RWTH Aachen

Department of Computer Science
Technical Report

A New Satisfiability Algorithm With Applications To Max-Cut

Joachim Kneis and Peter Rossmanith

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2005-08

RWTH Aachen · Department of Computer Science · April 2005

A New Satisfiability Algorithm With Applications To Max-Cut

Joachim Kneis and Peter Rossmanith

Lehr- und Forschungsgebiet Theoretische Informatik
RWTH Aachen, Germany
Email: {kneis,rossmani}@cs.rwth-aachen.de

Abstract. We prove an upper bound of $m/5.217 + 3$ on the treewidth of a graph with m edges. Moreover, we can always find efficiently a set of no more than $m/5.217 + 1$ nodes whose removal yields a partial 2-tree. As an application, we immediately get simple algorithms for several problems, including Max-Cut, Max-2-SAT and Max-2-XSAT. The resulting algorithms have running times of $O^*(2^{t/5.217})$, where t is the number of distinct clause types. In particular, this implies a record-breaking time complexity of $O^*(2^{m/5.217})$.

1 Introduction

We are currently experiencing a renaissance in the investigation of exponential-time algorithms. The first subsection will elaborate on this. One example of obvious importance is testing a boolean formula for satisfiability. Following a short review of earlier scholarship in this area, the contributions of the paper at hand will be highlighted.

1.1 Worst-case upper bounds for NP-hard problems

We believe that no polynomial time algorithms exist for NP-hard problems. Still, there is no doubt as to the practical relevance of many of them. For some of these problems, there are polynomial-time approximation algorithms that give solutions within a factor α, usually called the performance ratio, of the optimal solution. However, for problems that are Max-SNP-hard [1, 29], it is known that the performance ratio of a polynomial-time algorithm cannot be better than some constant ζ, called the inapproximability ratio, unless P=NP. For example, the ratios known for Max-2-SAT are $\alpha = 0.931$ [10] and $\zeta = 0.955$ [15].

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. 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. In the early 1980s, the trivial bound of $O^*(2^n)$ has been improved for formulæ in 3-CNF[1], where every clause contains at most three literals, by Monien and Speckenmeyer [27]. After that, improved upper bounds for k-SAT [8, 17, 23, 24, 30, 32], Max-SAT [6, 3, 26, 28], Max-2-SAT [3, 28], and other NP-hard problems could be obtained.

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

1.2 Related work

Concerning the problems for formulæ in CNF, most authors consider bounds with respect to three parameters:

- the length l of the input formula (i.e., the number of literal occurrences),
- the number m of its clauses, and
- the number n of the variables occurring in it.

As of today, $O^*(2^{l/9.7})$ and $O^*(2^{m/3.23})$ are the best bounds for SAT [17]. In contrast, with respect to the number of variables, nothing better than the trivial bound of $O^*(2^n)$ is known. For the special case of 3-SAT, the bounds with respect to l and m are the same as for SAT, whereas a randomized algorithm [18] taking $O^*(1.324^n)$ steps and a deterministic one [8] running in $O^*(1.481^n)$ time are known.

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 [29]. Max-SAT and Max-2-SAT are well-studied in the context of approximation algorithms [2, 7, 10, 15, 19, 37]. Recently, numerous results regarding worst-case time bounds for the exact solution of Max-SAT and Max-2-SAT have been published [3, 7, 14, 16, 26, 28, 12]. The best bounds that have been achieved for Max-SAT [3] are $O^*(2^{l/6.89})$ and $O^*(2^{m/2.36})$. For Max-2-SAT, the considerably better bounds of $O^*(2^{m/2.46})$, $O^*(2^{m/2.88})$ and $O^*(2^{m/3.44})$ [6, 28, 3] follow from Max-SAT algorithms. The best algorithm developed particularly for Max-2-SAT [12] has time complexity $O^*(2^{m/5})$, which implies $O^*(2^{l/10})$. 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 [36] for an $\omega < 2.379$. More precisely, $O^*(n^\omega)$ is the asymptotic running time of the best algorithm for matrix product over a ring. For Max-SAT, there is another relevant parameter, namely the number k of satisfiable clauses. An algorithm that is efficient with respect to k might be faster than one that is good with respect to m if k is much smaller than m [26, 22].

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*. 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.

1.3 Contributions of this paper

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 one by Gramm et al. [12]. 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 [13], Max-2-XSAT [25] and Max-Cut [33].

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 treewidth of a graph $G = (V, E)$ is bounded by $|E|/5.217+3$. Furthermore, a tree decomposition of this size can be obtained in polynomial time. 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.

2 Preliminaries

In this section we describe the notation we use for MAX-CUT and satisfiability problems. Moreover, we recall the notion of treewidth.

2.1 Maximum satisfiability

Throughout this paper, we adhere to the notation for boolean formulæ used by Gramm et al. [12]. Let V be a set of boolean variables. A *literal* is either a variable or its negation. As usual, the negation of a variable x is denoted by $\bar{x}$, and whenever l denotes a negated variable $\bar{x}$, then $\bar{l}$ stands for the variable x.

Algorithms for finding the exact solution of MAX-SAT are often designed for the unweighted MAX-SAT problem. However, MAX-SAT formulæ are generally represented by multisets, i.e., formulæ in CNF with positive integer weights. Thus, we consider the weighted MAX-SAT problem with positive integer weights. A *(weighted) clause* is a pair (ω, S), where ω is a positive integer and S is a nonempty finite set of literals that does not contain any variable and its negation simultaneously. We call ω the *weight* of the clause.

An *assignment* is a finite set of literals that does not contain any variable together with its negation. Informally speaking, if an assignment A contains a literal l, then the literal l has the value *True* in A. In addition to usual clauses, we allow a special *true clause* $(\omega, \mathbf{T})$, also called a $\mathbf{T}$-*clause*, which is satisfied by every assignment.

The *length* of a clause (ω, S) is the cardinality of S, or 0 in the case of a $\mathbf{T}$-clause; a k-*clause* is a clause of length exactly k. In this paper, a *formula* — more precisely a *formula in (weighted) CNF* — is a finite set of weighted clauses (ω, S), with at most one clause for each S. A formula is in 2-*CNF* if it contains only 2-clauses, 1-clauses and possibly a $\mathbf{T}$-clause. The *length of a formula* is the sum of the lengths of all its clauses.

Pairs of the $(0, S)$ variety are *not* clauses; for simplicity, however, we assume $(0, S) \in F$ for all S and all F when defining the operators $+$ and $-$:

$$F + G = \big\{ (\omega_1 + \omega_2, S) \bigm| (\omega_1, S) \in F \text{ and } (\omega_2, S) \in G, \text{ and } \omega_1 + \omega_2 > 0 \big\},$$
$$F - G = \big\{ (\omega_1 - \omega_2, S) \bigm| (\omega_1, S) \in F \text{ and } (\omega_2, S) \in G, \text{ and } \omega_1 - \omega_2 > 0 \big\}.$$

Example 1. If

$$F = \{\ (2, \mathbf{T}),\ (3, \{x, y\}),\ (4, \{\bar{x}, \bar{y}\})\ \}$$

and

$$G = \{\ (2, \{x, y\}),\ (4, \{\bar{x}, \bar{y}\})\ \},$$

then

$$F - G = \{\ (2, \mathbf{T}),\ (1, \{x, y\})\ \}.$$

For a literal l and a formula F, the formula $F[l]$ is obtained by setting the value of l to *true*. More precisely, we define

$$\begin{aligned} F[l] = (\{\ (\omega, S) \mid (\omega, S) \in F \text{ and } l, \bar{l} \notin S\ \} +& \\ \{\ (\omega, S \setminus \{\,\bar{l}\,\}) \mid (\omega, S) \in F \text{ and } S \neq \{\,\bar{l}\,\} \text{ and } \bar{l} \in S\ \} +& \\ \{\ (\omega, \mathbf{T}) \mid \omega \text{ is the sum of the weights } \omega'& \\ \text{of all clauses } (\omega', S) \text{ of } F \text{ such that } l \in S\ \}.& \end{aligned}$$

Note that no $(\omega, \emptyset)$ or $(0, S)$ is included in $F[l]$, $F+G$ or $F-G$. For an assignment $A = \{l_1, \ldots, l_s\}$ and a formula F, we define $F[A] = F[l_1][l_2] \ldots [l_s]$. Evidently, $F[l][l'] = F[l'][l]$ for every pair of literals l, l' with $l \neq \bar{l}'$. In short, we write $F[l_1, \ldots, l_s]$ instead of $F[\{l_1, \ldots, l_s\}]$.

