAX-2-CSP in time $O^*(2^{n(1-\frac{3}{d+1})})$.

Max2CSPAlg is obtained from MetaAlg. Max2CSPAlg takes itself as parameter A if $i > 3$ and takes CubicAlg if $i = 3$. We will choose α_i satisfying (2). As mentioned above, $(n, 3)$-MAX-2-CSP can be solved by CubicAlg in $O^*(2^{n/4})$ time. Hence, we can choose $\alpha_3 = \frac{1}{4}$. Now let $\Delta > 3$. For (2) to hold, we can set α_i as follows:

$$\alpha_i = \frac{1 + i \cdot \alpha_{i-1}}{i+1} = \frac{i - 3 + 4\alpha_3}{i+1} = 1 - \frac{3}{i+1}.$$

We show that (2) holds. First, note that $\frac{1-\alpha_\Delta}{\Delta} \leq \alpha_3 = \frac{1}{4}$, for $\Delta \geq 4$.

$$\frac{1-\alpha_\Delta}{\Delta} = \frac{3}{\Delta(\Delta+1)} < \frac{1}{4}.$$

Now observe that $\frac{1-\alpha_\Delta}{\Delta} \leq \alpha_i - \alpha_{i-1}$ for $i \leq \Delta$.

$$\frac{1-\alpha_\Delta}{\Delta} = \frac{3}{\Delta(\Delta+1)} \leq \frac{3}{i(i+1)} = \alpha_i - \alpha_{i-1}.$$

By Lemma 3 the running time of the algorithm is $O^*(2^{\alpha_3 n_3 + \ldots + \alpha_\Delta n_\Delta})$. From $\alpha_i = 1 - \frac{3}{i+1}$ and Corollary 1 it follows that the running time of the algorithm is $O^*(2^{n(1-\frac{3}{d+1})})$. □

2.5 An Algorithm for MAX-2-SAT

We further improve the bound for MAX-2-SAT, using algorithms analyzed with respect to the number of clauses for instances of small average degree. We use the analysis outlined in Section 2.3. We can again remove variables of degree at most two without splitting, but there is a subtle difference compared to the MAX-2-CSP problem. In MAX-2-CSP one can assume that all neighbors of a given vertex are different since multi-edges can be unified in a single constraint over two variables. On the other hand in a 2-CNF formula a given variable can have the same neighbor more than once. Thus, by simply removing variables of degree at most two we cannot assure that by assigning a Boolean value to a variable x (2) will hold for the decrease in all of x's neighbors. Fortunately, by Lemma 2 we can assume that each of v's neighbors occurs at least two times outside of v's neighborhood. Thus, (2) will hold again implying a complexity decrease.

The algorithm by Binkele-Raible and Fernau [2], in the following denoted as BRF, turns out to be suitable for our goals. It improves upon the approach by Kojevnikov and Kulikov [18] by performing a careful analysis of the possible cases, applying recently developed simplification rules like clause learning [19] and using the best known bound for $(n,3)$-MAX-2-SAT from [19]. The running time they achieve can be written as $2^{\frac{1}{6.265}\sum w_i n_i}$, where $w_3 = 0.9354, w_4 = 1.8230, w_5 = 2.4779$ and $w_i = \frac{i}{2}$ for $i > 5$.

The best known upper bound for MAX-2-SAT however is $O(2^{m/6.321})$ [10]. The complexity measure in [10] does not have the convenient form of the measure from [2]. Moreover, it does not use the best known bound for $(n,3)$-MAX-2-SAT from [19] and thus it does not achieve results as good for formulas of small average degree. In order to improve upper bounds for graphs of large average degree, we use the algorithm by Gaspers and Sorkin [10], denoted as GS. In Max2SATAlg we decide which algorithm to apply, GS or BRF, depending on the *average* variable degree. Then, in Max2SATAlgParam, we simplify the formula, if possible, and recursively branch on a variable of maximum degree until there exists a variable of degree more than six. Once the *maximum* variable degree is at most six, we apply the selected algorithm to the formula.

Algorithm 3 Max2SATAlg.

Input: F – instance of MAX-2-SAT.
Output: $Opt(F)$.

1: **if** the average variable degree $d(F) \leq 6$ **then**
2: $A =$ BRF.
3: **else**
4: $A =$ GS.
5: **return** Max2SATAlgParam(A, F).

