View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Elsevier - Publisher Connector

ACADEMIC PRESS

Available at www.ComputerScienceWeb.com POWERED BY SCIENCE DIRECT®

Journal of Computer and System Sciences 67 (2003) 772–788

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

http://www.elsevier.com/locate/jcss

Improving a fixed parameter tractability time bound for the shadow problem

Peter Heusch, Stefan Porschen,* and Ewald Speckenmeyer

Institut für Informatik, Universität zu Köln, Pohligstr. 1, D-50969 Köln, Germany

Received 14 September 2001; revised 13 November 2002

Abstract

Consider a forest of k trees and n nodes together with a (partial) function σ mapping leaves of the trees to non-root nodes of other trees. Define the shadow of a leaf ℓ to be the subtree rooted at $\sigma(\ell)$. The shadow problem asks whether there is a set S of leaves exactly one from each tree such that none of these leaves lies in the shadow of another leaf in S. This graph theoretical problem as shown in Franco et al. (Discrete Appl. Math. 96 (1999) 89) is equivalent to the falsifiability problem for pure implicational Boolean formulas over n variables with k occurences of the constant false as introduced in: Heusch J. Wiedermann, P. Hajek (Eds.), Proceedings of the Twentieth International Symposium on Mathematical Foundations of Computer Science (MFCS'95), Prague, Czech Republic, Lecture Notes in Computer Science, Vol. 969, Springer, Berlin, 1995, pp. 221–226, where its NP-completeness is shown for arbitrary values of k and a time bound of $O(n^k)$ for fixed k was obtained. In Franco et al. (1999) this bound is improved to $O(n^2k^k)$ showing the problem's fixed parameter tractability (Congr. Numer. 87 (1992) 161). In this paper the bound $O(n^3 3^k)$ is achieved by dynamic programming techniques thus significantly improving the fixed parameter part.
© 2003 Elsevier Science (USA). All rights reserved.

Keywords: Shadow independent set; Shadow pattern; Pure implicational formula; Dynamic programming; Fixed parameter tractability

1. Introduction

The *shadow independent set problem* (*for short shadow problem or SIS*) is a graph theoretical problem first introduced in [3]. It arises from testing the falsifiability of so-called pure implicational formulas [7], which can be transformed into a set of formulas in disjunctive normal

*Corresponding author.

E-mail addresses: heusch@informatik.uni-koeln.de (P. Heusch), porschen@informatik.uni-koeln.de (S. Porschen), esp@informatik.uni-koeln.de (E. Speckenmeyer).

0022-0000/03/$ - see front matter © 2003 Elsevier Science (USA). All rights reserved.
doi:10.1016/S0022-0000(03)00079-5

form (DNF) yielding a forest whose trees are connected by directed edges. The satisfiability problem of propositional logic which is known to have a wide range of applications (see e.g. [5]) can be transformed into a falsifiability problem of the implicational calculus. The shadow independent set problem turns out to be a graph theoretical formulation of the falsifiability problem for a special class of implicational formulas. For this NP-complete problem a fixed parameter tractability [2] time bound can be achieved according to a parameterized classification of the set of its instances. We investigate SIS, which as far as the authors know is a new problem in algorithmic graph theory deserving interest for its own. It may be challenging to find other relevant applications besides fasifiability resp. satisfiability testing. The present paper may be of interest for researchers in propositional logic as well as for researchers in algorithmic graph theory and parameterized complexity theory.

The connection between propositional logic and the shadow problem is outlined in the following. In 1948 Lukasiewicz [9] studied the "pure" implicational calculus (cf. also [10]). Roughly speaking, (pure) implicational formulas are Boolean formulas only containing propositional variables and the Boolean junction $\rightarrow$. The class of implicational formulas is denoted IMP. Lukasiewicz proved that every tautology in pure implicational form can be deduced by the following axiom being a shortest one (for a tree representation cf. Fig. 1):

$$L = [(p \rightarrow q) \rightarrow r] \rightarrow [(r \rightarrow p) \rightarrow (s \rightarrow p)].$$

For each $F \in IMP$ there is a satisfying truth assignment (assigning truth value 1 to the formula's right most variable). The falsifiability problem (FALS) for IMP is NP-complete, which can be shown by reducing the satisfiability problem (SAT) for conjunctive normal form-(CNF-)formulas to FALS for IMP. The NP-completeness of SAT for CNF is a classical result (cf. e.g. [1,4,8]). Each instance F of CNF can be transformed in polynomial time into an instance $I(F)$ of IMP in two steps. First, transform each clause of F separately into an implicational subformula using the following tautologies:

$$\bar{a} \equiv a \rightarrow z,$$
$$a \vee b \equiv \bar{a} \rightarrow b \equiv (a \rightarrow z) \rightarrow b.$$

Here z is a variable not contained in the variable set of F having the meaning of the constant false; and $\bar{x} := \neg x$ denotes the negation of the Boolean variable x. Second, compose the subformulas via the junction $\rightarrow$ and rightmost nested implication $\rightarrow z$ to an appropriate instance $I(F)$ of IMP such that F is satisfiable if and only if $I(F)$ is falsifiable. This is demonstrated by the following

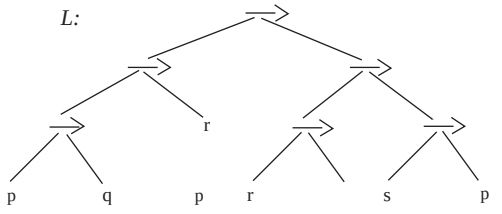

Fig. 1. Tree representation of Lukasiewicz's axiom.

Example. Consider the CNF formula

$$F = (a \vee \bar{b}) \wedge (\bar{a} \vee b \vee c) \wedge (\bar{a} \vee \bar{c}) =: C_1 \wedge C_2 \wedge C_3.$$

Transforming each clause C_i into an equivalent implicational formula I_i $(i = 1, 2, 3)$ yields

$$I_1 = b \to a, \quad I_2 = a \to ((b \to z) \to c), \quad I_3 = a \to (c \to z).$$

Defining $I := I(F) \in IMP$ as $I = (I_1 \to (I_2 \to (I_3 \to z)))$ it is easy to see that F is satisfiable if and only if I is falsifiable. This is illustrated by the so-called *backbone tree representation*. A backbone tree representation for I in the example is shown in Fig. 2. As indicated in the figure, I can be viewed as consisting of its *backbone* denoted as $BB(I)$, to which all the *backbone subformulas*, i.e., all the implicational subformulas I_i, are connected by $\to$. The backbone tree of any such formula I has z as its *right most leaf* denoted $rml(I)$ (cf. Fig. 2). Denoting the set of backbone subformulas by $BS(I)$ it follows that I is falsifiable if and only if there is a truth assignment A such that $A(rml(I)) = 0$ and $A(I') = 1, \forall I' \in BS(I)$.

