Algorithms Based on the Treewidth of Sparse Graphs

Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith
Email: {kneis,moelle,richter,rossmani}@cs.rwth-aachen.de

Computer Science Department, RWTH Aachen University, Fed. Rep. of Germany

Abstract. We prove that given a graph, one can efficiently find a set of no more than $m/5.217 + 1$ nodes whose removal yields a partial two-tree. As an application, we immediately get simple algorithms for several problems, including Max-Cut, Max-2-SAT and Max-2-XSAT. All of these take a record-breaking time of $O^*(2^{m/5.217})$, where m is the number of clauses or edges, while only using polynomial space. Moreover, the existence of the aforementioned node sets implies an upper bound of $m/5.217 + 3$ on the treewidth of a graph with m edges. Letting go of polynomial space restrictions, this can be improved to a bound of $m/5.769 + O(\log n)$ on the pathwidth, leading to algorithms for the above problems that take $O^*(2^{m/5.769})$ time.

1 Introduction

Recently, there has been a wave of effort in proving exponential-time worst-case upper bounds for NP-hard problems—in particular for the exact solution of Max-SNP-hard problems [11]. One of the most intensely investigated problems in this area seems to be SAT, the problem of satisfiability of a propositional formula in conjunctive normal form (CNF). The maximum satisfiability problem—Max-SAT—is an important generalization of SAT. Given a formula in CNF, it asks for the maximum number of simultaneously satisfiable clauses. The decision variant of this problem is complete for both NP and Max-SNP, even if each clause contains at most two literals—this restriction is called Max-2-SAT. Recently, numerous results regarding worst-case time bounds for the exact solution of Max-SAT and Max-2-SAT have been published. The best bound that has been achieved for Max-SAT [3] with regards to m, the number of clauses, is $O^*(2^{m/2.46})$.[1] The best algorithm developed particularly for Max-2-SAT [6] has time complexity $O^*(2^{m/5})$. With respect to the number of variables, the trivial $O^*(2^n)$ algorithm has not been improved until recently, when Williams came up with a new algorithm solving Max-2-SAT in $O^*(2^{\omega n/3})$ steps [10] for $\omega \approx 2.379$.

In general, a Max-SAT instance is represented by a multiset rather than a set of clauses, since a clause may occur more than once. In order to account for this, we let m denote the number of clause occurrences—the total *weight*.

[1] The O^*-notation was introduced by Woeginger and suppresses all polynomial factors; e.g., $2^k n^5 = O^*(2^k)$.

Furthermore, we declare t to stand for the number of *clause types*. A *clause type* will be understood as a maximum distinct set of variables occuring together in at least one clause, disregarding negations. It is easy to see that t can be much smaller than m, even in formulæ that do not have multiple identical clauses.

In this paper, we present a very simple algorithm for MAX-2-SAT that has time complexity $O^*(2^{t/5})$, and thus $O^*(2^{m/5})$ just like the involved one by Gramm et al. [6]. Moreover, we analyze a slightly more complicated version of the algorithm, lowering the bound to $O^*(2^{t/5.217})$. The latter improves upon the best known upper bounds for solving MAX-2-SAT [6], MAX-2-XSAT [7], and MAX-CUT [8]—while still using only polynomial space. Furtherwise, we present the fastest polynomial space algorithm for DOMINATING SET on cubic graphs, which takes $O^*(2^{n/2})$ time. Whereas the runtime bound for this problem has recently been improved from $O^*(1.51^n)$ [5] to $O^*(1.21^n)$ [4], the latter algorithm is very involved and requires exponential space.

Impressive as these new record bounds may seem, they are just the tip of the iceberg. In fact, they represent little more than mere by-products of a much more general technique. It relies on our main graph theoretical result, which states that the removal of at most $|E|/5.217+1$ nodes from a graph $G=(V,E)$ yields a series-parallel graph. The method that stems from this observation enables a narrowing of the search space for many important NP-hard problems. In particular, a simple application yields the above-mentioned record-breaking bounds.

Moreover, it follows that the treewidth and pathwidth of G are at most $|E|/5.217+3$ and $|E|/5.217+3+\log n$, respectively. Employing a recent result by Fomin and Høie, we can improve these bounds to $|E|/5.769+O(\log n)$, yielding $O^*(2^{m/5.769})$ time algorithms for MAX-2-SAT, MAX-2-XSAT, and MAX-CUT based on dynamic programming [9]—at the price of exponential space.