Theorem 3 *Let F be a Boolean formula over n variables with average degree $d \geq 3$. Let $\alpha_3 = 0.1494, \alpha_4 = 0.2909, \alpha_5 = 0.3956, \alpha_6 = 0.4789$. Then the algorithm* Max2SATAlg *solves MAX-2-SAT in time $O(2^{(\alpha_{i+1}-\alpha_i)d+(i+1)\alpha_i - i\alpha_{i+1}})$ for $3 \leq d < 6, i = \lfloor d \rfloor$, and in time $O(2^{n(1-\frac{3.677}{d+1})})$ for $d \geq 6$.*

Algorithm 4 MAX2SATALGPARAM.

Input: Algorithm A for $(n,6)$-MAX-2-SAT, F – instance of MAX-2-SAT.
Output: $Opt(F)$.

1: remove all variables of degree < 3 by Lemma 1 and all frequently co-occurring variables by Lemma 2.
2: **if** F contains no 2-clauses **then**
3: **return** the result.
4: **if** the maximum variable degree of $F \leq 6$ **then**
5: **return** $A(F)$.
6: choose a vertex x of maximum degree Δ.
7: **return** $\max(\text{MAX2SATALGPARAM}(A, F[x]), \text{MAX2SATALGPARAM}(A, F[\neg x]))$.

Proof: First we analyze the running time of MAX2SATALGPARAM for the case $d \leq 6$, i.e. $A = \text{BRF}$. Note that in this case the maximum degree Δ might be greater than 6. We observe that we can set

$$\alpha_3 = 0.1494 > \frac{0.9354}{6.265}, \alpha_4 = 0.2909 > \frac{1.8230}{6.265}, \alpha_5 = 0.3956 > \frac{2.4779}{6.265}, \alpha_6 = 0.4789 > \frac{6}{2 \cdot 6.265}.$$

For the case that $\Delta \leq 6$ we do not need to satisfy (2) since Binkele-Raible and Fernau have shown that for the given values of α_i for all possible cases the algorithm may have to handle, the running time is bounded by $O(2^{\mu(F)})$. For $i > 6$ we set the weights α_i such that (2) holds:

$$\alpha_i = \frac{1 + i \cdot \alpha_{i-1}}{i+1} = 1 - \frac{3.6477}{i+1}.$$

If the maximum vertex degree $\Delta \leq 6$, then MAX2SATALGPARAM solves MAX-2-SAT in $O(2^{\alpha_3 n_3 + \ldots + \alpha_\Delta n_\Delta})$ time by [2]. Now we prove that (2) holds for $\Delta > 6$. Indeed,

$$\delta = \min(\alpha_\Delta - \alpha_{\Delta-1}, \alpha_{\Delta-1} - \alpha_{\Delta-2}, \ldots, \alpha_4 - \alpha_3, \alpha_3) =$$
$$\min(\alpha_\Delta - \alpha_{\Delta-1}, \alpha_{\Delta-1} - \alpha_{\Delta-2}, \ldots, \alpha_7 - \alpha_6, 0.0833) = \alpha_\Delta - \alpha_{\Delta-1} = \frac{3.6477}{\Delta(\Delta+1)} = \frac{1-\alpha_\Delta}{\Delta}.$$

Lemma 3 implies that the running time of the algorithm is $2^{\alpha_3 n_3 + \ldots + \alpha_\Delta n_\Delta}$.

Now we generalize to the case when the formula has average degree $d \geq 3$, i.e. $\sum_{i=3} i n_i = dn$. We define a function $f(x)$ with straight lines connecting points α_i, α_{i+1} for $3 \leq i < 6$ and for $x \geq 6$ we set $f(x)$ according to (2):

$$f(x) = \begin{cases} (\alpha_{i+1} - \alpha_i)x + (i+1)\alpha_i - i\alpha_{i+1} & \text{for } x \in [i, i+1),\ 3 \leq i < 6 \\ 1 - \frac{3.6477}{x+1} & \text{for } x \geq 6 \end{cases}.$$

It is easy to see that for the given values of the α_i's the function is piecewise-linear concave for $3 \leq x < 6$ and concave for $x \geq 6$. Thus, Corollary 1 implies that $\alpha_3 n_3 + \ldots + \alpha_\Delta n_\Delta \leq n f(d)$. This yields the claimed running time.