By the transformation demonstrated in the example, SAT for CNF can be reduced to FALS for IMP. Testing for falsifiability in turn can be done by a recursive backtracking approach. Suppose for example an instance $I \in IMP$ satisfies $rml(I') = z = rml(I)$ for some $I' \in BS(I)$ (cf. Fig. 3). Then simultaneously requiring $A(z) = 0, A(I') = 1$ we have to determine $I_1' \in BS(I')$ such that $A(I_1') = 0$, which according to the previous discussion results in $A(I) = 0$. Consider the substitution for I of $\hat{I}_i \in BS(I')$ $(i = 1, 2)$ as shown in Fig. 4. Here the subformula I' has been removed and $rml(I)$ has been replaced by one of the backbone subformulas of I'. Then I is falsifiable if and only if $\hat{I}_1$ or $\hat{I}_2$ is falsifiable. To handle such situations, proceed inductively by backtracking as described until a falsifying solution is found or the search ends unsuccessfully.

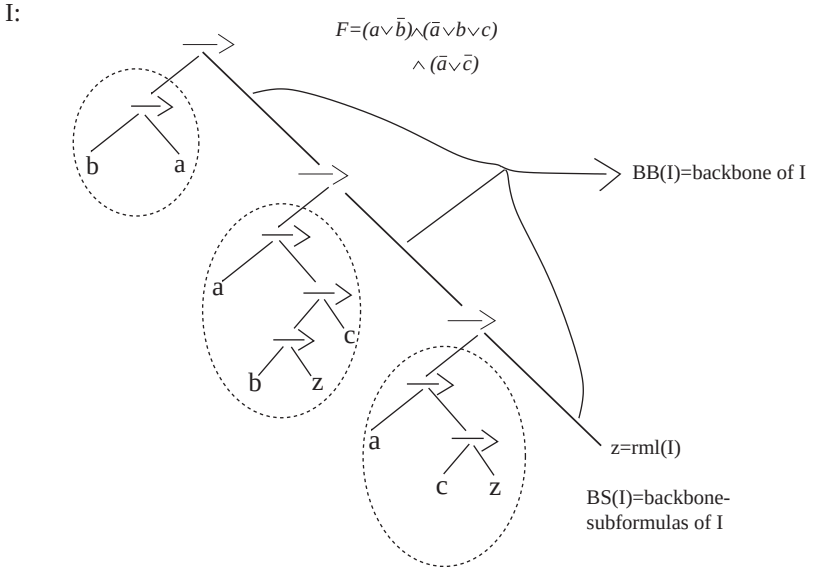

Fig. 2. Example: Backbone-tree representation of I with right most leaf z.

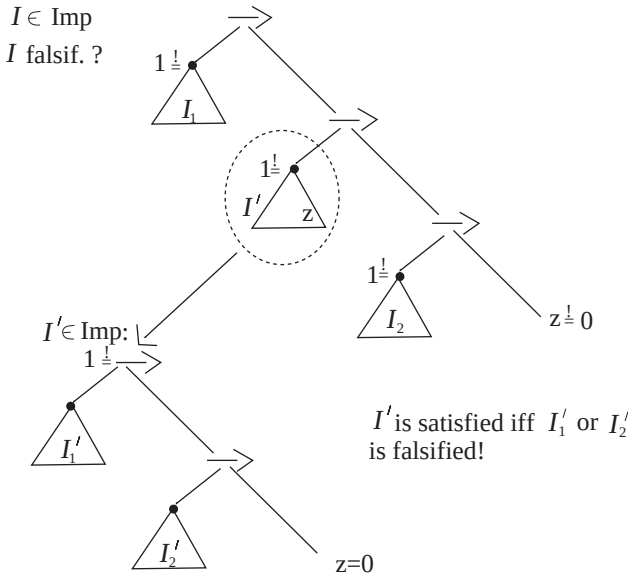

Fig. 3. Subformula I' having the same right most leaf as I.

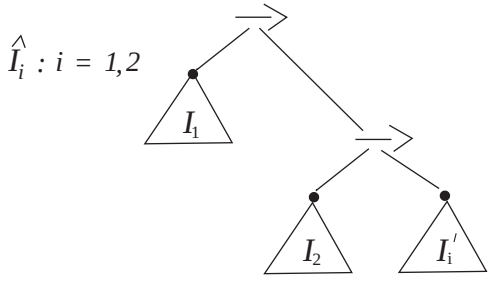

Fig. 4. Replacing I by $\hat{I}_i$ for $I'_i \in BS(I')$.

In the following we are interested in a subclass hierarchy of IMP. For $k \in \mathbb{N}$ consider the set $IMP(k) = \{I \in IMP;$ each variable occurs at most k times in $I\}$. Testing the falsifiability for $I \in IMP(k)$ is known to be NP-complete if $k \geqslant 3$, otherwise this can be decided in quadratic time [6]. There is another interesting class, which is closely related:

$$IMP(2,k) = \{I \in IMP; \text{each variable (besides } z) \text{ occurs in } I \text{ at most twice,}\\ z \text{ occurs in } I \text{ at most } k \text{ times}\}.$$

As shown in [6,7] FALS can be solved for $I \in IMP(2,k)$ in time $O(|I|^k)$. This yields a hierarchy such that FALS is NP-complete for an arbitrary $I \in \bigcup_{k\in\mathbb{N}} IMP(2,k)$. An improvement of this result can be achieved by transforming $I \in IMP(2,k)$ into a set $\Theta := \{\Theta_1, \ldots, \Theta_m\}$ of DNF formulas, simultaneously satisfiable if and only if I is falsifiable. The constraints for this transformation are $|I| = \sum_{i=1}^{m} |\Theta_i|$, each variable occurs at most twice, and z occurs at most k

times in Θ. Each Θ_i has the form $\bar{x} \to (\bar{a} \wedge b \wedge c \cdots) \vee (\bar{d} \wedge e \wedge f \cdots) \vee \cdots \vee (\bar{g} \wedge h \wedge i)$, where x may be any variable or z. As proven in [3] the set Θ can be transformed into a set $T \coloneqq \{T_1, \ldots, T_{k'}\}$ $(k' \leqslant k)$ of trees, whose roots are labeled by $\neg z$, whereas the other nodes are labeled by literals in some disjunct such that satisfying Θ is equivalent to satisfying T in the following sense. A single T_i may be satisfied by choosing one of its leaf nodes ℓ and setting all literals to 1, which label the nodes on the unique path from the root of T_i to ℓ. Choosing an arbitrary leaf of each T_i may yield contradictions, because of literal dependencies. Such a situation is shown in Fig. 5: No leaf contained in the subtree of T_j rooted at node $\neg c, a$, called the *shadow* of the leaf u in T_i (labeled $\neg a, b$), is allowed to be chosen together with u. In this terminology, testing the satisfiability of T means checking whether there is a choice of leaves exactly one from each tree such that no one lies in the shadow of another.