2 Preliminaries

We assume the reader to be familiar with the notion of treewidth [1]. Throughout this paper, we adhere to the notation for boolean formulæ used by Gramm et al. [6].

Let F be a formula in 2-CNF whose set of variables will be called V. The corresponding *connectivity graph* is $G_F=(V,E)$ where

$$E=\big\{\,\{x,y\}\mid \text{the distinct variables } x \text{ and } y \text{ occur together in a clause}\,\big\},$$

representing the way variables interact in a formula. Notice that it does not make a difference in how many clauses a pair of variables occurs, or whether a variable is negated or not. For instance, the two graphs $G_{F[x]}$ and $G_{F[\bar{x}]}$ are identical. As a consequence, the formula F cannot be reconstructed from G_F.

3 An algorithm with only one reduction rule

In what follows, we prove our foundational result: a graph with m edges has treewidth at most $m/5+2$, and we can quickly find a set of no more than $m/5$

nodes whose removal leaves a very simply structured graph, namely a special case of a partial 2-tree. As an application of this technique we can solve several optimization problems efficiently.

More precisely, our method becomes applicable when problems can be expressed in graph terms as follows: There is a graph $G = (V, E)$ for every instance, and given a node v in the graph, we can reduce the instance to a smaller one whose graph is $G[V \setminus \{v\}]$. Moreover, the problem must be easy to solve when the corresponding graphs are partial 2-trees. Finally, reduction steps on nodes of degree two or more may be *expensive*, whereas nodes of degree one have to be *easy* to deal with in the problem context. Then, the algorithm derived from our graph-theoretical result takes at most $m/5$ expensive operations to reduce any input instance with m egdes to one whose graph is a special case of a partial 2-tree.

Many problems have the aforementioned properties, where the expensive operations usually originate from case distinctions that lead to branching in the recursion tree. Consider MAX-CUT as an example: Vertices of degree one can be deleted, since they will increase the overall size of a maximum cut by one in any case, whereas nodes of higher degree require branching.

The algorithm presented in this section is rather simple and broadly applicable. An even simpler algorithm will emerge in the next section; however, it will have the additional requirement that nodes of degree two are easy to deal with as well. Hence, if this condition does not hold for a problem, we have to stick to the more general algorithm from this section; otherwise, the simpler algorithm from the next section is preferable.

The special rôle of degree-one nodes in the first algorithm is reflected in the following definition:

Definition 1. Let $G = (V, E)$ be a graph. Then $R(G)$ is the graph obtained by deleting vertices v with $\deg(v) = 1$ repeatedly until there are no such vertices left.

Observe that $R(G)$ is well-defined, since it does not make any difference in what order nodes are chosen for deletion. What is more, the following lemma shows that even when we delete arbitrary nodes between reductions, the order is irrelevant. This property greatly simplifies algorithmic application of the rule. From now on, we shorten $G[V \setminus \{v\}]$ to $G - v$ as well as $G[V \setminus D]$ to $G - D$.

Lemma 1. *Let $G = (V, E)$ be a graph and $D = \{v_1, \ldots, v_k\}$ a set of vertices from V. Then, $R(G - D) = R(R(\ldots(R(R(G - v_1) - v_2) - v_3) \cdots - v_{k-1}) - v_k)$.*

The previous as well as the following lemma are straightforward. In all interesting cases, R-reducing a graph does not affect its treewidth:

Lemma 2. *Let G be a graph containing a cycle. Then $tw(G) = tw(R(G))$.*

Having investigated the properties of our only reduction rule, we turn our attention to the simple family of hot dog graphs. Surprisingly, any graph can be turned into a hot dog graph by deleting a small set of nodes and applying the R-reduction.

Definition 2. A path of length at least one between two possibly identical nodes s and t in a graph G is called a *leg* if all its nodes other than s and t have degree two in G. A *hot dog graph* consists of nodes $v_1, \ldots, v_k$ such that v_i and v_{i+1} are connected by arbitrarily many legs. Additionally, v_k and v_1 may be connected in this fashion as well.

Definition 3. Let $G = (V, E)$ be a graph whose nodes have degree at least two. A 4-*spider* is a subgraph that consists of a *head* $h \in V$ with degree four, three or four distinct *feet* $u_1, \ldots, u_l \in V \setminus \{h\}$ of degree at least three, and four disjoint legs connecting head and feet.

A 3-*spider* is defined similarly for a head of degree three and exactly three distinct feet connected to it via three legs. In any case, the *body* of a spider consists of all its nodes except the feet.