The algorithm by Gaspers and Sorkin [10] solves MAX-2-SAT in time $O^*(2^{m/6.321}) = O^*(2^{\sum_{i>2} \frac{i n_i}{2 \cdot 6.321}})$. Therefore, we analyze the running time of MAX2SATALGPARAM for $A = \text{GS}$ by setting $\alpha_i = \frac{i}{2 \cdot 6.321}$ for $3 \leq i < 6$. This yields the measure $\mu(F) = \sum_{i \geq 3} \alpha_i n_i$ on the complexity of solving MAX-2-SAT w.r.t. n. We see that for $i \geq 6$ one can set $\alpha_i = 1 - \frac{3.677}{i+1}$ such that (2) holds for branching on a variable of degree ≥ 6. Connecting again the α_i for $i \in \{3,4,5,6\}$ by straight lines and for $x \geq 6$ setting $f(x) = 1 - \frac{3.677}{x+1}$ we obtain a concave function and by applying Corollary 1 the claimed running time follows for the case $d \leq 6$. □

We would like to note that Theorem 3 naturally extends to obtaining a bound on the complexity of the MAX-2-SAT problem with respect to the number of variables from an upper bound on the problem with

respect to the number of clauses. In particular, given an algorithm with running time of $O^*(2^{m/c})$, $c > 1$, and set $\ell = \lfloor c \rfloor$. We then set $\alpha_i = \frac{i}{2c}$ for $i \leq \ell$ and $\alpha_i = 1 - \frac{(\ell+1)(1-\alpha_\ell)}{i+1}$ for $i > \ell$. However, as we have seen, this is not guaranteed to be the optimal result for arbitrary average degree d and a more careful analysis can yield better bounds.

2.6 Discussion

Figure 1 plots the running times of Scott and Sorkin's MAX-2-CSP algorithm, our MAX-2-CSP algorithm and the MAX-2-SAT algorithm from the previous section. Note that the MAX-2-SAT running time is obtained by combining the bounds from BRF and GS. For formulas with average degree at most 6 the bounds yielded by the BRF algorithm are better, while for higher average degree GS yields the best bound. We note that the best known deterministic poly-space algorithm for MAX-2-CSP w.r.t. the number of constraints is $O^*(2^{m/5.263})$ (from [25, 10]) essentially works in the same way as Max2CSPAlg: branch on a vertex of maximum degree and eliminate vertices of degree at most 2. This is the reason why we cannot obtain a better bound for MAX-2-CSP using similar techniques as in Max2SATAlgParam. Therefore, an improved upper bound for either MAX-2-CSP or MAX-2-SAT w.r.t. the number of constraints will also improve the above bounds.

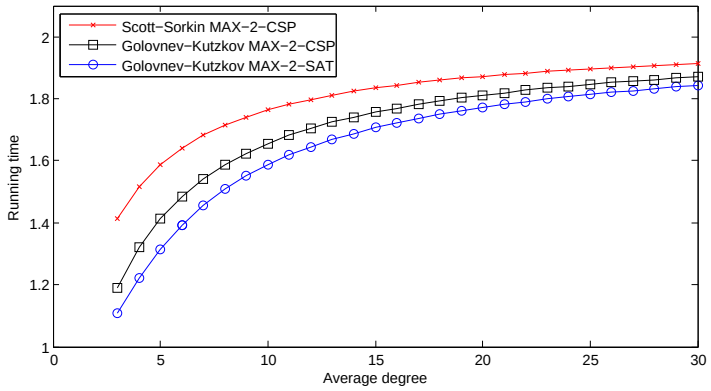

Figure 1: Running times of the Scott-Sorkin Max-2-CSP algorithm and the two new algorithms.

3 An algorithm based on graph separators

In this section we show how to obtain an exponential time algorithm for MAX-2-CSP such that the exponent is growing asymptotically slower with the average degree than the algorithms from the previous sections.

Algorithm 5 Max-2-CSP-GraphSeparator.

Input: (G, S) – instance of MAX-2-CSP.
Output: $Opt(G, S)$.

