Peer-to-Peer Agent Systems for Textual Information Dissemination: Algorithms and Complexity

Manolis Koubarakis and Christos Tryfonopoulos

Dept. of Electronic and Computer Engineering
Technical University of Crete
73100 Chania, Crete, Greece
Tel: +30821037222, Fax: +30821037202

{manolis|trifon}@intelligence.tuc.gr, www.intelligence.tuc.gr

Abstract. We discuss the models $\mathcal{WP}$ and $\mathcal{AWP}$ especially designed for the selective dissemination of textual information by peer-to-peer agent systems. We briefly present the syntax and semantics of these models and concentrate on the complexity of query satisfiability, satisfaction, filtering and entailment (these four problems arise naturally in the intended application domain). Finally we discuss an efficient algorithm for the problem of filtering and evaluate it in a realistic application domain. In previous research we have shown that the problems of satisfaction and filtering can be solved in PTIME but this paper demonstrates that the satisfiability and entailment problems are computationally hard (NP-complete and coNP-complete respectively).

1 Introduction

The selective dissemination of information to interested users is a problem arising frequently in today's information society. This problem has recently received the attention of various research communities including researchers from agent systems [7, 18], databases [12, 1, 21, 10], digital libraries [11, 17], distributed computing [4, 3] and others.

We envision an information dissemination scenario in the context of a *peer-to-peer agent architecture* like the one shown in Figure 1. Users utilize their *end-agents* to post *profiles* or *documents* (expressed in some appropriate language) to some *middle-agents*. End-agents play a dual role: they can be information producers and information consumers at the same time. The P2P network of middle-agents is the "glue" that makes sure that published documents arrive at interested subscribers. To achieve this, middle-agents forward posted profiles to other middle-agents using an appropriate P2P protocol. In this way, matching of a profile with a document can take place at a middle-agent that is as close as possible to the origin of the incoming document. Profile forwarding can be done in a sophisticated way to minimize network traffic e.g., no profiles that are less general than one that has already been processed are actually forwarded.

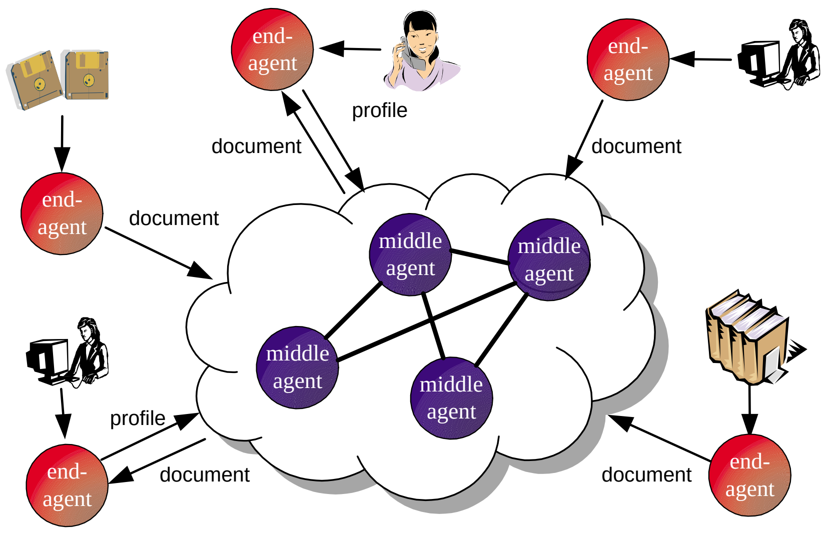

Fig. 1. A P2P agent architecture for information dissemination

In their capacity as information producers, end-agents can also post *advertisements* that describe in a "concise" way the documents that will be produced by them. These advertisements can also be forwarded in the P2P network of middle-agents to *block* the forwarding of *irrelevant* profiles towards a source. Advertisement forwarding can also be done in a sophisticated way using ideas similar to the ones for profile forwarding. The most elegant and complete presentation of these concepts available in the literature can be found in [4] where the *distributed event dissemination system* SIENA is presented. SIENA uses different terminology than the one used here: *event, client* and *server* instead of *document, end-agent* and *middle-agent.*

Our work in this paper concentrates on models and languages for expressing documents and queries/profiles in *textual* information dissemination systems that follow the general architecture of Figure 1.[1] We are motivated by a desire to develop useful P2P agent systems in a *principled* and *formal* way, and make the following technical contributions.