Example 2. If

$$F = \{\ (1, \mathbf{T}),\ (1, \{x, y\}),\ (5, \{\bar{y}\}),\ (2, \{\bar{x}, \bar{y}\}),\ (10, \{\bar{z}\}),\ (2, \{\bar{x}, z\})\ \},$$

then

$$F[x, \bar{z}] = \{\ (12, \mathbf{T}),\ (7, \{\bar{y}\})\ \}.$$

□

The *weight of satisfied clauses* for a formula F and an assignment A is defined as ω where $(\omega, \mathbf{T})$ is the $\mathbf{T}$-*clause* in $F[A]$, or 0 if there is none such. As expected, the *maximum weight of satisfied clauses* for a formula F is $\mathit{OptVal}(F) = \max_A \{\, \omega \mid (\omega, \mathbf{T}) \in F[A]\,\}$, where A is taken over all possible assignments. An assignment A is *optimal* iff $F[A]$ only contains $(\omega, \mathbf{T})$ and $\omega = \mathit{OptVal}(F)$. Note that when $\omega = 0$, the simplified formula $F[A]$ does not contain any clause. We say that two formulæ F_1 and F_2 are *equivalent* if there is no assignment A such that the weight of satisfied clauses for F_1 and A differs from the one for F_2 and A.

2.2 Maximum cut

Let $G = (V, E)$ be an undirected graph. If $S \,\dot\cup\, T$ is a partition of V, we call the pair (S, T) a *cut*. The *size* of a cut (S, T) is the number of edges connecting S and T. The Max-Cut problem is to find a cut of maximal size. Its complexity is well investigated in terms of the number of edges m [9, 12, 14]; the best algorithm so far [33] has time complexity $O^*(2^{m/5})$.

It is well known that Max-Cut can be solved by transforming an instance of Max-Cut into a Max-2-SAT instance as follows: The set of variables corresponds to the set of vertices. For every edge $\{x, y\}$ in the graph we add the two clauses $\{x, y\}$ and $\{\bar{x}, \bar{y}\}$ to the formula. It is easy to see that the graph has a cut of size k iff $m + k$ clauses can be satisfied in the corresponding formula, where m is the number of edges. In this way, an $O^*(2^{\alpha m})$ step algorithm for Max-2-SAT can be employed to solve Max-Cut in $O^*(2^{2\alpha m})$ steps.

Recently, the problem Max-2-XSAT has been investigated. It is defined similar to Max-2-SAT, but a clause is only considered fulfilled by an assignment A if A satisfies *exactly one* of its literals. There is an algorithm that solves

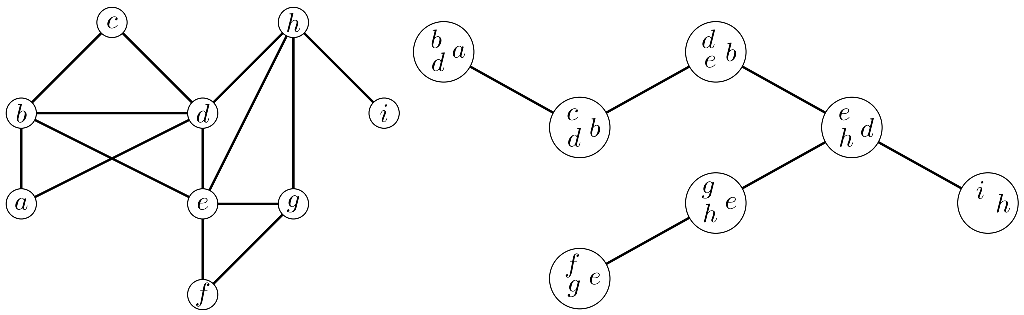

Fig. 1. A graph of treewidth four and an optimal tree decomposition.

Max-2-XSAT in $O^*(2^{m/4})$ steps [25]. The connection between Max-2-XSAT and Max-Cut is even tighter than the one between Max-2-SAT and Max-Cut: The Max-2-XSAT-formula containing the clause $\{x, y\}$ for each edge $\{x, y\}$ has m simultaneously satisfiable clauses iff there is a cut of size m in the given graph. It is thus only fair to say that Max-Cut is a special case of the more general problem Max-2-XSAT, where negative literals are allowed. Still, the algorithm for Max-2-XSAT [25] only yields an $O^*(2^{m/4})$ algorithm for Max-Cut, which is exactly the same time complexity achieved earlier by Fedin and Kulikov [9].

The results of this paper also imply a simpler algorithm for Max-2-XSAT whose running time is $O^*(2^{m/5.217})$. This implies a runtime bound of $O^*(2^{m/5.217})$ for Max-Cut. There is, however, another possibility to show the new bound for Max-Cut in terms of an algorithm for Max-2-SAT itself: We will show that the new Max-2-SAT-algorithm has a running time of only $O^*(2^{t/5.217})$, where t is the number of clause types. Using the above reduction, a graph with m edges is transformed into a Max-2-SAT-formula with $2m$ clauses, but only m *types* of clauses.

2.3 Treewidth

Treewidth measures how "treelike" a graph is. The notion of treewidth was introduced by Robertson and Seymour [31]. Bodlaender [4] and Kloks [20] give an introduction to this concept. Many graph problems that are hard in general can be solved efficiently, i.e., in polynomial and often linear time, for graphs of bounded treewidth. Well-known examples are Hamiltonian Path, Max-Cut, Independent Set and Vertex Cover [35]. Formally, we can define treewidth via tree decompositions:

A pair $\bigl(\{\, X_i \subseteq V \mid i \in I\}, T\bigr)$ is a *tree decomposition* of a graph $G = (V, E)$ if T is a tree with node set I, every edge $\{u, v\}$ is contained in some X_i (that is, $u, v \in X_i$), and $X_i \cap X_j \subseteq X_k$ for every k that lies on the path from i to j in T. The *width* of a tree decomposition $\bigl(\{\, X_i \mid i \in I\}, T\bigr)$ is $\max_{i \in I} |X_i| - 1$. The *treewidth* $tw(G)$ of G is the minimal width of all tree decompositions of G. Graphs of treewidth k are also called partial k-trees.

There is an immediate analogy to treewidth known as the *robber and cops game* [34]. In this game, a robber and some cops move between nodes of a graph according to simple rules. The robber may move along edges at any speed, whereas only one cop may slowly jump from his current location to any other node at a time. The game ends if the cops catch the robber. If and only if there exists a strategy for $k+1$ cops to catch a robber on a graph according to these simple rules, its treewidth is bounded by k.

We can derive all the main results in this paper without resorting to treewidth at all. It is, however, used as a main ingredient indirectly. Our algorithms can be seen as finding a tree decomposition of width $m/5+2$ for a graph with m edges. Unfortunately, there do not seem to exist many results that relate the number of edges of a graph to its treewidth. The only result known to us shows that *most* sparse graphs have large treewidth [21]. In particular, there are graphs with $m = \Theta(n)$ and $tw(G) = \Theta(m)$.

Is there a better upper bound on the treewidth than $m/5+2$? Note how improving the bound of $m/5+2$ (including a polynomial algorithm to find a tree or path decomposition) improves the running time of our first Max-2-SAT algorithm without further ado. On the other hand, we are looking for a family $\mathcal{G}$ of graphs, such that $tw(G) > \alpha m$ for all $G \in \mathcal{G}$ and an α as large as possible. Results in this vein should enable a clearer sight on the tightness of our upper bounds.

2.4 Formulæ and graphs

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.

Still, we can read a lot of information off G_F. An edge between x and y represents a direct dependency between x and y. On the other hand, x is isolated in G_F iff it only occurs in 1-clauses. Similarly, if x has degree one in G_F, it lends itself to simplification by what we will later call the companion rule.

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. These problems need to be expressable 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 these 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,*

$$R(G - D) = R(R(\ldots(R(R(G - v_1) - v_2) - v_3) \cdots - v_{k-1}) - v_k).$$

Proof. Let $V_1 = V \setminus D$ and V_2 the vertices from $R(\ldots(R(R(G - v_1) - v_2) - v_3) \cdots - v_{k-1}) - v_k$, that is, the sets of nodes on the left and right hand side of the equation before the final R-reduction is applied. Note that to show the claim, it suffices to look at the sets of nodes, because edges are only removed upon deletion of incident nodes. Moreover, the R-operation can only decrease degrees in the graph. That is why, before the final R-operation, $V_1 \supseteq V_2$. This implies one direction of the set equality.

We show the other inclusion by contradiction. Call v the first node that is removed on the right hand side but remains on the left. Clearly, $v \notin D$, and v has been deleted because of the reduction rule. As all its predecessors have been removed in $R(G - D)$, too, v will be deleted by the R-operation on the left hand side as well. □

In all interesting cases, R-reducing a graph does not affect its treewidth:

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

Proof. Let S be a strategy to catch a robber on $R(G)$. Note that only dead ends are removed by the reduction rule. A combination of different dead ends results in a tree. Therefore, the only difference between G and $R(G)$ are trees attached to nodes in G. If the robber is locked in a part of $R(G)$ by S, the attached trees in G do not change this. The only advantage the robber can gain from such trees is the possibility to hide in them, that is, in a tree attached to some node v guarded by a cop. Hence, in order to adapt S to G, we only need to catch the robber in such a tree if he really retreats to it. This is easily done by two cops. □

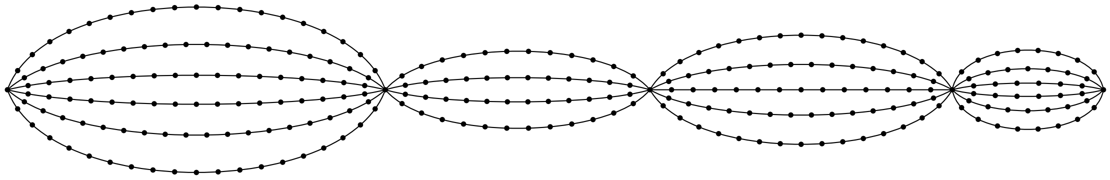

Fig. 2. A (non-cyclic) hot dog graph.