1: Find a balanced graph separator $\mathcal{S}$ of maximum cardinality.
2: Recursively branch on each vertex $v \in \mathcal{S}$.
3: In each leaf of the splitting tree solve MAX-2-CSP for the connected components induced by $V \backslash \mathcal{S}$ by the brute-force splitting algorithm.

The main idea is based on the observation that for a graph of average degree $d \leq n/2$ one can find a

balanced graph separator $\mathcal{S}$ of cardinality nc_d for $c_d < 1$. Then a simple algorithm for the Max 2-CSP problem is straightforward: branch on all vertices in $\mathcal{S}$ and for each leaf in the resulting tree solve the problem for each connected component. (See Fig. **??**.)

Building on a result by Feige and Kogan [8] we present results on the existence of balanced graph separators and efficient algorithms for finding them. The following result is a straightforward adjustment of Lemma 3.4 in [8] for our purposes and shows the existence of a balanced graph separator of cardinality depending on the average degree d.

Lemma 4 *Let $G = (V, E)$ be a simple connected graph over m edges and n vertices, and let G have average degree $d = o(n)$. Then for any constant $0 < \alpha < 1$ and sufficiently large d, there exists a balanced graph separator S of cardinality $n(1 - \frac{2\alpha \ln d}{d})$.*

Proof: Let $k = \frac{\alpha n \ln d}{d}$ for some constant $\alpha < 1$. We choose at random a set R of k vertices in G. We next show that the probability there exists a set $D \subset V, D \cap R = \emptyset$ with cardinality at least k such that there are no edges $(u, v) \in E : u \in R, v \in D$ is more than 0. By the probabilistic method this will prove the theorem.

For each $v \in V \backslash R$ the probability that v has no neighbors in R is $\frac{\binom{n-d_v}{k}}{\binom{n}{k}}$. Let X be a random variable counting the number of vertices in $V \backslash R$ with no neighbors in R. For each vertex $v \in V \backslash R$ we introduce an indicator random variable I_v such that $I_v = 1$ if v has no neighbors in R, and $I_v = 0$ otherwise. Thus, $X = \sum_{v \in V \backslash R} I_v$ and for the expectation of X we get

$$E[X] = \sum_{v \in V \backslash R} \frac{\binom{n-d_v}{k}}{\binom{n}{k}} \geq (n-k)\frac{\binom{n-d}{k}}{\binom{n}{k}} \geq \gamma n \prod_{i=0}^{k-1} \frac{n-d-i}{n-i}$$

for sufficiently large d and some $\gamma < 1$. The first inequality follows by convexity of a sum of binomial coefficients. Thus, we need to show that $\prod_{i=0}^{k-1} \frac{n-i}{n-d-i} \leq \frac{\gamma n}{k}$. We have

$$\prod_{i=0}^{k-1} \frac{n-i}{n-d-i} = \prod_{i=0}^{k-1} (1 + \frac{d}{n-d-i}) \leq (1 + \frac{d}{n-d-k})^k.$$

Since $d = o(n)$, one can choose sufficiently large d such that the last term can be bounded by $(1 + \frac{cd}{n})^k = O(\exp(\frac{ckd}{n}))$ for some $c > 1$ such that $\alpha c < 1$. For $k = \frac{n\alpha \ln d}{d}$ we obtain that $\prod_{i=0}^{k-1} \frac{n-i}{n-d-i} = O(d^{\alpha c})$. On the other hand we have $\frac{\gamma n}{k} = \frac{\gamma d}{\alpha \ln d} = \Omega(\frac{d}{\ln d})$. Thus, for a sufficiently large average degree d there exists a graph separator of cardinality $n(1 - \frac{2\alpha \ln d}{d})$. □

For sufficiently large average degree d, an asymptotically better bound on the complexity of MAX-2-CSP than the ones presented in the previous sections is now easy to show.