2. Graph theoretical problem formulation

Let $F_k(n)$ $(n, k \in \mathbb{N})$ be a forest with n nodes consisting of k trees. For each tree $T_i \in F_k(n)$ at least one leaf ℓ is mapped to a non-root node $v(\ell)$ of one of the other trees. All leaves of the subtree rooted at $v(\ell)$ are said to lie in the *shadow* $s_{v(\ell)}$ of ℓ. This mapping can be considered as a (partial) function from the set of all leaves in $F_k(n)$ into the set of all its nodes without the trees' roots such that no tree shadows itself.

For a formal definition let us introduce some notation: For a partial map $f : A \to B$ we denote by $D(f) \subseteq A$ its domain. Let T_i be a tree, then $V_i \coloneqq V(T_i)$ denotes the node set, $L_i \coloneqq L(T_i)$ denotes the set of its leaves and $r_i \coloneqq r_{T_i}$ its root. For a finite set U we define the index set $I(U) \coloneqq \{1, \ldots, \lfloor \frac{|U|}{2} \rfloor\}$, its powerset is denoted by 2^U and $\binom{U}{p} \coloneqq \{W \in 2^U; |W| = p\}$. As abbreviation we use $K \coloneqq \{1, \ldots, k\}$.

Definition 2.1. The *shadow* (*independent set*) *problem* (*SIS*) is stated as follows:

Input: $(F_k(n), \sigma)$ $(2 \leqslant k < n)$ where $F_k(n) = \{T_i; i \in K\}$ is a forest $(|V(F_k(n))| = n)$ and $\sigma : \bigcup_{i \in K} L_i \to \bigcup_{i \in K} V_i \backslash \{r_i\}$ is a partial map such that $\sigma(L_i) \cap V_i = \emptyset, D(\sigma) \cap L_i \neq \emptyset$.

Output: Boolean value true (1) if there is a set $S = \{\ell_i \in L_i; i \in K\}$ containing exactly one leaf of each tree such that $\forall \ell_i, \ell_j \in S : \ell_j \notin s_{\sigma(\ell_i)}$, Boolean value false (0) otherwise.

A set S as defined above is called a *shadow independent set* (*of leaves*).

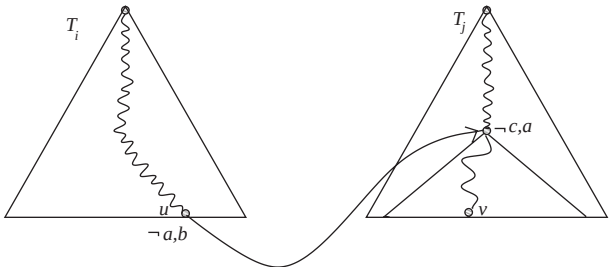

Fig. 5. Contradiction $a \stackrel{!}{=} \neg a \stackrel{!}{=} 1$ caused by dependent leaves u, v.

By representing each T_i as a (super-)node an instance $(F_k(n), \sigma)$ can be viewed as a digraph $G_{k,n}$, the *shadow digraph*, with node set $V(G_{k,n}) = K$ and edge set $E(G_{k,n}) := \{(i,j) \in K^2; \exists \ell \in T_i : \sigma(\ell) \in T_j\}$. (For an arc (i,j) of G the tree T_i is its *source* (*tree*) and T_j is its *sink* (*tree*).) In the sequel we may represent an instance $(F_k(n), \sigma)$ of SIS also by the corresponding shadow digraph $G_{k,n}$ having in mind that the full information on the tree level only is contained in $(F_k(n), \sigma)$.

$G_{k,n}$ reflects the structure of σ: First, consider only the number of arcs in a shadow digraph. A "simple" σ corresponds to a shadow digraph where each node i has exactly one outgoing arc. Following [3] we call this a *shadow pattern* (*SP*), which in general is a subdigraph Π of $G_{k,n}$ having the same node set K. On the tree level a shadow pattern selects for each $i \in K$ all leaves from L_i matching the outgoing arc of T_i, i.e., the set of all leaves that are mapped by σ into the sink of that arc. In the worst case σ may define a $G_{k,n}$ such that each node T_i has one outgoing arc to every other node $T_j \neq T_i$. In this case $G_{k,n}$ is a *complete shadow digraph* containing $(k-1)^k$ different shadow patterns at the same time.

There is a second aspect: $G := G_{k,n}$ may decompose into several weakly connected components (which are the connected components of G formed by ignoring the orientation of each arc): $G := G_1 \cup \cdots \cup G_j$. Here we have $j \leqslant k/2$, because there can be no component having fewer than two nodes. From this point of view a map σ has a more complex structure than another map σ' if the corresponding shadow digraphs satisfy $|E(G')| \leqslant |E(G)|$ and G has fewer components than G'. In what follows assume that G is weakly connected, otherwise treat each weakly connected component separately.

Definition 2.2. Let $(F_k(n), \sigma)$ with the corresponding $G := G_{k,n}$ be fixed. For a node v of the forest denote by $t(v)$ the unique tree it belongs to. Let $M = \{\ell_i \in L(T_i) \cap D(\sigma); i \in K\}$ be a set of leaves containing exactly one from each tree. Then a *shadow pattern* $\Pi(M)$ *induced by* M is defined by restricting G to the outgoing arcs of $t(\ell_i), \forall \ell_i \in M$. If S is a shadow independent set in $(F_k(n), \sigma)$, a shadow pattern Π of G is *consistent with* S if and only if S is also a shadow independent set for the substructure in $(F_k(n), \sigma)$ corresponding to Π.

There may be a variety of shadow patterns consistent with shadow independent sets containing leaves $\ell \notin D(\sigma)$. To incorporate such leaves carry out the following procedure:

Lemma 2.1. *Let $(F_k(n), \sigma)$ be an instance of SIS with shadow digraph $G := G_{k,n}$, let T be a tree in K. For each leaf $\ell \in L(T) \setminus D(\sigma)$ choose exactly one arc of G outgoing from T and associate it with ℓ, let remain empty the shadow of ℓ. Then the following holds*: (i) *The graph G has not changed.* (ii) *For any set of leaves $M = \{\ell_i \in L(T_i); i \in K\}$ there is a unique induced shadow pattern $\Pi(M)$.* (iii) *There has been no shadow independent set generated, nor has any prior one been destroyed.*

Proof. Since only for arcs of G already existing the set of matching leaves in their source trees may have been enlarged, (i) follows. Claim (ii) obviously holds. Finally, there has been generated no shadow independent set, because no shadow has been removed. Also no shadow has been added, therefore no shadow independent set has been destroyed, hence (iii). □

As shown in [3] testing whether there is a shadow independent set in a fixed SP needs time $O(n^2)$. Therefore, by separately checking each SP one obtains for SIS the worst case bound $O(n^2(k-1)^k)$. Exploiting the SP-structural properties of $G_{k,n}$ our aim in the remainder of this paper is to improve the fixed parameter part k^k.