In [15, 16, 18] we have proposed the models $\mathcal{WP}$ and $\mathcal{AWP}$ especially designed for the dissemination of textual information. Data model $\mathcal{WP}$ is based on free text and its query language is based on the *boolean model with proximity operators*. The concepts of $\mathcal{WP}$ extend the traditional concept of proximity in IR [2, 5, 6] in a significant way and utilize it in a content language targeted at

[1] We use the terms *query* and *profile* interchangeably. In an information dissemination setting, a profile is simply a long-standing query. We do not consider advertisements, but it should be clear from our presentation that appropriate subsets of the query languages that we will present could be used for expressing advertisements as well.

information dissemination applications. Data model $\mathcal{AWP}$ is based on *attributes* or *fields* with finite-length strings as values. Its query language is an extension of the query language of data model $\mathcal{WP}$. Our work on $\mathcal{AWP}$ complements other recent proposals for querying textual information in distributed event-based systems [4, 3] by using linguistically motivated concepts such as *word* and not arbitrary strings. This makes $\mathcal{AWP}$ potentially very useful in some applications e.g., alert systems for digital libraries or other commercial systems such as Dialog[2] or Lexis-Nexis[3] where similar models are supported already for retrieval. In [17] we have proposed DIAS, a distributed information alert system for digital libraries that follows the architecture of Figure 1 and makes use of model $\mathcal{AWP}$.

For an information dissemination architecture like the one in Figure 1 to become a reality, several algorithmic problems need to be solved efficiently. The first problem is the *satisfaction problem*: Deciding whether a document satisfies (or matches) a profile. The second problem (which includes the first one) is the *filtering problem*: Given a database of profiles db and a document d, find all profiles $q \in db$ that match d. This functionality is very crucial at each middle-agent because we expect deployed information dissemination systems to handle hundreds of thousands or millions of profiles. In [18] we presented PTIME worst-case upper bounds for the complexity of satisfaction and filtering, and we are currently evaluating *profile indexing algorithms* that allow us to solve the filtering problem efficiently for millions of profiles [9]. These results are currently leading to an implementation of a prototype information dissemination system in the context of project DIET [19, 13].

In this paper, we built on the foundation of [15, 16, 18] and study the computational complexity of query satisfiability and entailment for models $\mathcal{WP}$ and $\mathcal{AWP}$. Our results show that the satisfiability and entailment problems for queries in $\mathcal{WP}$ is NP-complete and coNP-complete respectively. The most important contributions of our work are the proof techniques we develop for the upper bounds based on the model theory of $\mathcal{WP}$ (the lower bounds are easy to see). Our results for $\mathcal{AWP}$ show that even for queries in some “canonical” form satisfiability is NP-complete and query entailment is coNP-complete. The proof techniques utilised here for the derivation of the upper bounds are mappings to Boolean logic. These techniques are of significant interest again because they make available to us all the arena of tools from this area for using them in the solution of practical problems arising in P2P agent systems.

We also concentrate on the problem of filtering and present an efficient algorithm that is able to handle millions of profiles in just a few hundred milliseconds. We outline this algorithm and evaluate it experimentally in a realistic application domain.

The four algorithmic problems we are concerned with arise naturally in agent approaches to information dissemination. [4] were the first to define carefully the notions of satisfaction and entailment (called “covers” in [4]) for the SIENA language of events and profiles, but no complexity analysis has been carried out.

[2] www.dialog.com
[3] www.lexis-nexis.com

[3] presented a similar language for events and profiles with a textual part very close to the model $\mathcal{AWP}$, but only the problems of satisfaction and filtering have been considered. Thus our work is the first to address the computational complexity of query satisfiability and entailment in agent systems.

The rest of the paper is organised as follows. Section 2 presents data model $\mathcal{WP}$ based on free text and its sophisticated query language. Then Section 3 builds on this foundation and develops the same machinery for data model $\mathcal{AWP}$. Sections 4 and 5 present our complexity results for the problem of query satisfiability and entailment. Section 6 outlines a filtering algorithm for textual information dissemination and presents its experimental evaluation. Finally, Section 7 gives our conclusions and discusses useful implications of our work. Most proofs are omitted and will be given in the long version of this paper.

2 The Model $\mathcal{WP}$

Let us start by presenting the data model $\mathcal{WP}$ and its query language. $\mathcal{WP}$ assumes that textual information is in the form of *free text* and can be queried by *word patterns* (hence the acronym for the model). The basic concepts of $\mathcal{WP}$ are subsequently used in Section 3 to define the data model $\mathcal{AWP}$ and its query language.