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. $\square$

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

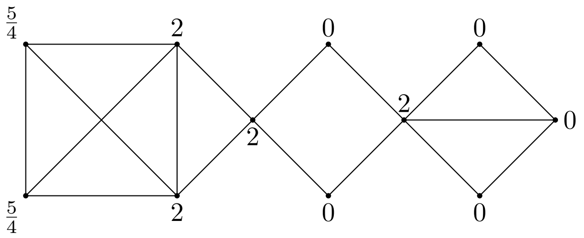

Fig. 3. A graph and the potential of its nodes: $\Psi(G) = 10.5$ and $|E| = 17$.

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*

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

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.

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

□

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 $2 + 4(3/4) = 5$.

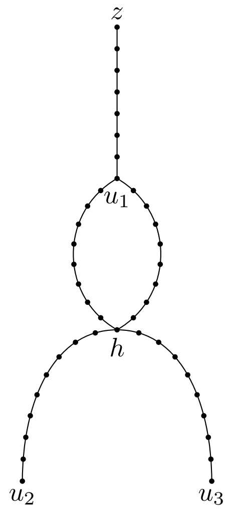

Fig. 4. A 4-spider with only three feet. Removing the spider and consequently erasing all nodes of degree one also decreases the potential of z by at least 3/4 if $\deg(u_1) = 3$.

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. See Figure 4 for an illustration.

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. □

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 $\deg_3$. □

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 the set given by Theorem 1. By Lemma 2, R-reducing $G - D$ leaves its treewidth intact, provided that it contains a cycle. Hence, the treewidth of $G - D$ is not higher than 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 [5, p. 174]. Otherwise, $G - D$ is but a forest. Altogether, we have that $tw(G) \leq |D| + 2 = |E|/5 + 2$. □

Having thus achieved our graph theoretic main result, we continue with an application to Max-2-SAT. The interpretation of the above result in the context of connectivity graphs immediately yields the following corollary:

Corollary 1. *Let F be a 2-SAT-formula with t clause types. Then we can find a set of variables $z_1, \ldots, z_r$, $r \leq t/5$ in polynomial time such that: If A is an assignment to $z_1, \ldots, z_r$, then the reduced connectivity graph $R(G_{F[A]})$ is a hot dog graph.*

Henceforth, when discussing connectivity graphs for formulæ, we do not distinguish between nodes and the variables they represent, that is we use the same names for both.

Lemma 7. *Let F be a 2-SAT-formula such that G_F is a hot dog graph. The maximum number of satisfiable clauses can be determined in polynomial time.*

Proof. Let $x_1, \ldots, x_k$ be the nodes of degree at least three, and let C_i denote the set of clauses containing a variable that lies on a path between two vertices in $\{x_1, \ldots, x_i\}$. Define c_i^0 as the maximum number of satisfied clauses in C_i when x_i is set to 0 (c_i^1 analogous). Clearly, $max(c_k^0, c_k^1)$ is the solution to the Max-2-SAT problem on F.

Both c_1^0 and c_1^1 are easy to calculate. We show how c_{i+1}^0 can be computed from c_i^0 and c_i^1 in polynomial time. Consider for instance the case where $x_i = x_{i+1} = 0$. For every leg between x_i and x_{i+1}, we compute the optimum assignment to the variables on this leg. This can be done in polynomial time using dynamic programming, because every such node has at most two neighbors. Adding up the values for every leg, we obtain the maximum number of satisfied clauses for $x_i = x_{i+1} = 0$. Repeating this procedure for $x_i = 1$ immediately yields c_{i+1}^0. □

Definition 5. *Let F be a 2-SAT-formula. We call the variable x a* companion *(*of *y) if there is a unique variable $y \neq x$ that occurs together with x in a clause.*

In terms of the respective connectivity graph G_F, the variable x is a companion if and only if the degree of x in G_F is one. Again, we may do away with such appendices in a fashion similar to R-reduction. Insofar, the next lemma is in analogy to Lemma 2.

Lemma 8 (The companion reduction rule). *Let F be a 2-SAT formula. If x is a companion, we can transform F into an equivalent formula F' containing the same variables except for x, where $G_{F'} = G_F - x$. This can be done in polynomial time.*

Proof. Let F be a formula, x a companion of y, F' consist of all clauses in F with an occurrence of the variable x, and $F'' = F - F'$. Let furthermore $a = \mathit{OptVal}(F'[y])$, $b = \mathit{OptVal}(F'[\bar{y}])$, and

$$H = \begin{cases} \{(b, \mathbf{T}), (a - b, \{y\})\} & \text{if } a > b \\ \{(a, \mathbf{T}), (b - a, \{\bar{y}\})\} & \text{otherwise} \end{cases}$$

It is easy to see that $a = \mathit{OptVal}(H[y])$ and $b = \mathit{OptVal}(H[\bar{y}])$. We immediately get

$$\begin{aligned} &\mathit{OptVal}(H + F'') = \\ &\quad \max\{\mathit{OptVal}(H[y]) + \mathit{OptVal}(F''[y]), \mathit{OptVal}(H[\bar{y}]) + \mathit{OptVal}(F''[\bar{y}])\} = \\ &\quad \max\{\mathit{OptVal}(F'[y] + \mathit{OptVal}(F''[y]), \mathit{OptVal}(F'[\bar{y}] + \mathit{OptVal}(F''[\bar{y}])\} = \\ &\qquad\qquad\qquad\qquad = \mathit{OptVal}(F' + F'') = \mathit{OptVal}(F). \end{aligned}$$

Hence, we can replace F by the equivalent formula $H + F''$. Note that it is very easy to calculate a and b, and that $H + F''$ does not contain the variable x anymore. □

Putting together Theorem 1 as well as Lemmata 7 and 8 analogously to Theorem 2 yields the following runtime bound:

Theorem 3. Max-2-SAT *can be solved in $O^*(2^{t/5})$ steps.*

4 A second rule simplifies the algorithm

In this section, we develop a simpler algorithm which employs a second reduction rule in addition, which 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, in this case, a different problem encoding will emerge that enables an application of the more straightforward second approach.

Algorithm A
Input: A graph $G = (V, E)$
Output: $D \subseteq V$, $|D| \leq |E|/5$, such that $R'(G - D)$ has no edges
$D \leftarrow \emptyset$;
while there is a node v with $\deg(v) \geq 3$ **do**
  choose a node v with maximum degree;
  $D \leftarrow D \cup \{v\}$; $G \leftarrow R'_v(G)$
od;
return D

Fig. 5. A simpler algorithm that uses R' rather than R

Definition 6. *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 $R'_v(G) := R'(G - v)$.*

Lemma 9. *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 10. *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 4. *Algorithm A finds a set $D \subseteq V$ such that $|D| \leq m/5$ and $R'(G - D)$ 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 9 and 10. 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. The companion 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 7. *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 defined 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. For the following theorem, remember the definition of our new parameter t, the number of clause types.

Lemma 11 (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 $a = OptVal(F'[y,z])$, $b = OptVal(F'[y,\bar{z}])$, $c = OptVal(F'[\bar{y},z])$, and $d = OptVal(F'[\bar{y},\bar{z}])$. 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 $a = OptVal(G[y,z])$, $b = OptVal(G[y,\bar{z}])$, $c = OptVal(G[\bar{y},z])$, and $d = OptVal(G[\bar{y},\bar{z}])$. Therefore, $OptVal(F' + F'') = OptVal(G + F'')$. Moreover, x does obviously not occur in $G + F''$. □

Example 3. Let F be the formula

$$F = \{\ (2,\{y,x\}),\ (1,\{\bar{y},\bar{x}\}),\ (2,\{\bar{x},\bar{z}\}),\ (1,\{x,\bar{z}\}),\ (1,\{y,\bar{z}\})\ \}$$

with double companion x. We get

$$F' = \{\ (2,\{y,x\}),\ (1,\{\bar{y},\bar{x}\}),\ (2,\{\bar{x},\bar{z}\}),\ (1,\{x,\bar{z}\})\ \}$$

and thus $OptVal(F'[y,z]) = 2$, $OptVal(F'[y,\bar{z}]) = 5$, $OptVal(F'[\bar{y},z]) = 1$, and $OptVal(F'[\bar{y},\bar{z}]) = 4$. By the double companion reduction rule, F' reduces to

$$G = \{(2+5+1+4, \mathbf{T}),\ (-4,\{y,z\}),\ (-1,\{y,\bar{z}\}),\ (-5,\{\bar{y},z\}),\ (-2,\{\bar{y},\bar{z}\})\},$$

and we obtain

$$G + F'' = \{\ (12,\mathbf{T}),\ (-4,\{y,z\}),\ (-5,\{\bar{y},z\}),\ (-2,\{\bar{y},\bar{z}\})\ \}.$$

The optimum assignment $y = 1, z = 0$ satisfies clauses weighted six in both $G + F''$ and 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. Algorithm B 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 $t \le m$ is the number of different clause types.