The nice thing about spiders is that their bodies can be removed from a graph quite easily: First remove the head, which is a node with relatively high degree, and then remove the remainder of the body by consecutively removing nodes of degree one.

It is interesting to note that hot dog graphs cannot contain spiders. The following lemma shows that the converse is also true in a fairly general setting. This enables us to turn any graph into a hot dog graph using relatively cheap operations.

Lemma 3. *Let $G = (V, E)$ be a connected graph whose nodes have degree between two and four. G is a hot dog graph iff it does not contain a 3- or 4-spider.*

Proof. It is obvious that a hot dog graph cannot contain spiders. On the other hand, let G be a graph as postulated in the premise that does not contain a spider. Let H be the set of nodes that do not have exactly two neighbors. Observe that every $v \in H$ may be connected to at most two more nodes from H via legs, because otherwise v_i would be the head of a spider. Thus, we can arrange the nodes from H in a linear or cyclical fashion as in the definition of a hot dog graph. □

Interestingly, if a node v has been the head of a spider in G, it keeps this rôle in the contracted graph. In what follows, we want to estimate the spider bodycount required to carve out a hot dog graph. In effect, we need to look for the number of edges that have to be removed. As it turns out, this feat is substantially eased if we analyze in terms of a potential function of nodes instead.

Definition 4. Let $G = (V, E)$ be a graph, $v \in V$, and

$$\deg_3(v) = \bigl|\{u \in V \mid \text{ there is a leg connecting } u \text{ and } v, \text{ and } \deg(u) \geq 3\}\bigr|.$$

We define the potential functions $\psi\colon V \to \mathbf{N}$ and $\Psi\colon \mathcal{G} \to \mathbf{N}$ as follows:

$$\psi(v) = \begin{cases} 0 & \text{if } \deg(v) \leq 2 \\ 0 & \text{if } \deg(v) = 3 \text{ and } \deg_3(v) = 1 \\ 5/4 & \text{if } \deg(v) = 3 \text{ and } \deg_3(v) > 1 \\ 2 & \text{if } \deg(v) \geq 4 \end{cases}$$

We extend the definition to graphs via $\Psi(G) = \sum_{v \in V} \psi(v)$.

Lemma 4. *Let $G = (V, E)$ be a graph. Then $\Psi(G) \leq |E|$.*

Proof.

$$|E| = \frac{1}{2} \sum_{v \in V} \deg(v) = \frac{1}{2} \sum_{i=1}^{|V|-1} \sum_{\substack{v \in V \\ \deg(v)=i}} i \geq \sum_{\substack{v \in V \\ \deg(v)=3}} \frac{5}{4} + \sum_{\substack{v \in V \\ \deg(v)\geq 4}} 2 \geq \Psi(G). \ \square$$

Lemma 5. *Let $G = (V, E)$ be a graph whose nodes have degree between two and four. If G contains a 4-spider with head h, then $\Psi(R(G - h)) \leq \Psi(G) - 5$.*

Proof. Let S be a 4-spider with head h. We have to distinguish several cases.

In the first case, S has four different feet $u_1, \ldots, u_4$ with $3 \leq \deg(u_i) \leq 4$. Removing h and all nodes of degree one consecutively has the following effect: Because h is erased, the potential decreases by 2. As a consequence, the degree of each foot is lowered by one. This means that the potential decreases by $2 - 5/4 = 3/4$ or $5/4 - 0 = 5/4$ per foot. The total loss of potential thus amounts to at least $2 + 4(3/4) = 5$.

In the second case, there are only three feet $u_1, \ldots, u_3$, and the situation is slightly more complicated. W.l.o.g. two paths are leading to u_1 and one path each to u_2 and u_3. If $\deg(u_1) = 4$, removing the body of S does the following: The potential of u_1 is lowered by 2, the potential of h decreased by 2 as well, and the potentials of u_2 and u_3 shrink by $2 - 5/4 = 3/4$ or $5/4 - 0 = 5/4$ each. Altogether, these values sum up to a loss of potential greater than 5.

Otherwise, if $\deg(u_1) = 3$, only one other leg starts from u_1. Let z denote the node this leg ends in. Note that z and h have to be different, since otherwise S would not be a spider at all: There would be three paths to u_1, but only two feet.