We assume the existence of a finite *alphabet* $\mathbf{\Sigma}$. A *word* is a finite non-empty sequence of letters from $\mathbf{\Sigma}$. We also assume the existence of a (finite or infinite) set of words called the *vocabulary* and denoted by $\mathcal{V}$.

Definition 1. *A* text value *s of length n over vocabulary $\mathcal{V}$ is a total function $s : \{1, 2, \ldots, n\} \to \mathcal{V}$.*

In other words, a text value s is a finite sequence of words from the assumed vocabulary and $s(i)$ gives the i-th element of s. Text values can be used to represent finite-length strings consisting of words separated by blanks. The length of a text value s (i.e., its number of words) will be denoted by $|s|$.

We now give the definition of word pattern. We assume the existence of a set of *(distance) intervals* $\mathcal{I}$ defined as follows:

$$\mathcal{I} = \{[l, u] : \ l, u \in \mathbb{N}, l \geq 0 \text{ and } l \leq u\} \cup \{[l, \infty) : \ l \in \mathbb{N} \text{ and } l \geq 0\}$$

The symbols $\in$ and $\subseteq$ will be used to denote membership and inclusion in an interval as usual.

Definition 2. *Let $\mathcal{V}$ be a vocabulary. A* word pattern *over vocabulary $\mathcal{V}$ is an expression generated by the following grammar:*

$$PF \to \mathbf{w} \mid \neg PF \mid PF \wedge PF \mid PF \vee PF \mid (PF)$$

$$P \to \mathbf{w_1} \prec_{\mathbf{i_1}} \cdots \prec_{\mathbf{i_{n-1}}} \mathbf{w_n}$$

$$WP \to PF \mid P \mid WP \wedge WP \mid WP \vee WP \mid (WP)$$

The start symbol is WP. Terminals $\mathbf{w}, \mathbf{w_1}, \ldots, \mathbf{w_n}$ represent words of $\mathcal{V}$, and $\mathbf{i_1}, \ldots, \mathbf{i_n}$ represent intervals of $\mathcal{I}$.

Example 1. The following are examples of word patterns:

$$\begin{gathered} \mathit{constraint} \wedge (\mathit{optimisation} \vee \mathit{programming}) \\ \mathit{applications} \wedge (\mathit{constraint} \prec_{[0,0]} \mathit{programming}) \wedge \neg \mathit{e\text{-}commerce}, \\ \mathit{algorithms} \wedge ((\mathit{complexity} \prec_{[1,5]} \mathit{satisfaction}) \vee (\mathit{complexity} \prec_{[1,5]} \mathit{filtering})) \end{gathered}$$

The expressions generated by the first production of the above grammar are called *proximity-free word patterns* while the ones generated by the second production are called *proximity word patterns*. Operators $\prec_i$ are called *proximity operators* and are extensions of the traditional IR operators kW and kN [2, 5, 6]. Proximity operators are used to capture the concepts of *order* and *distance* between words in a text document. The proximity word pattern $w_1 \prec_{[l,u]} w_2$ stands for "word w_1 is *before* w_2 and is separated by w_2 by *at least* l and *at most* u *words*". In the above example $\mathit{complexity} \prec_{[1,5]} \mathit{satisfaction}$ denotes that the word "satisfaction" appears after word "complexity" and at a distance of at least 1 and at most 5 words. The word pattern $\mathit{constraint} \prec_{[0,0]} \mathit{programming}$ denotes that the word "constraint" appears exactly before word "programming" so this is a way to encode the string "constraint programming". We can also have arbitrarily long sequences of proximity operators with similar meaning. Note that proximity-free subformulas in proximity word-patterns cannot be more complex than words. In [15, 16, 18] this restriction is not present as it is the tradition in IR systems [2, 5, 6] but the resulting language is provably equivalent to the one developed here.

We now give semantics to word patterns and define the notions of satisfaction, satisfiability and entailment.