Algorithm B
Input: A Max-2-SAT-formula F
Output: $OptVal(F)$
Let D be the result of Algorithm A on G_F;
$r \leftarrow 0$;
for all assignments A on D **do**
    $F' \leftarrow F[A]$;
    Reduce F' by the (double) companion reduction rule while possible;
    $t \leftarrow OptVal(F')$;
    if $t > r$ **then** $r \leftarrow t$ **fi**
od;
return r

Fig. 6. An algorithm for Max-2-SAT that uses Algorithm A to find a small set of variables for which all assignments have to be tested.

It turns out that we need not use the connectivity graph explicitly. Instead, we can employ a recursive procedure as described in Algorithm C. 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 C 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 C
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 $\max\{\text{Algorithm C}(F[x]), \text{Algorithm C}(F[\bar{x}])\}$
fi

Fig. 7. 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 6.

Theorem 5. *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.

We redefine the potential function ψ:

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

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 preceeded by $\langle 3, 3, 3, 3 \rangle$: Clearly, it cannot be preceeded 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 6. *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

---

Algorithm A′
Input: A graph $G = (V, E)$
Output: $D \subseteq V$, $|D| \leq |E|/5.217 + 1$, such that $R'(G - D)$ has no edges
$D \leftarrow \emptyset$;
while there is a node v with $\deg(v) \geq 3$ **do**
  choose a node v with maximum degree,
  avoiding the case $\langle 4, 4, 4, 4 \rangle$ if possible;
  $D \leftarrow D \cup \{v\}$; $G \leftarrow R'_v(G)$
od;
return D

Fig. 8. A slightly more complicated variant of Algorithm A

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. $\square$

Modifiying Algorithm A according to the above result, as depicted in Figure 8, leads to the following improved running times.

Corollary 2. *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.*

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 [5, p. 174].

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

Acknowledgements. We would like to thank Hans Bodlaender for valuable pointers on literature.

References

1. S. Arora and C. Lund. Hardness of approximation. In D. Hochbaum, editor, *Approximation algorithms for NP-hard problems*, chapter 10, pages 399–446. PWS Publishing Company, Boston, 1997.
2. T. Asano and D. P. Williamson. Improved approximation algorithms for MAX SAT. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, 2000.
3. N. Bansal and V. Raman. Upper bounds for MaxSat: Further improved. In *Proceedings of the 10th International Symposium on Algorithms and Computation (ISAAC)*, number 1741 in Lecture Notes in Computer Science, pages 247–258, Chennai, India, December 1999. Springer-Verlag.
4. H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–21, 1993.
5. A. Brandstädt, V. B. Le, and J. P. Spinrad. *Graph Classes: A Survey*. SIAM monographs on discrete mathematics and applications. SIAM, 1999.

6. J. Chen and I. Kanj. Improved exact algorithms for MAX-SAT. In *Proceedings of the 5th Symposium on Latin American Theoretical Informatics*, number 2286 in Lecture Notes in Computer Science, pages 341–355, Cancun, Mexico, 2002. Springer-Verlag.
7. E. Dantsin, M. Gavrilovich, E. A. Hirsch, and B. Konev. Approximation algorithms for Max SAT: A better performance ratio at the cost of a longer running time. Technical Report PDMI preprint 14/1998, Steklov Institute of Mathematics at St. Petersburg, 1998. Available at `http://logic.pdmi.ras.ru/~hirsch/`.
8. E. Dantsin, A. Goerdt, E. A. Hirsch, and U. Schöning. Deterministic algorithms for k-SAT based on covering codes and local search. In U. Montanari, J. D. P. Rolim, and E. Welzl, editors, *Proceedings of the 27th International Colloquium on Automata, Languages, and Programming (ICALP)*, number 1853 in Lecture Notes in Computer Science. Springer-Verlag, July 2000.
9. S. S. Fedin and A. S. Kulikov. A $2^{|E|/4}$-time algorithm for Max-Cut. In *Zapiski nauchnyh seminarov POMI*, volume 293, pages 129–138, 2002. English translation is to appear in Journal of Mathematical Sciences.
10. U. Feige and M. X. Goemans. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In *3d IEEE Israel Symposium on the Theory of Computing and Systems*, pages 182–189, 1995.
11. J. Gramm, E. A. Hirsch, R. Niedermeier, and P. Rossmanith. New worst-case upper bounds for MAX-2-SAT with application to MAX-CUT. Invited for submission to a special issue of *Discrete Applied Mathematics*. Preliminary version available as ECCC Technical Report R00-037, Trier, Fed. Rep. of Germany, May 2000.
12. 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.
13. J. Gramm and R. Niedermeier. Faster exact solutions for Max2Sat. In *Proceedings of the 4th Italian Conference on Algorithms and Complexity*, number 1767 in Lecture Notes in Computer Science, pages 174–186, Rome, Italy, March 2000. Springer-Verlag.
14. J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 1–10, 1997.
15. E. A. Hirsch. A new algorithm for MAX-2-SAT. In *Proceedings of the 17th Symposium on Theoretical Aspects of Computer Science (STACS)*, number 1770 in Lecture Notes in Computer Science. Springer-Verlag, 2000.
16. E. A. Hirsch. New worst-case upper bounds for SAT. *Journal of Automated Reasoning*, 24(4):397–420, 2000.
17. K. Iwama and S. Tamaki. Improved upper bounds for 3-sat. Technical Report TR03-053, ECCC Trier, 2003.
18. H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *Proceedings of the 38th IEEE Symposium on Foundations of Computer Science*, pages 406–415, 1997.
19. T. Kloks. *Treewidth.* Number 842 in Lecture Notes in Computer Science. Springer-Verlag, 1994.
20. T. Kloks and H. Bodlaender. Only few graphs have bounded treewidth. Technical Report RUU-CS-92-35, Utrecht University, Dept. of Comp. Science, 1992.
21. J. Kneis, D. Mölle, S. Richter, and P. Rossmanith. On the parameterized complexity of exact satisfiability problems. Submitted for publication, 2005.
22. O. Kullmann. New methods for 3-SAT decision and worst-case analysis. *Theoretical Computer Science*, 223:1–72, 1999.
23. O. Kullmann and H. Luckhardt. Deciding propositional tautologies: Algorithms and their complexity. *Information and Computation*, 1997. Submitted. Available at `http://mi.informatik.uni-frankfurt.de/people/kullmann/papers.html`.
24. B. A. Madsen and P. Rossmanith. Maximum exact satisfiability: NP-completeness proofs and exact algorithms. Technical Report RS-04-19, BRICS, October 2004.
25. M. Mahajan and V. Raman. Parameterizing above guaranteed values: MaxSat and MaxCut. *Journal of Algorithms*, 31:335–354, 1999.
26. B. Monien and E. Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Informatica*, 22:115–123, 1985.
27. R. Niedermeier and P. Rossmanith. New upper bounds for maximum satisfiability. *Journal of Algorithms*, 36:63–88, 2000.
28. C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

29. R. Paturi, P. Pudlák, M. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science*, pages 628–637, 1998.
30. N. Robertson and P. D. Seymour. Graph minors I. Excluding a forest. *Journal on Combinatorial Theory Series B*, 35:39–61, 1983.
31. U. Schöning. A probabilistic algorithm for k-SAT and constraint satisfaction problems. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 410–414, 1999.
32. 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 Lecture Notes in Computer Science, pages 382–395. Springer-Verlag, 2003.
33. P. D. Seymour and R. Thomas. Graph searching and a min-max theorem for tree-width. *J. of Combinatorial Theory*, 58:22–33, 1993.
34. 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.
35. R. Williams. A new algorithm for optimal constraint satisfaction and its implications. In *Proceedings of the 31st International Colloquium on Automata, Languages, and Programming (ICALP)*, number 3142 in Lecture Notes in Computer Science, pages 1227–1237. Springer-Verlag, 2004.
36. M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994.