Theorem 4 *Given a 2-CSP formula over n variables with constraint graph of average degree $d = o(n)$,* Max-2-CSP-GraphSeparator *finds an optimal variable assignment in time $O^*(2^{\frac{(1+c_d)n}{2}})$ for $c_d = 1 - \frac{2\alpha \ln d}{d}$, $0 < \alpha < 1$, and polynomial space.*

Proof: A (k_1, k_2)-pairwise independent set in G can be found in time $O(1.561^n)$ by [20]. Clearly, the remaining vertices are balanced graph separator. By Lemma 4 there exists a balanced graph separator $\mathcal{S}$ of cardinality $n(1 - \frac{2\alpha \ln d}{d})$. Then after branching on all vertices S, in each leaf of the splitting tree the graph induced by $V \backslash \mathcal{S}$ consists of at least two pairwise disjoint connected components with at most $\frac{n - nc_d}{2}$ vertices. Thus we can separately solve MAX-2-CSP for each of them and combine the solutions in order to obtain an optimal variable assignment. We assume we apply the brute-force algorithm with running time $O^*(2^{\frac{n-nc_d}{2}})$, and thus we obtain a running time of $O^*(2^{nc_d} \cdot 2^{\frac{n-nc_d}{2}+1}) = O^*(2^{\frac{(1+c_d)n}{2}})$. □

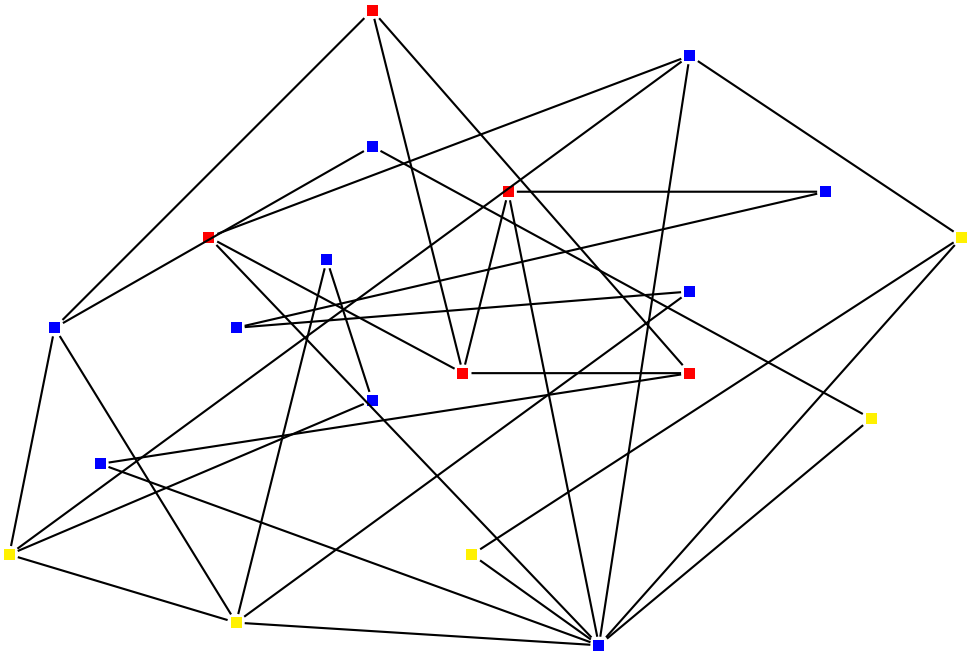

Figure 2: An example of the graph separator construction. The red and yellow vertices form a (5,5)-pairwise independent set. Thus, the blue vertices are balanced graph separator of cardinality 10. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4 Conclusions and future directions

The second algorithm seems to pose some interesting questions. Graph separators are widely used in the design of efficient algorithms for planar graphs since by the planar separator theorem [22] there exist graph separators of size $O(\sqrt{n})$. The bounds from Lemma 4 cannot yield asymptotically faster polynomial time algorithms. On the other hand, for exponential time algorithms the existence of small graph separators of linear size can result in asymptotically better upper bounds. To the best of our knowledge, the upper bound on the size of a balanced graph separator shown in Lemma 4 is the first of this kind. It is natural to ask whether better bounds in terms of the average graph degree are possible. Also, it is interesting whether exponential time algorithms for other hard problems can be designed using similar techniques.

Acknowledgements. We would like to thank Huxley Bennett for his help in writing the paper and Alexander S. Kulikov for valuable comments and suggestions. Also we would like to thank the anonymous referees for very detailed comments, which helped to improve the paper.