Definition 3. *Let $\mathcal{V}$ be a vocabulary, s a text value over $\mathcal{V}$ and wp a word pattern over $\mathcal{V}$. The concept of s* satisfying *wp (denoted by $s \models wp$) is defined as follows:*

1. *If wp is a word of $\mathcal{V}$ then $s \models wp$ iff there exists $p \in \{1, \ldots, |s|\}$ and $s(p) = wp$.*
2. *If wp is a proximity word pattern of the form $w_1 \prec_{i_1} \cdots \prec_{i_{n-1}} w_n$ then $s \models wp$ iff there exist $p_1, \ldots, p_n \in \{1, \ldots, |s|\}$ such that, for all $j = 2, \ldots, n$ we have $s(p_j) = w_j$ and $p_j - p_{j-1} - 1 \in i_{j-1}$.*
3. *If wp is of the form $\neg wp_1, wp_1 \wedge wp_2, wp_1 \vee wp_2$ or (wp_1) then $s \models wp$ is defined exactly as satisfaction for Boolean logic [20].*

A word pattern wp is called satisfiable *if there is a text value s that satisfies it. Otherwise it is called* unsatisfiable.

Example 2. The word patterns of Example 1 are satisfiable. The following word patterns are unsatisfiable:

$$\mathit{programming} \wedge \neg \mathit{programming}, \ (\mathit{constraint} \prec_{[0,0]} \mathit{programming}) \wedge \neg \mathit{programming}$$

Definition 4. *Let wp_1 and wp_2 be word patterns. We will say that wp_1* entails *wp_2 (denoted by $wp_1 \models wp_2$) iff for every text value s such that $s \models wp_1$, we have $s \models wp_2$. If $wp_1 \models wp_2$ and $wp_2 \models wp_1$ then wp_1 and wp_2 are called* equivalent *(denoted by $wp_1 \equiv wp_2$).*

Example 3. The word pattern *constraint* $\wedge$ *programming* entails *constraint*. The word pattern *optimization* $\wedge$ (*constraint* $\prec_{[0,0]}$ *programming*) entails *constraint* $\prec_{[0,10]}$ *programming*.

The following proposition gives the usual relation between entailment and unsatisfiability of Boolean logic as it should be stated in our framework. Note that wp_2 is required to be proximity-free so that the negation operator can be applied.

Proposition 1. *Let wp_1 and wp_2 be word patterns and wp_2 is proximity-free. $wp_1 \models wp_2$ iff $wp_1 \wedge \neg wp_2$ is unsatisfiable.*

We now define normal forms for word patterns.

Definition 5. *A word pattern is called* atomic *if it is a word, a negated word or a proximity word pattern. A word pattern is called* conjunctive *(resp.* disjunctive*) if it is a conjunction (resp. disjunction) of atomic word patterns. A word pattern is in* conjunctive normal form (CNF) *(resp.* disjunctive normal form (DNF)*) if it is a conjunction (resp. disjunction) of disjunctive (resp. conjunctive) word patterns.*

The following easy proposition is from [15].

Proposition 2. *Every word pattern is equivalent to a word pattern in CNF and a word pattern in DNF.*

A longer presentation of the semantic properties of model $\mathcal{WP}$ and a detailed discussion of related data models in IR, databases and pub/sub systems is given in [15, 16, 18, 17].

3 The Model $\mathcal{AWP}$

Data model $\mathcal{AWP}$ is based on *attributes* or *fields* with finite-length strings as values (in the acronym $\mathcal{AWP}$, the letter $\mathcal{A}$ stands for "attribute"). Strings will be understood as sequences of words as formalised by the model $\mathcal{WP}$ presented earlier. Attributes can be used to encode textual information such as author, title, date, body of text and so on. $\mathcal{AWP}$ is restrictive since it offers a flat view of a text document, but it has wide applicability as we show in [15, 16, 18, 17].

We start our formal development by defining the concepts of document schema and document. Throughout the rest of this paper we assume the existence of a countably infinite set of attributes $\mathbf{U}$ called the *attribute universe*.

Definition 6. *A* document schema *$\mathcal{D}$ is a pair $(\mathcal{A}, \mathcal{V})$ where $\mathcal{A}$ is a subset of the attribute universe $\mathbf{U}$ and $\mathcal{V}$ is a vocabulary.*

Definition 7. *Let $\mathcal{D}$ be a document schema. A* document *d over schema $(\mathcal{A}, \mathcal{V})$ is a set of attribute-value pairs (A, s) where $A \in \mathcal{A}$, s is a text value over $\mathcal{V}$, and there is at most one pair (A, s) for each attribute $A \in \mathcal{A}$.*

Example 4. The following is a document over schema $(\{AUTHOR,\ TITLE,\ ABSTRACT\}, \mathcal{V})$:

$$\begin{gathered}\{\ (AUTHOR, \text{“}\textit{John Brown}\text{”}),\\ (TITLE, \text{“}\textit{Local search and constraint programming}\text{”}),\\ (ABSTRACT, \text{“}\textit{In this paper we show that...}\text{”})\ \}\end{gathered}$$

The syntax of our query language is given by the following recursive definition.