Aachener Informatik-Berichte

This is a list of recent technical reports. To obtain copies of technical reports please consult http://aib.informatik.rwth-aachen.de/ or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

1987-01 *	Fachgruppe Informatik: Jahresbericht 1986
1987-02 *	David de Frutos Escrig, Klaus Indermark: Equivalence Relations of Non-Deterministic Ianov-Schemes
1987-03 *	Manfred Nagl: A Software Development Environment based on Graph Technology
1987-04 *	Claus Lewerentz, Manfred Nagl, Bernhard Westfechtel: On Integration Mechanisms within a Graph-Based Software Development Environment
1987-05 *	Reinhard Rinn: Über Eingabeanomalien bei verschiedenen Inferenzmodellen
1987-06 *	Werner Damm, Gert Döhmen: Specifying Distributed Computer Architectures in AADL*
1987-07 *	Gregor Engels, Claus Lewerentz, Wilhelm Schäfer: Graph Grammar Engineering: A Software Specification Method
1987-08 *	Manfred Nagl: Set Theoretic Approaches to Graph Grammars
1987-09 *	Claus Lewerentz, Andreas Schürr: Experiences with a Database System for Software Documents
1987-10 *	Herbert Klaeren, Klaus Indermark: A New Implementation Technique for Recursive Function Definitions
1987-11 *	Rita Loogen: Design of a Parallel Programmable Graph Reduction Machine with Distributed Memory
1987-12	J. Börstler, U. Möncke, R. Wilhelm: Table compression for tree automata
1988-01 *	Gabriele Esser, Johannes Rückert, Frank Wagner: Gesellschaftliche Aspekte der Informatik
1988-02 *	Peter Martini, Otto Spaniol: Token-Passing in High-Speed Backbone Networks for Campus-Wide Environments
1988-03 *	Thomas Welzel: Simulation of a Multiple Token Ring Backbone
1988-04 *	Peter Martini: Performance Comparison for HSLAN Media Access Protocols
1988-05 *	Peter Martini: Performance Analysis of Multiple Token Rings
1988-06 *	Andreas Mann, Johannes Rückert, Otto Spaniol: Datenfunknetze
1988-07 *	Andreas Mann, Johannes Rückert: Packet Radio Networks for Data Exchange
1988-08 *	Andreas Mann, Johannes Rückert: Concurrent Slot Assignment Protocol for Packet Radio Networks
1988-09 *	W. Kremer, F. Reichert, J. Rückert, A. Mann: Entwurf einer Netzwerktopologie für ein Mobilfunknetz zur Unterstützung des öffentlichen Straßenverkehrs
1988-10 *	Kai Jakobs: Towards User-Friendly Networking
1988-11 *	Kai Jakobs: The Directory - Evolution of a Standard
1988-12 *	Kai Jakobs: Directory Services in Distributed Systems - A Survey
1988-13 *	Martine Schümmer: RS-511, a Protocol for the Plant Floor

1988-14 * U. Quernheim: Satellite Communication Protocols - A Performance Comparison Considering On-Board Processing

1988-15 * Peter Martini, Otto Spaniol, Thomas Welzel: File Transfer in High Speed Token Ring Networks: Performance Evaluation by Approximate Analysis and Simulation

1988-16 * Fachgruppe Informatik: Jahresbericht 1987

1988-17 * Wolfgang Thomas: Automata on Infinite Objects

1988-18 * Michael Sonnenschein: On Petri Nets and Data Flow Graphs

1988-19 * Heiko Vogler: Functional Distribution of the Contextual Analysis in Block-Structured Programming Languages: A Case Study of Tree Transducers

1988-20 * Thomas Welzel: Einsatz des Simulationswerkzeuges QNAP2 zur Leistungsbewertung von Kommunikationsprotokollen

1988-21 * Th. Janning, C. Lewerentz: Integrated Project Team Management in a Software Development Environment

1988-22 * Joost Engelfriet, Heiko Vogler: Modular Tree Transducers

1988-23 * Wolfgang Thomas: Automata and Quantifier Hierarchies

1988-24 * Uschi Heuter: Generalized Definite Tree Languages

1989-01 * Fachgruppe Informatik: Jahresbericht 1988

1989-02 * G. Esser, J. Rückert, F. Wagner (Hrsg.): Gesellschaftliche Aspekte der Informatik

1989-03 * Heiko Vogler: Bottom-Up Computation of Primitive Recursive Tree Functions

1989-04 * Andy Schürr: Introduction to PROGRESS, an Attribute Graph Grammar Based Specification Language

1989-05 J. Börstler: Reuse and Software Development - Problems, Solutions, and Bibliography (in German)

1989-06 * Kai Jakobs: OSI - An Appropriate Basis for Group Communication?

1989-07 * Kai Jakobs: ISO's Directory Proposal - Evolution, Current Status and Future Problems

1989-08 * Bernhard Westfechtel: Extension of a Graph Storage for Software Documents with Primitives for Undo/Redo and Revision Control

1989-09 * Peter Martini: High Speed Local Area Networks - A Tutorial

1989-10 * P. Davids, Th. Welzel: Performance Analysis of DQDB Based on Simulation

1989-11 * Manfred Nagl (Ed.): Abstracts of Talks presented at the WG '89 15th International Workshop on Graphtheoretic Concepts in Computer Science

1989-12 * Peter Martini: The DQDB Protocol - Is it Playing the Game?

1989-13 * Martine Schümmer: CNC/DNC Communication with MAP

1989-14 * Martine Schümmer: Local Area Networks for Manufactoring Environments with hard Real-Time Requirements

1989-15 * M. Schümmer, Th. Welzel, P. Martini: Integration of Field Bus and MAP Networks - Hierarchical Communication Systems in Production Environments

1989-16 * G. Vossen, K.-U. Witt: SUXESS: Towards a Sound Unification of Extensions of the Relational Data Model

1989-17 * J. Derissen, P. Hruschka, M.v.d. Beeck, Th. Janning, M. Nagl: Integrating Structured Analysis and Information Modelling
1989-18 A. Maassen: Programming with Higher Order Functions
1989-19 * Mario Rodriguez-Artalejo, Heiko Vogler: A Narrowing Machine for Syntax Directed BABEL
1989-20 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Graph-based Implementation of a Functional Logic Language
1990-01 * Fachgruppe Informatik: Jahresbericht 1989
1990-02 * Vera Jansen, Andreas Potthoff, Wolfgang Thomas, Udo Wermuth: A Short Guide to the AMORE System (Computing Automata, MOnoids and Regular Expressions)
1990-03 * Jerzy Skurczynski: On Three Hierarchies of Weak SkS Formulas
1990-04 R. Loogen: Stack-based Implementation of Narrowing
1990-05 H. Kuchen, A. Wagener: Comparison of Dynamic Load Balancing Strategies
1990-06 * Kai Jakobs, Frank Reichert: Directory Services for Mobile Communication
1990-07 * Kai Jakobs: What's Beyond the Interface - OSI Networks to Support Cooperative Work
1990-08 * Kai Jakobs: Directory Names and Schema - An Evaluation
1990-09 * Ulrich Quernheim, Dieter Kreuer: Das CCITT - Signalisierungssystem Nr. 7 auf Satellitenstrecken; Simulation der Zeichengabestrecke
1990-11 H. Kuchen, R. Loogen, J.J. Moreno Navarro, M. Rodriguez Artalejo: Lazy Narrowing in a Graph Machine
1990-12 * Kai Jakobs, Josef Kaltwasser, Frank Reichert, Otto Spaniol: Der Computer fährt mit
1990-13 * Rudolf Mathar, Andreas Mann: Analyzing a Distributed Slot Assignment Protocol by Markov Chains
1990-14 A. Maassen: Compilerentwicklung in Miranda - ein Praktikum in funktionaler Programmierung (written in german)
1990-15 * Manfred Nagl, Andreas Schürr: A Specification Environment for Graph Grammars
1990-16 A. Schürr: PROGRESS: A VHL-Language Based on Graph Grammars
1990-17 * Marita Möller: Ein Ebenenmodell wissensbasierter Konsultationen - Unterstützung für Wissensakquisition und Erklärungsfähigkeit
1990-18 * Eric Kowalewski: Entwurf und Interpretation einer Sprache zur Beschreibung von Konsultationsphasen in Expertensystemen
1990-20 Y. Ortega Mallen, D. de Frutos Escrig: A Complete Proof System for Timed Observations
1990-21 * Manfred Nagl: Modelling of Software Architectures: Importance, Notions, Experiences
1990-22 H. Fassbender, H. Vogler: A Call-by-need Implementation of Syntax Directed Functional Programming
1991-01 Guenther Geiler (ed.), Fachgruppe Informatik: Jahresbericht 1990
1991-03 B. Steffen, A. Ingolfsdottir: Characteristic Formulae for Processes with Divergence
1991-04 M. Portz: A new class of cryptosystems based on interconnection networks