3. Structural properties of shadow patterns

Let $G := G_{k,n}$ $(V(G) = K)$ be a fixed shadow digraph. For $U \subseteq K$ let $\mathscr{S}(U)$ denote the set of all shadow patterns in G with node set U. We have the necessary constraint $|U| \geqslant 2$. Let us recall some simple facts about an arbitrary SP $\Pi \in \mathscr{S}(U)$ (for proofs the reader is refered to [3]):

- In general Π is a digraph consisting of $j \in I(U)$ weakly connected components $\Pi_1, \ldots, \Pi_j$ such that $U = \bigcup_{i=1}^{j} U_i$, $U_i := V(\Pi_i)$. We define $\mathscr{S}_j(U)$ to be the set of those shadow patterns having node set U and consisting of exactly j weakly connected components. Then we have $\mathscr{S}_j(U) \subseteq \mathscr{S}(U)$ and $\mathscr{S}(U) = \bigcup_{j \in I(U)} \mathscr{S}_j(U)$ as disjoint union.
- Each $\Pi_i \in \mathscr{S}_1(U_i)$ is a one component shadow pattern with the same graph structure: it is a root directed tree with exactly one further arc passing from the root to a specific node. In other words each Π_i is a directed cycle, whose nodes are the roots of root directed trees, some or all of which may be single nodes (cf. Fig. 6). Such a tree is called *Π_i-tree* having subtrees whenever it consists of more than the root node (cf. Fig. 7). In a similar way we speak of the Π_i-cycle.

Shadow patterns consisting of two or more components may be identical on some of them as shown in Fig. 8. We are looking for a systematic way to take into consideration such situations in order not to waste too much effort computing information already at hand. Having this in mind it is reasonable to define the following:

Definition 3.1. Let $(F_k(n), \sigma)$ be fixed. For each $i \in I(K)$ define the Boolean-valued function $C_i : 2^K \ni U \mapsto C_i(U) \in \{0, 1\}$ by $C_i(U) = 1$ if and only if there is a shadow independent set S_U on U and $\Pi(S_U) \in \mathscr{S}_i(U)$.

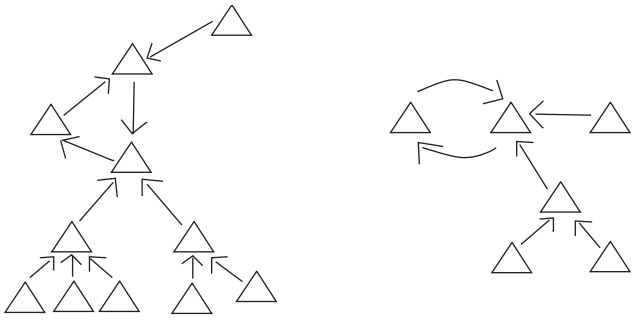

Fig. 6. Examples for the cycle-tree-structure of shadow pattern components.

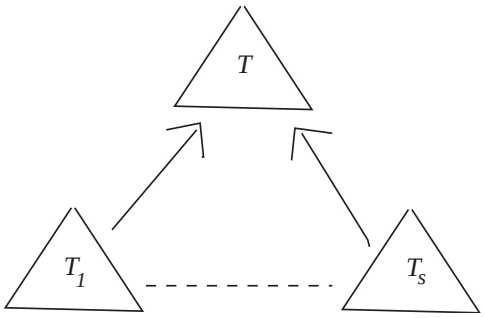

Fig. 7. Subtree rooted at T of a shadow pattern tree.

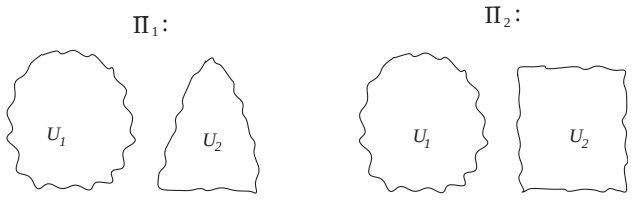

Fig. 8. Shadow patterns Π_1, Π_2 over $U_1 \cup U_2$ such that $\Pi_1|U_1 = \Pi_2|U_1$.

The next assertion can be viewed as the global concept:

Lemma 3.1. *Let G be the shadow digraph of an instance of SIS for which the procedure in Lemma* 2.1 *has been carried out. There is a shadow independent set in G if and only if $C(K) := \bigvee_{i\in I(K)} C_i(K) = 1$. Moreover, suppose the function C_1 is known, then defining $\mathscr{W}_{i-1}(U) := \{W \subset U; 2 \leqslant |W| \leqslant |U| - 2(i-1)\} \subset 2^U$ we have inductively (for fixed $U \in 2^K : |U| \geqslant 2i$):*

$$\forall 2 \leqslant i \in I(U):\ C_i(U) = \bigvee_{W \in \mathscr{W}_{i-1}(U)} [C_1(W) \wedge C_{i-1}(U \backslash W)]. \qquad (*)$$

Proof. Let S be a shadow independent set in G, then for the unique shadow pattern induced by S holds $\Pi(S) \in \mathscr{S}(K) = \bigcup_{i\in I(K)} \mathscr{S}_i(K)$ as disjoint union. The opposite direction is valid by definition. For proving $(*)$ by induction on $2 \leq i \in \mathbb{N}$ we assume (based upon Lemma 2.1) correctness of the values $C_1(U), U \subseteq K : |U| \geqslant 2$. Eq. $(^*)$ obviously is true for $i = 2$ and $U \in 2^K : |U| \geqslant 4$. Now let $i \geqslant 3$ and fix $U : |U| \geqslant 2i$. Observe that on the right-hand side of $(^*)$ each possible i-partition of U is considered which may correspond to a shadow pattern decomposition. Suppose $U_1 \cup \cdots \cup U_i$ is any i-partition of U. Setting $W = U_1$ we obtain by induction that $U_2 \cup \cdots \cup U_i$ is covered by $C_{i-1}(U \backslash W)$. Moreover, for $W \subset U$ there may be a single-component shadow pattern only if $|W| \geqslant 2$. On the other hand there may be a $i-1$-component shadow pattern for $U \backslash W$ only if any component has at least two elements, i.e., only if $|W| \leqslant |U| - 2(i-1)$, in accordance with the constraint $i - 1 \in I(U \backslash W)$. Hence, by restricting to $\mathscr{W}_{i-1}(U)$ for calculating $C_i(U)$ no choice is left. □

In order to be able to compute the value $C_1(U)$ corresponding to a weakly connected shadow pattern with node set U $(U \in 2^K : |U| \geqslant 2)$, we now address the question how a weakly connected SP itself can be composed of two weakly connected parts. As it turns out there are only a few possible cases to separate a weakly connected subgraph $H_1 := \Pi|U_1$ (denoting the restriction) from a SP $\Pi \in \mathscr{S}_1(U)$ such that also the subgraph of the complementary node set $H_2 := \Pi|U_2$ $(U_2 = U'_1 := U \backslash U_1)$ is weakly connected (cf. Fig. 9).