References

[1] N. Alon, J. Kahn, P. D. Seymour. Large induced degenerate subgraphs. *Graphs and Combinatorics 3(1)*: 203–211 (1987)

[2] D. Binkele-Raible, H. Fernau. A new upper bound for Max-2-SAT: A graph-theoretic approach. *J. Discrete Algorithms 8(4)*: 388–401 (2010)

[3] A. Björklund, T. Husfeldt, M. Koivisto. Set Partitioning via Inclusion-Exclusion. *SIAM J. Comput.* 39(2): 546–563 (2009)

[4] C. Calabro, R. Impagliazzo, R. Paturi. The Complexity of Satisfiability of Small Depth Circuits. *IWPEC 2009*: 75–85

[5] J. Chen, I. A. Kanj. Improved exact algorithms for MAX-SAT. *Discrete Applied Mathematics* 142(1-3): 17–27 (2004)

[6] F. D. Croce, M. J. Kaminski, V. Th. Paschos. An exact algorithm for MAX-CUT in sparse graphs. *Oper. Res. Lett.* 35(3): 403–408 (2007)

[7] E. Dantsin, A. Wolpert. MAX-SAT for Formulas with Constant Clause Density Can Be Solved Faster Than in $O(2^n)$ Time. *SAT 2006*: 266–276

[8] U. Feige, S. Kogan. Balanced coloring of bipartite graphs. *Journal of Graph Theory 64(4)*: 277–291 (2010)

[9] M. Fürer, S. P. Kasiviswanathan. Exact Max 2-Sat: Easier and Faster. *SOFSEM* (1) 2007: 272–283

[10] S. Gaspers, G. B. Sorkin. A universally fastest algorithm for Max 2-Sat, Max 2-CSP, and everything in between. *J. Comput. Syst. Sci.* 78(1): 305–335 (2012)

[11] A. Golovnev. New Upper Bounds for MAX-2-SAT and MAX-2-CSP w.r.t. the Average Variable Degree. *IPEC 2011*: 106–117

[12] J. Gramm, E. A. Hirsch, R. Niedermeier, P. Rossmanith. Worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT. *Discrete Applied Mathematics* 130(2): 139–155 (2003)

[13] J. Håstad. Some optimal inapproximability results. *J. ACM* 48(4): 798-859 (2001)

[14] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. *Journal of SIAM* 10 (1962), 196–210

[15] E. A. Hirsch. A New Algorithm for MAX-2-SAT. *STACS 2000*: 65–73

[16] E. Horowitz and S. Sahni. Computing partitions with applications to the knapsack problem. *Journal of the ACM* 21 (1974), 277–292

[17] R. Impagliazzo, R. Paturi, F. Zane. Which Problems Have Strongly Exponential Complexity? *J. Comput. Syst. Sci.* 63(4): 512–530 (2001)

[18] A. Kojevnikov, A. S. Kulikov. A new approach to proving upper bounds for MAX-2-SAT. *SODA 2006*: 11–17

[19] A. S. Kulikov, K. Kutzkov. New upper bounds for the problem of maximal satisfiability. *Discrete Mathematics and Applications*, 155–172, 2009

[20] K. Kutzkov. An exact exponential time algorithm for counting bipartite cliques. *Inf. Process. Lett. 112(13)*: 535–539 (2012)

[21] E. L. Lawler. A note on the complexity of the chromatic number problem. *Inf. Process. Lett. 5* (1976), 66–67.

[22] R. J. Lipton, R. E. Tarjan. Applications of a Planar Separator Theorem. *SIAM J. Comput.* 9(3): 615–627 (1980)

[23] R. A. Moser, D. Scheder. A full derandomization of Schöning's k-SAT algorithm. *STOC 2011*: 245–252

[24] U. Schöning. A probabilistic algorithm for k-SAT based on limited local search and restart. *Algorithmica* 32(4) (2002) 615–623

[25] A. D. Scott, G. B. Sorkin. Linear-programming design and analysis of fast algorithms for Max 2-CSP. *Discrete Optimization* 4(3–4): 260–287 (2007)

[26] R. E. Tarjan and A. E. Trojanowski. Finding a maximum independent set. *SIAM Journal on Computing 6* (1977), 537–546

[27] V. Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. *STOC 2012*: 887–898

[28] R. Williams. A new algorithm for optimal 2-constraint satisfaction and its implications. *Theor. Comput. Sci.* 348(2-3): 357–365 (2005)