1991-05 H. Kuchen, G. Geiler: Distributed Applicative Arrays
1991-06 * Ludwig Staiger: Kolmogorov Complexity and Hausdorff Dimension
1991-07 * Ludwig Staiger: Syntactic Congruences for w-languages
1991-09 * Eila Kuikka: A Proposal for a Syntax-Directed Text Processing System
1991-10 K. Gladitz, H. Fassbender, H. Vogler: Compiler-based Implementation of Syntax-Directed Functional Programming
1991-11 R. Loogen, St. Winkler: Dynamic Detection of Determinism in Functional Logic Languages
1991-12 * K. Indermark, M. Rodriguez Artalejo (Eds.): Granada Workshop on the Integration of Functional and Logic Programming
1991-13 * Rolf Hager, Wolfgang Kremer: The Adaptive Priority Scheduler: A More Fair Priority Service Discipline
1991-14 * Andreas Fasbender, Wolfgang Kremer: A New Approximation Algorithm for Tandem Networks with Priority Nodes
1991-15 J. Börstler, A. Zündorf: Revisiting extensions to Modula-2 to support reusability
1991-16 J. Börstler, Th. Janning: Bridging the gap between Requirements Analysis and Design
1991-17 A. Zündorf, A. Schürr: Nondeterministic Control Structures for Graph Rewriting Systems
1991-18 * Matthias Jarke, John Mylopoulos, Joachim W. Schmidt, Yannis Vassiliou: DAIDA: An Environment for Evolving Information Systems
1991-19 M. Jeusfeld, M. Jarke: From Relational to Object-Oriented Integrity Simplification
1991-20 G. Hogen, A. Kindler, R. Loogen: Automatic Parallelization of Lazy Functional Programs
1991-21 * Prof. Dr. rer. nat. Otto Spaniol: ODP (Open Distributed Processing): Yet another Viewpoint
1991-22 H. Kuchen, F. Lücking, H. Stoltze: The Topology Description Language TDL
1991-23 S. Graf, B. Steffen: Compositional Minimization of Finite State Systems
1991-24 R. Cleaveland, J. Parrow, B. Steffen: The Concurrency Workbench: A Semantics Based Tool for the Verification of Concurrent Systems
1991-25 * Rudolf Mathar, Jürgen Mattfeldt: Optimal Transmission Ranges for Mobile Communication in Linear Multihop Packet Radio Networks
1991-26 M. Jeusfeld, M. Staudt: Query Optimization in Deductive Object Bases
1991-27 J. Knoop, B. Steffen: The Interprocedural Coincidence Theorem
1991-28 J. Knoop, B. Steffen: Unifying Strength Reduction and Semantic Code Motion
1991-30 T. Margaria: First-Order theories for the verification of complex FSMs
1991-31 B. Steffen: Generating Data Flow Analysis Algorithms from Modal Specifications
1992-01 Stefan Eherer (ed.), Fachgruppe Informatik: Jahresbericht 1991
1992-02 * Bernhard Westfechtel: Basismechanismen zur Datenverwaltung in strukturbezogenen Hypertextsystemen
1992-04 S. A. Smolka, B. Steffen: Priority as Extremal Probability
1992-05 * Matthias Jarke, Carlos Maltzahn, Thomas Rose: Sharing Processes: Team Coordination in Design Repositories

1992-06 O. Burkart, B. Steffen: Model Checking for Context-Free Processes

1992-07 * Matthias Jarke, Klaus Pohl: Information Systems Quality and Quality Information Systems

1992-08 * Rudolf Mathar, Jürgen Mattfeldt: Analyzing Routing Strategy NFP in Multihop Packet Radio Networks on a Line

1992-09 * Alfons Kemper, Guido Moerkotte: Grundlagen objektorientierter Datenbanksysteme

1992-10 Matthias Jarke, Manfred Jeusfeld, Andreas Miethsam, Michael Gocek: Towards a logic-based reconstruction of software configuration management

1992-11 Werner Hans: A Complete Indexing Scheme for WAM-based Abstract Machines

1992-12 W. Hans, R. Loogen, St. Winkler: On the Interaction of Lazy Evaluation and Backtracking

1992-13 * Matthias Jarke, Thomas Rose: Specification Management with CAD

1992-14 Th. Noll, H. Vogler: Top-down Parsing with Simultaneous Evaluation on Noncircular Attribute Grammars

1992-15 A. Schuerr, B. Westfechtel: Graphgrammatiken und Graphersetzungssysteme(written in german)

1992-16 * Graduiertenkolleg Informatik und Technik (Hrsg.): Forschungsprojekte des Graduiertenkollegs Informatik und Technik

1992-17 M. Jarke (ed.): ConceptBase V3.1 User Manual

1992-18 * Clarence A. Ellis, Matthias Jarke (Eds.): Distributed Cooperation in Integrated Information Systems - Proceedings of the Third International Workshop on Intelligent and Cooperative Information Systems

1992-19-00 H. Kuchen, R. Loogen (eds.): Proceedings of the 4th Int. Workshop on the Parallel Implementation of Functional Languages

1992-19-01 G. Hogen, R. Loogen: PASTEL - A Parallel Stack-Based Implementation of Eager Functional Programs with Lazy Data Structures (Extended Abstract)

1992-19-02 H. Kuchen, K. Gladitz: Implementing Bags on a Shared Memory MIMD-Machine

1992-19-03 C. Rathsack, S.B. Scholz: LISA - A Lazy Interpreter for a Full-Fledged Lambda-Calculus

1992-19-04 T.A. Bratvold: Determining Useful Parallelism in Higher Order Functions

1992-19-05 S. Kahrs: Polymorphic Type Checking by Interpretation of Code

1992-19-06 M. Chakravarty, M. Köhler: Equational Constraints, Residuation, and the Parallel JUMP-Machine

1992-19-07 J. Seward: Polymorphic Strictness Analysis using Frontiers (Draft Version)

1992-19-08 D. Gärtner, A. Kimms, W. Kluge: pi-Red^+ - A Compiling Graph-Reduction System for a Full Fledged Lambda-Calculus

1992-19-09 D. Howe, G. Burn: Experiments with strict STG code

1992-19-10 J. Glauert: Parallel Implementation of Functional Languages Using Small Processes

1992-19-11 M. Joy, T. Axford: A Parallel Graph Reduction Machine

1992-19-12 A. Bennett, P. Kelly: Simulation of Multicache Parallel Reduction

1992-19-13 K. Langendoen, D.J. Agterkamp: Cache Behaviour of Lazy Functional Programs (Working Paper)
1992-19-14 K. Hammond, S. Peyton Jones: Profiling scheduling strategies on the GRIP parallel reducer
1992-19-15 S. Mintchev: Using Strictness Information in the STG-machine
1992-19-16 D. Rushall: An Attribute Grammar Evaluator in Haskell
1992-19-17 J. Wild, H. Glaser, P. Hartel: Statistics on storage management in a lazy functional language implementation
1992-19-18 W.S. Martins: Parallel Implementations of Functional Languages
1992-19-19 D. Lester: Distributed Garbage Collection of Cyclic Structures (Draft version)
1992-19-20 J.C. Glas, R.F.H. Hofman, W.G. Vree: Parallelization of Branch-and-Bound Algorithms in a Functional Programming Environment
1992-19-21 S. Hwang, D. Rushall: The nu-STG machine: a parallelized Spineless Tagless Graph Reduction Machine in a distributed memory architecture (Draft version)
1992-19-22 G. Burn, D. Le Metayer: Cps-Translation and the Correctness of Optimising Compilers
1992-19-23 S.L. Peyton Jones, P. Wadler: Imperative functional programming (Brief summary)
1992-19-24 W. Damm, F. Liu, Th. Peikenkamp: Evaluation and Parallelization of Functions in Functional + Logic Languages (abstract)
1992-19-25 M. Kesseler: Communication Issues Regarding Parallel Functional Graph Rewriting
1992-19-26 Th. Peikenkamp: Charakterizing and representing neededness in functional loginc languages (abstract)
1992-19-27 H. Doerr: Monitoring with Graph-Grammars as formal operational Models
1992-19-28 J. van Groningen: Some implementation aspects of Concurrent Clean on distributed memory architectures
1992-19-29 G. Ostheimer: Load Bounding for Implicit Parallelism (abstract)
1992-20 H. Kuchen, F.J. Lopez Fraguas, J.J. Moreno Navarro, M. Rodriguez Artalejo: Implementing Disequality in a Lazy Functional Logic Language
1992-21 H. Kuchen, F.J. Lopez Fraguas: Result Directed Computing in a Functional Logic Language
1992-22 H. Kuchen, J.J. Moreno Navarro, M.V. Hermenegildo: Independent AND-Parallel Narrowing
1992-23 T. Margaria, B. Steffen: Distinguishing Formulas for Free
1992-24 K. Pohl: The Three Dimensions of Requirements Engineering
1992-25 * R. Stainov: A Dynamic Configuration Facility for Multimedia Communications
1992-26 * Michael von der Beeck: Integration of Structured Analysis and Timed Statecharts for Real-Time and Concurrency Specification
1992-27 W. Hans, St. Winkler: Aliasing and Groundness Analysis of Logic Programs through Abstract Interpretation and its Safety
1992-28 * Gerhard Steinke, Matthias Jarke: Support for Security Modeling in Information Systems Design
1992-29 B. Schinzel: Warum Frauenforschung in Naturwissenschaft und Technik

1992-30 A. Kemper, G. Moerkotte, K. Peithner: Object-Orientation Axiomatised by Dynamic Logic

1992-32 * Bernd Heinrichs, Kai Jakobs: Timer Handling in High-Performance Transport Systems

1992-33 * B. Heinrichs, K. Jakobs, K. Lenßen, W. Reinhardt, A. Spinner: Euro-Bridge: Communication Services for Multimedia Applications

1992-34 C. Gerlhof, A. Kemper, Ch. Kilger, G. Moerkotte: Partition-Based Clustering in Object Bases: From Theory to Practice

1992-35 J. Börstler: Feature-Oriented Classification and Reuse in IPSEN

1992-36 M. Jarke, J. Bubenko, C. Rolland, A. Sutcliffe, Y. Vassiliou: Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis

1992-37 * K. Pohl, M. Jarke: Quality Information Systems: Repository Support for Evolving Process Models