Definition 3.2. For $U : |U| \geqslant 2$ let $\Pi \in \mathscr{S}_1(U)$ be fixed, then for $U_1 \subseteq U$ the induced subdigraph $H_1 := \Pi|U_1$ is called a *shadow pattern part* (*SPP*) if and only if H_1 and also $H_1' := \Pi|U'_1$ are weakly connected (then H_1' itself is a shadow pattern part).

Lemma 3.2. *A SPP H has exactly one of the following two shapes*:

(i) *H is a* (*root directed*) *tree* (*hence, each node has exactly one outgoing arc except for the root*, *which has none*); *in this case H is called* SPP of tree-type.

(ii) *H contains exactly one directed cycle and therefore is itself a shadow pattern* (*hence*, *each node has exactly one outgoing arc*).

Proof. For proceeding by induction on $2 \leqslant |U| \in \mathbb{N}$ fix $\Pi \in \mathscr{S}_1(U)$ where $|U| = 2$. Then Π is a cycle and each proper shadow pattern part is a single node being the root of a root directed tree thus is a tree. Now let us assume that the Lemma is valid for $2 \leqslant |U| \leqslant n \in \mathbb{N}$, and consider $|U| = n + 1$. Splitting Π into two SPPs means: (a) Cutting out a subtree of a Π-tree, if there is one, in which case one part is a tree whose node set is a proper subset of U and by induction each of its SPPs is of shape (i). The other part is a shadow pattern on a smaller node set and by induction each of its SPPs is of shape (i) or (ii). (b) Removing two arcs from the cycle of Π results in two trees of smaller node sets. By induction all their SPPs are trees. Finally, any cutting procedure different from (a), (b) produces more than two connected parts at the same time. □

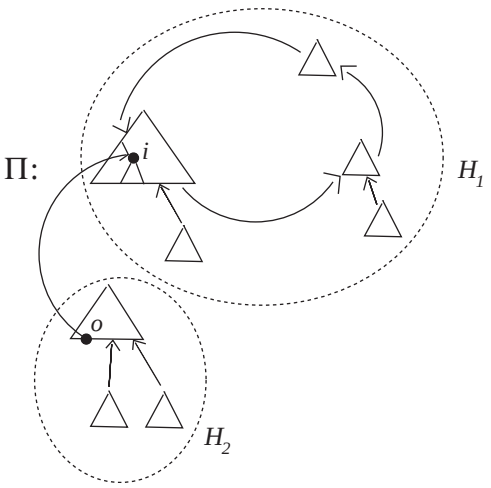

Fig. 9. Shadow pattern parts H_1, H_2 yielding a weakly connected shadow pattern Π by the joining arc (o, i).

Remark. We only have to consider decompositions of a shadow pattern part into exactly two smaller weakly connected components, because in this way, recursively, we consider all partitions into more components.

Now we take the opposite point of view and, consistent with Lemma 3.2, call a digraph H a *SPP* if and only if H is a root directed tree or a shadow pattern.

Lemma 3.3. *Let H_1, H_2 be SPPs with disjoint non-empty node sets $U_i := V(H_i)$ $(i = 1, 2)$. There are only the following cases possible for composing a SPP H of H_1, H_2 without removing any arc such that $U := V(H) = U_1 \cup U_2$:*

(a) *W.l.o.g. let H_1 be a tree, H_2 a shadow pattern: Plug H_1 in H_2 by an arc passing from the root of H_1 to any node of H_2 resulting in a shadow pattern H.*

(b) *Both H_1, H_2 are trees*: (i) *Plug H_1 in H_2 by an arc passing from the root of H_1 to any node of H_2 (or exchanging the roles of H_1, H_2) resulting in a larger tree H.* (ii) *Plug H_1 in H_2 by an arc passing from the root of H_1 to any node of H_2, and simultaneously plug H_2 in H_1 by an arc passing from the root of H_2 to any node of H_1 resulting in a shadow pattern H.*

(c) *Both H_1, H_2 are shadow patterns: they cannot be combined to a larger SPP.*

Proof. Regarding Lemma 3.2 there are no other graph structures possible for H_1, H_2 than those considered in (a)–(c). The procedures described in (a) and (b) yield the corresponding structures for H as claimed. There are no other combination schemes, because the root of an involved tree is the only node having no outgoing arc. Hence, any other scheme composing H would produce a node with two outgoing arcs. Finally, the claim of (c) is true, because H has to be weakly connected, as this is a defining property of any SPP.

The following definition summarizes the cases treated above:

Definition 3.3. Let H be a SPP with node set U, then H is called:

SPP with out-connector H_o iff H is a tree, the root being the out-connector,

SPP with in-out-connector of the first kind H_{io}^1 iff H is a tree, the root at the same time being both connectors,

SPP with in-out-connector of the second kind H_{io}^2 iff H is a tree, the root being the out-connector and any other node of H being the in-connector,

SPP with in-connector H_i iff H is a SP any node being the in-connector,

SPP with empty connectors iff H is an isolated weakly connected SP.

4. A Dynamic programming approach

First we make some remarks concerning the data structures and types used in what follows: Thinking of K as a sorted alphabet each subset $U \subset K : |U| \geqslant 2$ corresponds to a unique word over K denoted $word(U)$ or U for short. Thus 2^K may be sorted by the corresponding lexicographic order. Let $ind(U)$ denote the unique index of U according to this order. Suppose that for each

$U \subseteq K$ we have a data type indexed by $word(U)$ and containing $ind(U)$. Then refering to that data type we can get access to $ind(U)$ in constant time making available in $O(1)$ the Boolean value $C_j(U)$ for each U. For that we make use of two arrays val_i $(i = 0, 1)$ of length $N := 2^k - (k+1)$ $(k := |K|)$ for storing the values $C_j(U)$ $(|U| \geqslant 2)$ by increasing $ind(U)$ during the computation. They are read and updated alternatively and are initialized by a Boolean vector $\mathbf{0}$ (of appropriate length) at the beginning of each iteration.