If z, u_2, u_3 are all different, the potential of h decreases by 2, the potential of u_1 by $5/4$, and the potentials of z, u_2, and u_3 by at least $3/4$ each, which is again more than 5 in total. If z, u_2, u_3 are not all different, say $z = u_3 \neq u_2$, then the potential of h is lowered by 2, the potential of u_1 by $5/4$, the potential of $z = u_3$ by at least $5/4$, and the potential of u_2 by at least $3/4$, which is more than 5 altogether. $\square$

Lemma 6. *Let $G = (V, E)$ be a connected graph whose nodes have degree between two and four. If G does not contain any 4-spider, but a 3-spider with head h, then $\Psi(R(G - h)) \leq \Psi(G) - 5$.*

Proof. Let h be the head of a 3-spider with feet u_1, u_2, u_3. Removing this spider causes the potential to decrease by at least 5, since $\psi(h) = 5/4$, and we lose at least $5/4$ on each foot, too. To see this, distinguish the following two cases: Either, $\deg(u_i) = 3$—this leads to a decrease in potential of exactly $5/4$—or, $\deg(u_i) = 4$ for some i. In the latter case, observe that there is exactly one leg between u_i and h, as u_i is a foot of the 3-spider with head h. Since u_i cannot be the head of a 4-spider, the three other legs starting in u_i end in the same node z. Then, however, we have that $\psi(u_i) = 0$ in $R(G - h)$ due to the definition of ψ and $\deg_3$. $\square$

Let us now begin putting the pieces together.

Theorem 1. *Let $G = (V, E)$ be a graph. There is a set $D \subseteq V$ such that $R(G - D)$ is a hot dog graph and $|D| \leq |E|/5$.*

Proof. We construct a set of nodes D such that $R(G - D)$ is a hot dog graph. As long as G contains a node v with degree at least five, remove v from G and set $D := D \cup \{v\}$. Now delete the bodies of all 4-spiders from G, and then do the same for 3-spiders. Add the heads of all these spiders to D. Note that removing a spider's body is the same as removing its head and applying the reduction rule R afterwards.

We obtain a set D such that $R(G - D)$ is a hot dog graph. Using Lemmata 4, 5, and 6, it is easy to see that $|D| \leq m/5$. □

Theorem 2. *The treewidth of a graph $G = (V, E)$ is at most $|E|/5 + 2$.*

Proof. Let D be the set given by Theorem 1. By Lemma 2, R-reducing the graph $G - D$ leaves its treewidth intact, provided that $G - D$ contains a cycle. Hence, the treewidth of $G - D$ is at most that of a hot dog graph. It is easy to see that hot dog graphs constitute a special case of series-parallel graphs, which have treewidth at most two [2, p. 174]. Otherwise, $G - D$ is but a forest. Altogether, we have that $tw(G) \leq |D| + 2 = |E|/5 + 2$. □

4 A second rule simplifies the algorithm

In this section, we develop a simpler algorithm which employs a second reduction rule in addition, that is, a rule that replaces a path (u, v, w) with $\deg(v) = 2$ by the path (u, w). We call this operation *contracting* v. Notice that this introduces another constraint on the set of possible applications: Degree-two nodes must be easy to handle in the problem translation. That is, the way they contribute to a solution should only depend on their two neighbors.

In short, we trade simplicity for applicability: As we will see in what follows, the refined method allows for a much simpler implementation, and thus eases the analysis. Moreover, in the place of hot dog graphs, it leaves a trivial graph without any edges.

On the other hand, there are problems that do not meet the above extra constraint, while the technique from the previous section can still be employed. Again, consider Max-Cut: In the direct approach, it is not clear how to avoid branching on degree-two nodes. (Fortunately, Max-Cut can be reduced to Max-2-SAT by replacing an edge $\{x, y\}$ by two clauses $\{x, y\}$ and $\{\bar{x}, \bar{y}\}$. The number of edges becomes the number of clause types.)

Definition 5. Let $G = (V, E)$ be a graph and $v \in V$. Let $R'(G)$ be the graph that we get from G by repeatedly removing degree one vertices and contracting degree two vertices until no such operation is possible. Whenever a contraction leads to a double edge, only a single edge is retained. We also define $R'_v(G) := R'(G - v)$.