1992-38 A. Zuendorf: Implementation of the imperative / rule based language PROGRES

1992-39 P. Koch: Intelligentes Backtracking bei der Auswertung funktional-logischer Programme

1992-40 * Rudolf Mathar, Jürgen Mattfeldt: Channel Assignment in Cellular Radio Networks

1992-41 * Gerhard Friedrich, Wolfgang Neidl: Constructive Utility in Model-Based Diagnosis Repair Systems

1992-42 * P. S. Chen, R. Hennicker, M. Jarke: On the Retrieval of Reusable Software Components

1992-43 W. Hans, St.Winkler: Abstract Interpretation of Functional Logic Languages

1992-44 N. Kiesel, A. Schuerr, B. Westfechtel: Design and Evaluation of GRAS, a Graph-Oriented Database System for Engineering Applications

1993-01 * Fachgruppe Informatik: Jahresbericht 1992

1993-02 * Patrick Shicheng Chen: On Inference Rules of Logic-Based Information Retrieval Systems

1993-03 G. Hogen, R. Loogen: A New Stack Technique for the Management of Runtime Structures in Distributed Environments

1993-05 A. Zündorf: A Heuristic for the Subgraph Isomorphism Problem in Executing PROGRES

1993-06 A. Kemper, D. Kossmann: Adaptable Pointer Swizzling Strategies in Object Bases: Design, Realization, and Quantitative Analysis

1993-07 * Graduiertenkolleg Informatik und Technik (Hrsg.): Graduiertenkolleg Informatik und Technik

1993-08 * Matthias Berger: k-Coloring Vertices using a Neural Network with Convergence to Valid Solutions

1993-09 M. Buchheit, M. Jeusfeld, W. Nutt, M. Staudt: Subsumption between Queries to Object-Oriented Databases

1993-10 O. Burkart, B. Steffen: Pushdown Processes: Parallel Composition and Model Checking

1993-11 * R. Große-Wienker, O. Hermanns, D. Menzenbach, A. Pollacks, S. Repetzki, J. Schwartz, K. Sonnenschein, B. Westfechtel: Das SUKITS-Projekt: A-posteriori-Integration heterogener CIM-Anwendungssysteme

1993-12 * Rudolf Mathar, Jürgen Mattfeldt: On the Distribution of Cumulated Interference Power in Rayleigh Fading Channels
1993-13 O. Maler, L. Staiger: On Syntactic Congruences for omega-languages
1993-14 M. Jarke, St. Eherer, R. Gallersdoerfer, M. Jeusfeld, M. Staudt: ConceptBase - A Deductive Object Base Manager
1993-15 M. Staudt, H.W. Nissen, M.A. Jeusfeld: Query by Class, Rule and Concept
1993-16 * M. Jarke, K. Pohl, St. Jacobs et al.: Requirements Engineering: An Integrated View of Representation Process and Domain
1993-17 * M. Jarke, K. Pohl: Establishing Vision in Context: Towards a Model of Requirements Processes
1993-18 W. Hans, H. Kuchen, St. Winkler: Full Indexing for Lazy Narrowing
1993-19 W. Hans, J.J. Ruz, F. Saenz, St. Winkler: A VHDL Specification of a Shared Memory Parallel Machine for Babel
1993-20 * K. Finke, M. Jarke, P. Szczurko, R. Soltysiak: Quality Management for Expert Systems in Process Control
1993-21 M. Jarke, M.A. Jeusfeld, P. Szczurko: Three Aspects of Intelligent Cooperation in the Quality Cycle
1994-01 Margit Generet, Sven Martin (eds.), Fachgruppe Informatik: Jahresbericht 1993
1994-02 M. Lefering: Development of Incremental Integration Tools Using Formal Specifications
1994-03 * P. Constantopoulos, M. Jarke, J. Mylopoulos, Y. Vassiliou: The Software Information Base: A Server for Reuse
1994-04 * Rolf Hager, Rudolf Mathar, Jürgen Mattfeldt: Intelligent Cruise Control and Reliable Communication of Mobile Stations
1994-05 * Rolf Hager, Peter Hermesmann, Michael Portz: Feasibility of Authentication Procedures within Advanced Transport Telematics
1994-06 * Claudia Popien, Bernd Meyer, Axel Kuepper: A Formal Approach to Service Import in ODP Trader Federations
1994-07 P. Peters, P. Szczurko: Integrating Models of Quality Management Methods by an Object-Oriented Repository
1994-08 * Manfred Nagl, Bernhard Westfechtel: A Universal Component for the Administration in Distributed and Integrated Development Environments
1994-09 * Patrick Horster, Holger Petersen: Signatur- und Authentifikationsverfahren auf der Basis des diskreten Logarithmusproblems
1994-11 A. Schürr: PROGRES, A Visual Language and Environment for PROgramming with Graph REwrite Systems
1994-12 A. Schürr: Specification of Graph Translators with Triple Graph Grammars
1994-13 A. Schürr: Logic Based Programmed Structure Rewriting Systems
1994-14 L. Staiger: Codes, Simplifying Words, and Open Set Condition
1994-15 * Bernhard Westfechtel: A Graph-Based System for Managing Configurations of Engineering Design Documents
1994-16 P. Klein: Designing Software with Modula-3
1994-17 I. Litovsky, L. Staiger: Finite acceptance of infinite words

1994-18 G. Hogen, R. Loogen: Parallel Functional Implementations: Graphbased vs. Stackbased Reduction

1994-19 M. Jeusfeld, U. Johnen: An Executable Meta Model for Re-Engineering of Database Schemas

1994-20 * R. Gallersdörfer, M. Jarke, K. Klabunde: Intelligent Networks as a Data Intensive Application (INDIA)

1994-21 M. Mohnen: Proving the Correctness of the Static Link Technique Using Evolving Algebras

1994-22 H. Fernau, L. Staiger: Valuations and Unambiguity of Languages, with Applications to Fractal Geometry

1994-24 * M. Jarke, K. Pohl, R. Dömges, St. Jacobs, H. W. Nissen: Requirements Information Management: The NATURE Approach

1994-25 * M. Jarke, K. Pohl, C. Rolland, J.-R. Schmitt: Experience-Based Method Evaluation and Improvement: A Process Modeling Approach

1994-26 * St. Jacobs, St. Kethers: Improving Communication and Decision Making within Quality Function Deployment

1994-27 * M. Jarke, H. W. Nissen, K. Pohl: Tool Integration in Evolving Information Systems Environments

1994-28 O. Burkart, D. Caucal, B. Steffen: An Elementary Bisimulation Decision Procedure for Arbitrary Context-Free Processes

1995-01 * Fachgruppe Informatik: Jahresbericht 1994

1995-02 Andy Schürr, Andreas J. Winter, Albert Zündorf: Graph Grammar Engineering with PROGRES

1995-03 Ludwig Staiger: A Tight Upper Bound on Kolmogorov Complexity by Hausdorff Dimension and Uniformly Optimal Prediction

1995-04 Birgitta König-Ries, Sven Helmer, Guido Moerkotte: An experimental study on the complexity of left-deep join ordering problems for cyclic queries

1995-05 Sophie Cluet, Guido Moerkotte: Efficient Evaluation of Aggregates on Bulk Types

1995-06 Sophie Cluet, Guido Moerkotte: Nested Queries in Object Bases

1995-07 Sophie Cluet, Guido Moerkotte: Query Optimization Techniques Exploiting Class Hierarchies

1995-08 Markus Mohnen: Efficient Compile-Time Garbage Collection for Arbitrary Data Structures

1995-09 Markus Mohnen: Functional Specification of Imperative Programs: An Alternative Point of View of Functional Languages

1995-10 Rainer Gallersdörfer, Matthias Nicola: Improving Performance in Replicated Databases through Relaxed Coherency

1995-11 * M.Staudt, K.von Thadden: Subsumption Checking in Knowledge Bases

1995-12 * G.V.Zemanek, H.W.Nissen, H.Hubert, M.Jarke: Requirements Analysis from Multiple Perspectives: Experiences with Conceptual Modeling Technology

1995-13 * M.Staudt, M.Jarke: Incremental Maintenance of Externally Materialized Views

1995-14 * P.Peters, P.Szczurko, M.Jeusfeld: Oriented Information Management: Conceptual Models at Work

1995-15 * Matthias Jarke, Sudha Ram (Hrsg.): WITS 95 Proceedings of the 5th Annual Workshop on Information Technologies and Systems

1995-16 * W.Hans, St.Winkler, F.Saenz: Distributed Execution in Functional Logic Programming

1996-01 * Jahresbericht 1995

1996-02 Michael Hanus, Christian Prehofer: Higher-Order Narrowing with Definitional Trees

1996-03 * W.Scheufele, G.Moerkotte: Optimal Ordering of Selections and Joins in Acyclic Queries with Expensive Predicates

1996-04 Klaus Pohl: PRO-ART: Enabling Requirements Pre-Traceability

1996-05 Klaus Pohl: Requirements Engineering: An Overview

1996-06 * M.Jarke, W.Marquardt: Design and Evaluation of Computer–Aided Process Modelling Tools