Algorithm 1 (GLOBAL).

Input: Array C_1 of length N of Boolean entries $C_1(U)(U \subseteq K : |U| \geqslant 2)$ sorted by lexicographic order from 1 to N; Array of subsets $U \in 2^K : |U| \geqslant 2$ sorted by lexicographic order; a data type containing $ind(U)$ for each $U \subseteq K$

Output: $C(K) := \bigvee_{i \in I(K)} C_i(K)$

$C(K) \leftarrow C_1[ind(K)](= C_1(K))$, $j \leftarrow 2$

copy array C_1 to array val_0

while $\neg(C(K) = 1$ **or** $j > |I(K)|)$ **do**

  for all $U \subseteq K : |U| \geqslant 2j$ **do**

    $val_{(j-1) \bmod 2} \leftarrow \mathbf{0}$

      for all $W \in \mathscr{W}_{j-1}(U)$ **do**

        $val_{(j-1) \bmod 2}[ind(U)] \leftarrow C_1[ind(W)] \wedge val_{j \bmod 2}[ind(U \backslash W)]$

          if $val_{(j-1) \bmod 2}[ind(U)] = 1$ **then break fi**

      od $(*$ now: $val_{(j-1) \bmod 2}[ind(U)] = C_j(U)*)$

    od

  $C(K) \leftarrow val_{(j-1) \bmod 2}[ind(K)]$

  $j \leftarrow j + 1$

od

Proposition 4.1. *Algorithm GLOBAL correctly computes $C(K)$, has worst case time complexity $O(k3^k)$ and space complexity $O(2^k)$.*

Proof. The correctness of algorithm GLOBAL follows from Lemma 3.1. Establishing the worst case time bound is not hard: The number of iterations of the two nested for-loops, which are processed for each fixed $j \geqslant 2$, is determined by considering for each $U \subseteq K : |U| = p \geqslant 2j$ all $W \in \mathscr{W}_{j-1}(U) \subset 2^U$, and hence has an upper bound of $O(\sum_{p=2j}^{k} \binom{k}{p} \sum_{i=2}^{p-2(j-1)} \binom{p}{i}) = O(3^k)$. Here the binomial theorem has been used twice. Observing that the while-loop terminates after $O(k)$ iterations proves that the algorithm has time complexity $O(k3^k)$. The space bound follows from the fact that we have to store the $O(2^k)$ values of C_1. Storing the sorted power set 2^K together with the data entry for any subset index also yields a space amount of $O(2^k)$. Finally, the total length of val_i $(i = 0, 1)$ is bounded by $O(2^k)$ thus yielding the space requirement. □

Exploiting the structural features investigated in the previous section enables us to compute C_1 inductively. First we define the following predicate:

Definition 4.1. Let $(F_k(n), \sigma)$ be an instance of SIS. For $L := \bigcup_{j\in K} L(T_j)$ the set of all leaves define $L_e := L \cup \{e_o\}$, where e_o is called *empty outgoing shadow*. Denoting by $D := D(\sigma)$ the domain of σ let $V := V(\sigma) := \sigma(D)$ be the set of all nodes lying in the image of σ. Define $V_e := V \cup \{e_i\}$, where e_i is called *empty incoming shadow*. Then the function $P : 2^K \times L_e \times V_e \to \{0,1\}$ is defined by $P(U,o,i) = 1$ if and only if the following conditions are satisfied:

(i) there is a shadow independent set S_U on U inducing a SPP on U,

(ii) $t(o) \in U$ if $o \in L$, $t(i) \in U$ if $i \in V$,

(iii) the shadow rooted at $i \in V$ contains no element of S_U; and $o \in L \cap S_U$ is not shadowed by any element of S_U.

The next Lemma resting on the possible composition schemes for SPPs described in Lemma 3.3 forms the base of an algorithm computing the function C_1. Let $U \subseteq K : |U| \geqslant 2$ be fixed. In the sequel all $W \subset U$ are assumed to be proper and non-empty: $W' := U \setminus W \neq U$, $W' \neq \emptyset$. By $L|W$ are meant the leaves of only those trees contained in W, similar for L_e, V, V_e.

Lemma 4.1. *Let $(F_k(n), \sigma)$ be an instance of SIS. Supposing that the procedure described in Lemma* 2.1 *has been carried out for the corresponding shadow digraph G for each fixed $U \subseteq K$ $(|U| \geqslant 2)$ the following holds*:

$$C_1(U) = P(U, e_o, e_i) = \bigvee_{W \subset U} \bigvee_{o,o' \in L|U} P(W, o, \sigma(o')) \wedge P(W', o', \sigma(o)). \qquad (*)$$

$P(U,o,i)$ $(\emptyset \neq U \in 2^K, (o,i) \in L|U \times V_e|U)$ is determined inductively:

(i) *initial values*: $\forall T \in K$:

$$\forall (o,i) \in L|\{T\} \times V|\{T\} : P(\{T\}, o, i) = 1 \Leftrightarrow o \notin s_i$$

$$\forall o \in L|\{T\} : P(\{T\}, o, e_i) = 1.$$

(ii) $\forall U \in 2^K : |U| \geqslant 2$:

$\forall o \in L|U$:

$$P(U,o,e_i) = \bigvee_{t(o) \notin W \subset U} \bigvee_{o' \in L|W : \sigma(o') \in V|t(o) \wedge o \notin s_{\sigma(o')}} P(W, o', e_i) \wedge P(W', o, e_i),$$

$\forall o \in L|U, \forall i \in V|t(o)$:

$$P(U,o,i) = \begin{cases} 1, & P(U,o,e_i) = 1 \wedge o \notin s_i, \\ 0 & \text{else} \end{cases}$$

$\forall o \in L|U, \forall i \in V|U, t(i) \neq t(o)$:

$$P(U,o,i) = \bigvee_{t(o) \notin W \subset U : t(i) \in W} \bigvee_{o' \in L|t(i)} P(W, o', i) \wedge P(W', o, \sigma(o')).$$

Proof. For verifying (*) note that $P(U, e_o, e_i)$ corresponds by definition to a weakly connected shadow pattern (with empty connectors) having node set U and being consistent with a shadow independent set. Therefore by definition of C_1 it is valid that $C_1(U) = P(U, e_o, e_i)$. On the right

hand side of (*) all possible shadow independent sets are considered inducing a shadow pattern that is composed of two SPPs of tree type according to the combination scheme in Lemma 3.3(b), (ii). Note that when no shadow independent set is found in (*), then there is no way to compose a shadow pattern corresponding to a shadow independent set of a SPP of tree-type and a SPP containing a cycle. So, we never have to consider predicate values $P(U, e_o, i)$ corresponding to a SPP with in-connector. Thus by induction, there only will be involved predicate values corresponding to SPPs of tree-type, cf. Lemma 3.2. The procedure described in Lemma 2.1 ensuring that any leaf $o \notin D(\sigma)$ matches an outgoing arc of G and thus contributes to a weakly connected component guarantees that all weakly connected shadow patterns yielding a shadow independent set will be detected.