Algorithm A
Input: A graph $G = (V, E)$
Output: $C \subseteq V$, $|D| \leq |E|/5$, such that $R'(G - D)$ has no edges

```
D ← ∅;
while there is a node v with deg(v) ≥ 3 do
    choose a node v with maximum degree;
    D ← D ∪ {v}; G ← R'_v(G)
od;
return D
```

Fig. 1. A simpler algorithm that uses R' rather than R

Lemma 7. *Let $G = (V, E)$ be a graph with minimum degree three and maximum degree four. If $v \in V$ and $\deg(v) = 4$, then $\Psi(R'_v(G)) \leq \Psi(G) - 5$.*

Proof. Let $u_1, \ldots, u_4$ be the neighbors of v. We have $\psi(v) = 2$, and removing v decreases the degree of each u_i by one. In total, the operation lowers the potential by at least $2 + 4(3/4) = 5$. Since neither the removal of a degree one node nor the contraction of a degree two node can increase the potential, this implies $\Psi(R'_v(G)) \leq \Psi(G) - 5$. □

Lemma 8. *Let $G = (V, E)$ be a 3-regular graph. For every $v \in V$ we have that $\Psi(R'_v(G)) \leq \Psi(G) - 5$.*

Proof. Every node in a 3-regular graph has a potential of $\Psi(3) = 5/4$. Removing v hence lowers the potential by 5. □

Theorem 3. *Algorithm A finds a set $D \subseteq V$ such that $|D| \leq m/5$ and the reduced graph $R'(G - D)$ has no edges.*

Proof. As long as there are nodes of degree at least five, the body of the **while**-loop increases the size of D by one while removing at least five edges. As soon as all nodes have degree at most four, $\Psi(G) \leq |E|$ by Lemma 4. From then on, the potential $\Psi(G)$ decreases by at least five in the body of the **while**-loop according to Lemmata 7 and 8. Since $R'(G)$ never contains nodes of degree one or two, the graph cannot have any edges when the algorithm terminates.

It is, however, not obvious that $R'(G - D)$ is the same graph. We only know that removing the nodes of D in the right order and applying reduction rules in between yields a graph without edges. However, analogously to Lemma 1, it is easy to see that indeed $R'_{x_k}(R'_{x_{k-1}}(\cdots R'_{x_1}(G) \cdots)) = R'(G - \{x_1, \ldots, x_k\})$. □

In order to use Algorithm A for solving Max-2-SAT, we must find reduction rules for formulæ that correspond to removing a node of degree one and contracting a node of degree two. A simple reduction rule can be used on a formula F to remove a node of degree one from G_F. But what do we need to do with F in order to contract a node of degree two in G_F? It is easy to see that we have to eliminate a variable x that occurs with *exactly two* other variables y and z in 2-clauses, introducing new clauses of the type $\{y, z\}$ in return.

Definition 6. Let F be a 2-SAT-formula. A variable x is a *double companion* if and only if the degree of x in G_F is two.

To ease the introduction of a double companion reduction rule, we now generalize the notion of a clause. We are used to defining a clause to be a pair (ω, C) where C is a set of (non-complementary) literals and ω a positive integer. In this section, we allow ω to be a negative integer as well. When we add up formulæ, the weights of identical clauses add up. Let F be a formula. Then $F[A]$ denotes the formula we get by setting all literals in A to true. This includes simplification and typically yields some satisfied clauses that take the form $(\omega, \mathbf{T})$. Moreover, let $\mathit{OptVal}(F)$ be the maximum (weighted) number of satisfiable clauses in F under any assignment [6]. For the following lemma, remember the definition of our new parameter t, the number of clause types.

Lemma 9 (The double companion reduction rule). *Let F be an arbitrary 2-SAT formula. If x is a double companion, then we can transform F into an equivalent formula F' which contains the same variables as F except x, and possibly clauses of negative weight, in polynomial time. The formula F' does not have more clause types than F. Moreover, $G_{F'}$ is the graph obtained from G_F by contracting x.*

Proof. Let x be a double companion that occurs together with y and z. Let $F = F' + F''$, where F' consists of all the clauses that contain x and F'' holds all the other clauses. We define $a = \mathit{OptVal}(F'[y, z])$, $b = \mathit{OptVal}(F'[y, \bar{z}])$, $c = \mathit{OptVal}(F'[\bar{y}, z])$, and $d = \mathit{OptVal}(F'[\bar{y}, \bar{z}])$. Let

$$G = \big\{(a + b + c + d, \mathbf{T}), (-d, \{y, z\}), (-c, \{y, \bar{z}\}), (-b, \{\bar{y}, z\}), (-a, \{\bar{y}, \bar{z}\})\big\}.$$

We easily see $a = \mathit{OptVal}(G[y, z])$, $b = \mathit{OptVal}(G[y, \bar{z}])$, $c = \mathit{OptVal}(G[\bar{y}, z])$, and $d = \mathit{OptVal}(G[\bar{y}, \bar{z}])$. Therefore, $\mathit{OptVal}(F' + F'') = \mathit{OptVal}(G + F'')$. Moreover, x does obviously not occur in $G + F''$. □

We now have reduction rules for formulæ in 2-CNF that enable us to eliminate all nodes with degree up to two in the corresponding connectivity graph. A naïve approach uses this machinery on the connectivity graph of a 2-CNF formula to find the number of satisfiable clauses. The algorithm can be easily modified to return an optimal assignment, too. The running time is again $O^*(2^{t/5})$, where $t \leq m$ is the number of different clause types.