1996-07 Olaf Chitil: The Sigma-Semantics: A Comprehensive Semantics for Functional Programs

1996-08 * S.Sripada: On Entropy and the Limitations of the Second Law of Thermodynamics

1996-09 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP96 - Fifth International Conference on Algebraic and Logic Programming

1996-09-0 Michael Hanus (Ed.): Proceedings of the Poster Session of ALP 96 - Fifth International Conference on Algebraic and Logic Programming: Introduction and table of contents

1996-09-1 Ilies Alouini: An Implementation of Conditional Concurrent Rewriting on Distributed Memory Machines

1996-09-2 Olivier Danvy, Karoline Malmkjær: On the Idempotence of the CPS Transformation

1996-09-3 Víctor M. Gulías, José L. Freire: Concurrent Programming in Haskell

1996-09-4 Sébastien Limet, Pierre Réty: On Decidability of Unifiability Modulo Rewrite Systems

1996-09-5 Alexandre Tessier: Declarative Debugging in Constraint Logic Programming

1996-10 Reidar Conradi, Bernhard Westfechtel: Version Models for Software Configuration Management

1996-11 * C.Weise, D.Lenzkes: A Fast Decision Algorithm for Timed Refinement

1996-12 * R.Dömges, K.Pohl, M.Jarke, B.Lohmann, W.Marquardt: PRO-ART/CE* — An Environment for Managing the Evolution of Chemical Process Simulation Models

1996-13 * K.Pohl, R.Klamma, K.Weidenhaupt, R.Dömges, P.Haumer, M.Jarke: A Framework for Process-Integrated Tools

1996-14 * R.Gallersdörfer, K.Klabunde, A.Stolz, M.Eßmajor: INDIA — Intelligent Networks as a Data Intensive Application, Final Project Report, June 1996

1996-15 * H.Schimpe, M.Staudt: VAREX: An Environment for Validating and Refining Rule Bases

1996-16 * M.Jarke, M.Gebhardt, S.Jacobs, H.Nissen: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization

1996-17 Manfred A. Jeusfeld, Tung X. Bui: Decision Support Components on the Internet

1996-18 Manfred A. Jeusfeld, Mike Papazoglou: Information Brokering: Design, Search and Transformation

1996-19 * P.Peters, M.Jarke: Simulating the impact of information flows in networked organizations

1996-20 Matthias Jarke, Peter Peters, Manfred A. Jeusfeld: Model-driven planning and design of cooperative information systems

1996-21 * G.de Michelis, E.Dubois, M.Jarke, F.Matthes, J.Mylopoulos, K.Pohl, J.Schmidt, C.Woo, E.Yu: Cooperative information systems: a manifesto

1996-22 * S.Jacobs, M.Gebhardt, S.Kethers, W.Rzasa: Filling HTML forms simultaneously: CoWeb architecture and functionality

1996-23 * M.Gebhardt, S.Jacobs: Conflict Management in Design

1997-01 Michael Hanus, Frank Zartmann (eds.): Jahresbericht 1996

1997-02 Johannes Faassen: Using full parallel Boltzmann Machines for Optimization

1997-03 Andreas Winter, Andy Schürr: Modules and Updatable Graph Views for PROgrammed Graph REwriting Systems

1997-04 Markus Mohnen, Stefan Tobies: Implementing Context Patterns in the Glasgow Haskell Compiler

1997-05 * S.Gruner: Schemakorrespondenzaxiome unterstützen die paargrammatische Spezifikation inkrementeller Integrationswerkzeuge

1997-06 Matthias Nicola, Matthias Jarke: Design and Evaluation of Wireless Health Care Information Systems in Developing Countries

1997-07 Petra Hofstedt: Taskparallele Skelette für irregulär strukturierte Probleme in deklarativen Sprachen

1997-08 Dorothea Blostein, Andy Schürr: Computing with Graphs and Graph Rewriting

1997-09 Carl-Arndt Krapp, Bernhard Westfechtel: Feedback Handling in Dynamic Task Nets

1997-10 Matthias Nicola, Matthias Jarke: Integrating Replication and Communication in Performance Models of Distributed Databases

1997-11 * R. Klamma, P. Peters, M. Jarke: Workflow Support for Failure Management in Federated Organizations

1997-13 Markus Mohnen: Optimising the Memory Management of Higher-Order Functional Programs

1997-14 Roland Baumann: Client/Server Distribution in a Structure-Oriented Database Management System

1997-15 George Botorog: High-Level Parallel Programming and the Efficient Implementation of Numerical Algorithms

1998-01 * Fachgruppe Informatik: Jahresbericht 1997

1998-02 Stefan Gruner, Manfred Nagel, Andy Schürr: Fine-grained and Structure-Oriented Document Integration Tools are Needed for Development Processes

1998-03 Stefan Gruner: Einige Anmerkungen zur graphgrammatischen Spezifikation von Integrationswerkzeugen nach Westfechtel, Janning, Lefering und Schürr

1998-04 * O. Kubitz: Mobile Robots in Dynamic Environments

1998-05 Martin Leucker, Stephan Tobies: Truth - A Verification Platform for Distributed Systems

1998-06 * Matthias Oliver Berger: DECT in the Factory of the Future
1998-07 M. Arnold, M. Erdmann, M. Glinz, P. Haumer, R. Knoll, B. Paech, K. Pohl, J. Ryser, R. Studer, K. Weidenhaupt: Survey on the Scenario Use in Twelve Selected Industrial Projects
1998-08 * H. Aust: Sprachverstehen und Dialogmodellierung in natürlichsprachlichen Informationssystemen
1998-09 * Th. Lehmann: Geometrische Ausrichtung medizinischer Bilder am Beispiel intraoraler Radiographien
1998-10 * M. Nicola, M. Jarke: Performance Modeling of Distributed and Replicated Databases
1998-11 * Ansgar Schleicher, Bernhard Westfechtel, Dirk Jäger: Modeling Dynamic Software Processes in UML
1998-12 * W. Appelt, M. Jarke: Interoperable Tools for Cooperation Support using the World Wide Web
1998-13 Klaus Indermark: Semantik rekursiver Funktionsdefinitionen mit Striktheitsinformation
1999-01 * Jahresbericht 1998
1999-02 * F. Huch: Verifcation of Erlang Programs using Abstract Interpretation and Model Checking — Extended Version
1999-03 * R. Gallersdörfer, M. Jarke, M. Nicola: The ADR Replication Manager
1999-04 María Alpuente, Michael Hanus, Salvador Lucas, Germán Vidal: Specialization of Functional Logic Programs Based on Needed Narrowing
1999-05 * W. Thomas (Ed.): DLT 99 - Developments in Language Theory Fourth International Conference
1999-06 * Kai Jakobs, Klaus-Dieter Kleefeld: Informationssysteme für die angewandte historische Geographie
1999-07 Thomas Wilke: CTL+ is exponentially more succinct than CTL
1999-08 Oliver Matz: Dot-Depth and Monadic Quantifier Alternation over Pictures
2000-01 * Jahresbericht 1999
2000-02 Jens Vöge, Marcin Jurdzinski: A Discrete Strategy Improvement Algorithm for Solving Parity Games
2000-04 Andreas Becks, Stefan Sklorz, Matthias Jarke: Exploring the Semantic Structure of Technical Document Collections: A Cooperative Systems Approach
2000-05 Mareike Schoop: Cooperative Document Management
2000-06 Mareike Schoop, Christoph Quix (eds.): Proceedings of the Fifth International Workshop on the Language-Action Perspective on Communication Modelling
2000-07 * Markus Mohnen, Pieter Koopman (Eds.): Proceedings of the 12th International Workshop of Functional Languages
2000-08 Thomas Arts, Thomas Noll: Verifying Generic Erlang Client-Server Implementations
2001-01 * Jahresbericht 2000
2001-02 Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03 Thierry Cachat: The power of one-letter rational languages

2001-04 Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus

2001-05 Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages

2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic

2001-07 Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem

2001-08 Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling

2001-09 Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs

2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures

2001-11 Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung

2002-01 * Jahresbericht 2001

2002-02 Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems

2002-03 Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages

2002-04 Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting

2002-05 Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines

2002-06 Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata

2002-07 Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities

2002-08 Markus Mohnen: An Open Framework for Data-Flow Analysis in Java

2002-09 Markus Mohnen: Interfaces with Default Implementations in Java

2002-10 Martin Leucker: Logics for Mazurkiewicz traces

2002-11 Jürgen Giesl, Hans Zantema: Liveness in Rewriting

2003-01 * Jahresbericht 2002

2003-02 Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting

2003-03 Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations

2003-04 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs

2003-05 Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard

2003-06 Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates

2003-07 Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung

2003-08 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs

2004-01 * Fachgruppe Informatik: Jahresbericht 2003

2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic

2004-03 Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting

2004-04 Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming

2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming

2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming

2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination

2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information

2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity

2004-10 Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules

2005-01 * Fachgruppe Informatik: Jahresbericht 2004

2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: Aachen Summer School Applied IT Security

2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions

2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem

2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots

2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information

2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks

* These reports are only available as a printed version.
Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.