Next, we have to prove the assertions (i), (ii) of the Lemma concerning the predicate values used in equation (*). The correctness of (i) is obvious: A single tree T can be viewed only as a SPP of tree type H_o or H^1_{io} which only contains a root and therefore yields the claimed shadow independent set conditions. The first case in (ii) corresponds to a SPP with out-connector. For this we only have to consider node subsets W not containing the root, because the complement always contains the root. Moreover, it is sufficient to test only SPPs with out-connector (represented by o') having a maximal node subset W in the sense that it is plugged in $t(o) \in W'$, then $o \notin s_{\sigma(o')}$ has to be satisfied. If no such partition exhibits a shadow independent set, there is none in U with respect to o, and if there is one it is detected by this procedure inductively. The next equation of (ii) corresponding to a SPP with in-out-connector of the first kind is obtained from the values $P(U, o, e_i)(\forall U \in 2^K, o \in L|K)$ already computed. Finally, the last equation of (ii) corresponds to a SPP H^2_{io}. Here we never have to consider a node subset not containing $t(i)$, because this situation is already covered by the previous cases. Moreover, by induction it is sufficient only to consider SPPs over node sets $W \subset U$ which might be connected to a SPP over W' by an arc outgoing from a leaf of $t(i)$. □

Next we state an algorithm computing all relevant values of the function C_1. It uses an array $P[]$ storing all predicate values $P(U, o, i)$ $(U \in 2^K, o \in L_e, i \in V_e)$. Initially P is 0 everywhere. It is assumed that the elements of V_e, L_e are labeled from 1 to $|V_e|, |L_e|$ respectively so that the array entry $P(U, o, i)$ is available via the corresponding combined index $(ind(U), o, i)$ in constant time. Thus $P[]$ has bitlength $O(n^2 2^k)$. Moreover we assume that there is a data structure allowing to decide in constant time whether a pair (o, i) of nodes satisfies $o \in s_i$ or not. For convenience we set $\mathscr{W}_p(U) := \{W \subset U; 1 \leqslant |W| \leqslant p\}$.

Algorithm 2 (INITIAL).

Input: weakly connected shadow digraph $G_{k,n}$
Output: Array C_1 with entries $C_1(U), U \subseteq K : |U| \leqslant 2$
for all $(U, o, i) \in 2^K \times L_e \times V_e$ **do**
  $P[ind(U), o, i] \leftarrow 0$
od
for all $(T, o, i) \in K \times L|\{T\} \times V_e|\{T\}$ **do**
  if $i = e_i$ **or** $o \notin s_i$ **then** $P[ind(\{T\}), o, i] \leftarrow 1$ **fi**

od
(∗ main loop: ∗)
for $p = 2$ to k **do**
(∗ step 1: ∗)
for all $(U, o) \in \binom{K}{p} \times L|U$ **do**
for all $(W, o') \in \mathscr{W}_p(U) \times L|W : \sigma(o') \in t(o) \notin W$ **do**
if $o \notin s_{\sigma(o')}$ **then**
$P[ind(U), o, e_i] \leftarrow P[ind(W), o', e_i] \wedge P[ind(W'), o, e_i]$
fi
if $P[ind(U), o, e_i] = 1$ **then** continue to next tupel **fi**
od
od
(∗ step 2: ∗)
for all $(U, o) \in \binom{K}{p} \times L|U$ **do**
if $P[ind(U), o, e_i] = 1$ **then**
for all $i \in V(t(o))$ **do**
if $o \notin s_i$ **then** $P[ind(U), o, i] = 1$, continue to next triple **fi**
od
fi
od
(∗ step 3: ∗)
for all $(U, o, i) \in \binom{K}{p} \times L|U \times V|U$ **do**
for all $(W, o') \in \mathscr{W}_p(U) \times L|t(i) : t(o) \notin W, t(i) \in W$ **do**
$P[ind(U), o, i] \leftarrow P[ind(W), o', i] \wedge P[ind(W'), o, \sigma(o')]$
if $P[ind(U), o, i] = 1$ **then** continue to next triple **fi**
od
od
(∗ step 4: ∗)
for all $U \in \binom{K}{p}$ **do**
for all $(W, o_1, o_2) \in \mathscr{W}_p(U) \times [L|U]^2$ **do**
$C_1[U] \leftarrow P[ind(W), o_1, \sigma(o_2)] \wedge P[ind(W'), o_2, \sigma(o_1)]$
if $C_1[U] = 1$ **then** continue to next subset **fi**
od
od
od

Proposition 4.2. *Algorithm INITIAL correctly computes the values* $C_1(U), U \subseteq K : |U| \geqslant 2$, *has time complexity* $O(n^3 3^k)$ *and space complexity* $O(n^2 2^k)$.

Proof. The correctness of the algorithm follows immediately from Lemma 4.1. Initializing all array entries $P[]$ by 0 has been done for convenience and could be omitted by slightly rewriting the

algorithm introducing some additional branches. This step has time bound $O(n^2 2^k)$ and is dominated by the main loop. Computing all initial predicate values takes time $O(n^2 k)$, because there are k trees each having fewer than n nodes considered as leaves or inner nodes. Step 3 of the main loop never performs more than $\binom{k}{p}(n+1)^2 \sum_{r=1}^{p} n\binom{p}{r} \in O(n^3 \binom{k}{p} 2^p)$ iterations for fixed cardinality $|U| = p$. In each such iteration a constant amount of time is consumed for accessing the corresponding array entries and computing the intermediate values. The other three steps of the main loop in algorithm INITIAL are dominated by step 3. Hence, the summation over all values of p using the binomial theorem yields the time bound $O(n^3 3^k)$, as has been claimed. The space requirements are dominated by array $P[]$ of length $O(n^2 2^k)$.

We are ready to state our complete algorithm deciding the existence of a shadow independent set for an arbitrary instance $(F_k(n), \sigma)$ of SIS.

Algorithm 3 (SIS).