It turns out that we need not use the connectivity graph explicitly. Instead, we can employ a recursive procedure as described in Algorithm B. In this form it corresponds to classical satisfiability algorithms starting with the Davis–Putnam procedure: Apply reduction rules as long as possible and then choose a variable for branching. In the past, better and better algorithms included more and more complicated rules. This involves reduction rules as well as rules for choosing a variable (or a group of variables) to branch on, combined with clever pruning of cases that cannot lead to an optimal assignment. In contrast, Algorithm B is very simple: It is comprised of only two reduction rules and one rule to choose a variable for branching, none of which are complicated.

Algorithm B
Input: A MAX-2-SAT-formula F
Output: $OptVal(F)$
Reduce F by the (double) companion reduction rule while possible;
if $F = \{(k, \mathbf{T})\}$ **then return** k
else
  choose a variable x that occurs in a maximum number of clause types;
  return $\max\{\text{Algorithm B}(F[x]), \text{Algorithm B}(F[\bar{x}])\}$
fi

Fig. 2. A very simple algorithm for MAX-2-SAT that does not use the connectivity graph directly.

5 Improving beyond $t/5$

In this section, we apply a tiny modification to the algorithm discussed above. More precisely, we introduce the additional rule to avoid picking a node of degree four all of whose neighbors have degree four as well, whenever possible.

We begin by looking at a special case for graphs of low degree. This theorem is of independent interest, and its proof serves to introduce the methods we apply in Theorem 5.

Theorem 4. *Let $G = (V, E)$ be a graph with m edges and maximum degree four. Then there is a set $D \subseteq V$, $|D| \leq \frac{3}{16}m + 1$, such that $R'(G - D)$ has no edges.*

Proof. Given $G = (V, E)$, construct $D \subseteq V$ as follows. Pick a vertex of maximum degree, and while choosing vertices of degree four, only take a vertex all of whose neighbors have degree four if no other type of degree-four node remains. Note that the latter is only the case if the graph is 4-regular. Remove the chosen vertex, apply the two reduction rules, and repeat the procedure until the maximum degree in the remaining graph drops below three.

$$\psi(v) = \begin{cases} 0 & \text{if } \deg(v) \leq 2 \\ 4/3 & \text{if } \deg(v) = 3 \\ 2 & \text{if } \deg(v) \geq 4 \end{cases}$$

We redefine the potential function ψ as seen on the left. Let $\langle n_1, \ldots, n_d \rangle$ denote the case that we pick a node v of degree d whose neighbors have degree n_1 through n_d. The respective losses of potential caused by the removal of such nodes v can be computed easily: the potential of v drops to zero, whereas the degree of each of its neighbors decreases by one. For instance, the loss of potential in the case $\langle 4, 4, 4, 3 \rangle$ amounts to $2 + 3 \cdot (2 - 4/3) + 4/3$. The resulting values are listed in the following table.

case	$\langle 4,4,4,4 \rangle$	$\langle 4,4,4,3 \rangle$	$\langle 4,4,3,3 \rangle$	$\langle 4,3,3,3 \rangle$	$\langle 3,3,3,3 \rangle$	$\langle 3,3,3 \rangle$
loss	$4\frac{2}{3}$	$5\frac{1}{3}$	6	$6\frac{2}{3}$	$7\frac{1}{3}$	$5\frac{1}{3}$

Observe that the special case $\langle 4, 4, 4, 4 \rangle$ can only occur in the first iteration, which causes at most one extra step, or if preceded by $\langle 3, 3, 3, 3 \rangle$: Clearly, it cannot be preceded by $\langle 3,3,3\rangle$, because we always pick a vertex of maximum degree. Furthermore, a node of degree three is created in all the remaining cases, preventing the graph from becoming 4-regular and thus excluding the case $\langle 4,4,4,4\rangle$.

Except for the first step, the good case $\langle 3,3,3,3\rangle$ countervails against the bad case $\langle 4,4,4,4\rangle$. Since the average loss of potential in these two cases amounts to 6, we have that the potential decreases by an average of at least $5\frac{1}{3}$ per step. Hence, the overall potential will drop to zero after at most $\frac{3}{16}m$ additional iterations. □

Note that, analogously to Lemma 4, it is easily checked that $\Psi(G) \leq |E|$ for continuations of the potential functions in both the previous and the upcoming proof.

Theorem 5. *Let $G = (V, E)$ be a graph with m edges. Then there is a set $D \subseteq V$, $|D| \leq \frac{23}{120}m + 1$, such that $R'(G - D)$ has no edges.*

Proof. We use both the algorithm and the notation described in the proof to the previous theorem. Again, we redefine the potential function ψ:

$$\psi(v) = \begin{cases} 0 & \text{if } \deg(v) \leq 2 \\ 30/23 & \text{if } \deg(v) = 3 \\ 45/23 & \text{if } \deg(v) = 4 \\ 5/2 & \text{if } \deg(v) \geq 5 \end{cases}$$

Obviously, we get rid of at least six edges per iteration as long as the algorithm removes nodes of degree at least six. It hence suffices to switch to an analysis via potential as soon as the maximum degree in the remaining graph has decreased to at most five. When a node of degree five is deleted, this lowers the potential by at least $5/2 + 5 \cdot (5/2 - 45/23) = 5\frac{5}{23}$. The other cases are listed below.

case	$\langle 4,4,4,4\rangle$	$\langle 4,4,4,3\rangle$	$\langle 4,4,3,3\rangle$	$\langle 4,3,3,3\rangle$	$\langle 3,3,3,3\rangle$	$\langle 3,3,3\rangle$
loss	$4\frac{13}{23}$	$5\frac{5}{23}$	$5\frac{20}{23}$	$6\frac{12}{23}$	$7\frac{4}{23}$	$5\frac{5}{23}$

As detailed above, the good case $\langle 3,3,3,3\rangle$ countervails against the bad case $\langle 4,4,4,4\rangle$; their average loss of potential is $5\frac{20}{23}$. Hence, only nodes of degree at most two remain after at most $\frac{23}{120}m + 1$ iterations. □

Modifiying Algorithm A according to the above result leads to the following improved running times.

Corollary 1. Max-2-SAT *and* Max-2-XSAT *can be solved in $O^*(2^{t/5.217})$ and thus in $O^*(2^{m/5.217})$ time.* Max-Cut *can be solved in $O^*(2^{m/5.217})$ time. All algorithms require only polynomial space.*

In order to give an upper bound on the treewidth of a graph $G = (V, E)$ using the above results, it suffices to check that $tw(G - D) \leq 2$. This is because $tw(R'(G - D)) = 0$, and R' does not trivialize graphs of treewidth at least three [2, p. 174].

Corollary 2. *The treewidth of a graph $G = (V, E)$ is at most $|E|/5.217 + 3$.*

6 Dominating Set on graphs of maximum degree three

In the case of maximum degree three, Theorem 1 can be improved by redefining the potential function ψ as $\psi(v) = 1.5$ for degree three vertices v and $\psi(v) = 0$ otherwise. The refined theorem yields a set D of size at most $|E|/6$.

Consider a node v in such a graph. There are at most four ways to dominate v: by itself or not by itself, but by one of its neighbors. We try these four possibilities for each node from D, which accounts for at most $4^{m/6} = 2^{n/2}$ possibilities altogether. In each case, we mark the selected dominators as *dominating* and their neighbors as *dominated*. Then we remove the nodes in D from the graph, remembering the dominators in D as D'. An optimal dominating set D'' for the remaining annotated graph can be easily found in polynomial time. In any case, $D' \cup D''$ is a dominating set for the entire graph, and we obviously find a globally optimal dominating set by going through all the $2^{n/2}$ cases.

The overall algorithm is quite simple and has a running time of $O^*(2^{n/2}) = O^*(1.42^n)$, improving the result of $O^*(1.51^n)$ by Fomin, Kratsch, and Woeginger [5]. It is the fastest algorithm that uses polynomial space.

7 Trading space for time

The result on the size of D in the preceding section implicitly establishes an upper bound of $m/6+O(1)$ on the treewidth of cubic graphs. In a more general setting, such an upper bound might be interesting in its own right. According to Fomin and Høie [4], the pathwidth of a cubic graph is at most $(1+\epsilon)n/6 + O(\log n)$, where $\epsilon > 0$ is an arbitrarily small constant.