Input: $(F_k(n), \sigma)$
Output: Boolean true if there is a shadow independent set, Boolean false else
compute $G := G_{k,n}$ and its weakly connected components;
($*$ let these be $G_i := G_{k_1,n_1}, \ldots, G_j := G_{k_j,n_j}$, where $K_i := V(G_i), k_i := |K_i|$,
$K := \bigcup_{i=1}^{j} K_i$, $n = \sum_{i=1}^{j} n_i, k = \sum_{i=1}^{j} k_i$.$*$)
for $i = 1$ **to** j **do**
  carry out the procedure described in Lemma 2.1;
  by depth first search for each node, create a two dimensional array of n^2 binary
  entries storing 1 for the pair (ℓ, v) if and only if $\ell \in s_v$;
  sort 2^{K_i} by lexicographic order computing $ind(U), \forall U \in 2^{K_i} : |U| \geqslant 2$;
  compute $C_1(K_i)$ as described in algorithm INITIAL and store the intermediate
  values in array C_1;
  compute $C(K_i) \leftarrow \bigvee_{p \in I(K_i)} C_p(K_i)$ as described in algorithm GLOBAL;
  if $C(K_i) = 0$ **return** $C(K_i)$
  else $C(K) \leftarrow C(K) \wedge C(K_i)$
  fi
od
return $C(K)$

Theorem 4.1. *Algorithm SIS determines the existence of a shadow independent set for an arbitrary instance $(F_k(n), \sigma)$ of SIS in time $O(n^3 3^k)$ and with space requirement $O(n^2 2^k)$.*

Proof. There is a shadow independent set of G if and only if there is one for each G_i. Hence, the algorithm returns the correct value if each of the weakly connected components is processed correctly, which is guaranteed by Propositions 4.1 and 4.2. Computing $G_{k,n}$ and its weakly connected components can be done in time $O(n^2)$ or better. In the loop, first carrying out the

procedure described in Lemma 2.1 and constructing the two-dimensional array by depth first search for each node needs time $O(n^2)$. Next, sorting the power set 2^{K_i} by lexicographic order simultaneously computing $ind(U)$ $(\forall U \in 2^{K_i} : |U| \geqslant 2)$ can be done in time $O(k_i 2^{k_i})$. By Propositions 4.1 and 4.2 for each component G_i with $n_i \geqslant k_i$ nodes the bound $O((k_i + n_i^3)3^{k_i}) = O(n_i^3 3^{k_i})$ is obtained. Summing over all components this yields $O(\sum_{i=1}^{j} [k_i 2^{k_i} + n_i^3 3^{k_i}]) = O(\sum_{i=1}^{j} n_i^3 3^{k_i})$ and in the case $j = 1$ the claimed bound. For verifying the space requirements observe that there has been used a constant number of containers, each of which stores never more than $O(n^2 2^k)$ bits.

5. Discussion

The algorithm developed by Franco et al. in [3], called FGSSS for short, transforms formulas of $IMP(2, k)$ into instances of the shadow problem (SIS) such that an input formula is falsifiable if and only if the corresponding instance of SIS has a solution. By the same transformation it is possible to test falsifiability for formulas from $IMP(2, k)$ via the algorithm SIS developed in this paper. Thus falsifiability for such formulas can be tested in time $O(n^3 3^k)$. Note that, in contrast to FGSSS, algorithm SIS requires exponential space which is typical for dynamic programming procedures designed for NP-complete problems. SIS always performs $O(n^3 3^k)$ steps on a weakly connected shadow digraph $G_{k,n}$ whereas FGSSS accidentally may find a shadow independent set in the shadow pattern tested first and then may terminate in time $O(n^2)$.

An interesting aspect of the computational approach presented here is its decomposition into several nested dynamic programming algorithms. A complex process is performed to compute the induction base for another dynamic programming algorithm. Further, it is remarkable that the tree structure of the nodes of the input shadow digraph G is of little importance. This structure is analyzed only in a preprocessing step for making available in constant time the information whether a given pair (o, i) of nodes satisfies $o \in s_i$. Especially the procedure presented in [3] is not needed checking on the tree level in time $O(n^2)$ whether there is a shadow independent set in a fixed shadow pattern. It follows that algorithm SIS, slightly modified, also would work if $(F_k(n), \sigma)$ is replaced by $(\mathscr{D}_k(n), \sigma)$. Here $\mathscr{D}_k(n)$ denotes a set of k acyclic digraphs of n nodes altogether. A *shadow* then is defined to be the reachability set of a node's image with respect to σ. Since $|E(\mathscr{D})| \in O(n^2)$, the same bound as in the case of forest is guaranteed. This version of the problem may be the key to find applications of (generalizations of) the shadow independent set problem beyond propositional logic. Perhaps, one might imagine it as a modeling tool for certain traffic net questions.

Acknowledgments

The authors thank the anonymous referees for valuable comments significantly improving the presentation of this paper.

References

[1] S.A. Cook, The complexity of theorem proving procedures, in: Proceedings of the Third ACM Symposium on Theory of Computing, 1971, pp. 151–158.
[2] R.G. Downey, M.R. Fellows, Fixed parameter tractability and completeness, Congressus Numerantium 87 (1992) 161–178.
[3] J. Franco, J. Goldsmith, J. Schlipf, E. Speckenmeyer, R.P. Swaminathan, An algorithm for the class of pure implicational formulas, Discrete Appl. Math. 96 (1999) 89–106.
[4] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman and Company, San Francisco, 1979.
[5] J. Gu, P.W. Purdom, J. Franco, B.W. Wah, Algorithms for the satisfiability (SAT) problem: a survey, in: D. Du, J. Gu, P.M. Pardalos (Eds.), Satisfiability Problem: Theory and Applications, DIMACS Workshop, March 11–13, 1996, DIMACS Series, Vol. 35, American Mathematical Society, Providence, RI, 1997, pp. 19–151.
[6] P. Heusch, Implikationen der Implikation, Ph.D. Thesis, Mathematisches Institut, Universität Düsseldorf, Düsseldorf, 1993.
[7] P. Heusch, The complexity of the falsifiability problem for pure implicational formulas, in: J. Wiedermann, P. Hajek (Eds.), Proceedings of the Twentieth International Symposium on Mathematical Foundations of Computer Science (MFCS'95), Prague, Czech Republic, Lecture Notes in Computer Science, Vol. 969, Springer, Berlin, 1995, pp. 221–226.
[8] H. Kleine Büning, T. Lettman, Aussagenlogik: Deduktion und Algorithmen, B.G. Teubner, Stuttgart, 1994.
[9] J. Lukasiewicz, The shortest axiom of the implicational calculus of propositions, Proc. Irish Acad. A52 (1948) 25–33.
[10] J. Lukasiewicz, A. Tarski, Untersuchungen über den Aussagenkalkül, C. R. Soc. Sci. Lett. Varsovie Cl III 23 (1930) 30–50.