We can exploit this result in order to give an even tighter bound on the pathwidth of general graphs. Unfortunately, this method comes at the cost of losing several benefits of the approaches discussed so far. In particular, the resulting algorithms are no longer intuitive, and require exponential instead of polynomial space.

Theorem 6. *The pathwidth of a graph with m edges and n nodes is at most $m/5.769 + O(\log n)$. A corresponding path decomposition can be found in polynomial time. Thus* Max-Cut*,* Max-2-SAT*, and* Max-2-XSAT *can be solved in $O^*(2^{m/5.769})$ steps.*

Proof. Consider the following algorithm, which employs the aforementioned reduction rules for nodes of degree at most two without further notice: While there are nodes with degree at least six, we delete them from the graph. When only nodes of degree at most five remain, we have $\Psi(G) \le m$ for the continuation of the potential function ψ, when we choose $\alpha = 25/26$, $\beta = 25/13$, and $\gamma = 5/2$ in

$$\psi(v) = \begin{cases} \alpha & \text{if } \deg(v) = 3 \\ \beta & \text{if } \deg(v) = 4 \\ \gamma & \text{if } \deg(v) = 5 \\ 0 & \text{if } \deg(v) < 3. \end{cases}$$

It is easy to verify that the potential decreases by at least 75/13 whenever a node of degree four or five is removed. The sole exception is the removal of a degree-five node all of whose neighbors have degree five as well, which leads to a loss of only 145/26. We can avoid this case unless the graph is five-regular. As in the proof of Theorem 4, this can only occur if the node deleted in the previous step was of a special kind: Here, this would be nodes of degree at least six, and degree-five nodes with only degree-three neighbors. In the latter case the potential decreases by 90/13, implying a loss of $325/52 > 75/13$ on average.

When the algorithm terminates, a cubic graph remains. If p denotes its potential, it consists of exactly p/α nodes. According to the result of Fomin and Høie [4], its pathwidth is at most $(1+\epsilon)p/6\alpha + O(\log n) = (1+\epsilon)13p/75 + O(\log n)$. For the original graph and ϵ small enough, this implies a bound of $(1+\epsilon)13m/75 + O(\log n) \leq m/5.769 + O(\log n)$.

Acknowledgements. We would like to thank Hans Bodlaender for valuable pointers on literature and several anonymous referees for helpful comments.

References

1. H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–21, 1993.
2. A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey.* SIAM monographs on discrete mathematics and applications. SIAM, 1999.
3. J. Chen and I. A. Kanj. Improved exact algorithms for Max-Sat. *Discrete Applied Mathematics*, 142(1–3):17–27, 2004.
4. F. V. Fomin and K. Høie. Pathwidth of cubic graphs and exact algorithms. Technical Report 298, Department of Informatics, University of Bergen, May 2005.
5. F. V. Fomin, D. Kratsch, and G. J. Woeginger. Exact (exponential) algorithms for the dominating set problem. In *Proc. of 30th WG*, number 3353 in LNCS, pages 245–256. Springer, 2004.
6. J. Gramm, E. A. Hirsch, R. Niedermeier, and P. Rossmanith. New worst-case upper bounds for MAX-2-SAT with application to MAX-CUT. *Discrete Applied Mathematics*, 130(2):139–155, 2003.
7. B. A. Madsen and P. Rossmanith. Maximum exact satisfiability: NP-completeness proofs and exact algorithms. Technical Report RS-04-19, BRICS, Oct. 2004.
8. A. Scott and G. B. Sorkin. Faster algorithms for Max-CUT and Max-CSP, with polynomial expected time for sparse instances. In *Proc. of 7th RANDOM*, number 2764 in LNCS, pages 382–395. Springer, 2003.
9. J. A. Telle and A. Proskurowski. Algorithms for vertex partitioning problems on partial k-trees. *SIAM Journal on Discrete Mathematics*, 10(4):529–550, 1997.
10. R. Williams. A new algorithm for optimal constraint satisfaction and its implications. In *Proc. of 31st ICALP*, number 3142 in LNCS, pages 1227–1237. Springer, 2004.
11. G. Woeginger. Exact algorithms for NP-hard problems: A survey. In M. Junger, G. Reinelt, and G. Rinaldi, editors, *Combinatorial Optimization—Eureka, You Shrink!*, number 2570 in LNCS, pages 185–207. Springer, 2003.