Definition 8. *Let $\mathcal{D} = (\mathcal{A}, \mathcal{V})$ be a document schema. A* query *over $\mathcal{D}$ is a formula in any of the following forms:*

1. *$A \sqsupseteq wp$ where $A \in \mathcal{A}$ and wp is a* positive *word pattern over $\mathcal{V}$. A word pattern is called* positive *if it does not contain negation. The formula $A \sqsupseteq wp$ can be read as “A contains word pattern wp”.*[4]
2. *$A = s$ where $A \in \mathcal{A}$ and s is a text value over $\mathcal{V}$.*
3. *$\neg\phi,\ \phi_1 \vee \phi_2,\ \phi_1 \wedge \phi_2$ where ϕ, ϕ_1 and ϕ_2 are queries.*

Example 5. The following is a query over the schema of Example 4:

$$AUTHOR \sqsupseteq Brown \wedge TITLE \sqsupseteq (search \wedge (constraint \prec_{[0,0]} programming))$$

Let us now define the semantics of the above query language in our dissemination setting. We start by defining when a document satisfies a query.

Definition 9. *Let $\mathcal{D}$ be a document schema, d a document over $\mathcal{D}$ and ϕ a query over $\mathcal{D}$. The concept of document d* satisfying *query ϕ (denoted by $d \models \phi$) is defined as follows:*

1. *If ϕ is of the form $A \sqsupseteq wp$ then $d \models \phi$ iff there exists a pair $(A, s) \in d$ and $s \models wp$.*
2. *If ϕ is of the form $A = s$ then $d \models \phi$ iff there exists a pair $(A, s) \in d$.*
3. *If ϕ is of the form $\neg\phi_1$ then $d \models \phi$ iff $d \not\models \phi_1$.*
4. *If ϕ is of the form $\phi_1 \wedge \phi_2$ then $d \models \phi$ iff $d \models \phi_1$ and $d \models \phi_2$.*
5. *If ϕ is of the form $\phi_1 \vee \phi_2$ then $d \models \phi$ iff $d \models \phi_1$ or $d \models \phi_2$.*

Example 6. The query of Example 5 is satisfied by the document of Example 4.

The concepts of query satisfiability, entailment and equivalence can now be defined as in Section 2.

Definition 10. *A query is called* atomic *if it is in one of the following forms: $A = s, \neg A = s, A \sqsupseteq w, \neg A \sqsupseteq w,\ A \sqsupseteq wp$ or $\neg A \sqsupseteq wp$ where s is a text value, w is a word and wp is a proximity word pattern. A query is called* conjunctive *(resp.* disjunctive*) if it is a conjunction (resp. disjunction) of atomic queries. A query is in* conjunctive normal form (CNF) *(resp.* disjunctive normal form (DNF)*) if it is a conjunction (resp. disjunction) of disjunctive (resp. conjunctive) queries.*

[4] In previous papers [15, 16, 18, 17] we have used the less intuitive symbol $\sqsupset$ for “contains”.

The following proposition is easy to see.

Proposition 3. *Let A be an attribute and wp_1, wp_2 be word patterns. Then the following equivalences hold:*

1. $A \sqsupseteq (wp_1 \wedge wp_2) \equiv (A \sqsupseteq wp_1) \wedge (A \sqsupseteq wp_2)$
2. $A \sqsupseteq (wp_1 \vee wp_2) \equiv (A \sqsupseteq wp_1) \vee (A \sqsupseteq wp_2)$
3. $\neg(A \sqsupseteq (wp_1 \wedge wp_2)) \equiv (\neg A \sqsupseteq wp_1) \vee (\neg A \sqsupseteq wp_2)$
4. $\neg(A \sqsupseteq (wp_1 \vee wp_2)) \equiv (\neg A \sqsupseteq wp_1) \wedge (\neg A \sqsupseteq wp_2)$

From Proposition 2 and 3, we now have the following result of [15] which closes this section.

Proposition 4. *Every query is equivalent to a query in DNF and a query in CNF.*

4 Satisfiability and Entailment in $\mathcal{WP}$

We now turn our attention to the satisfiability and entailment problems for queries in $\mathcal{WP}$. Let the satisfiability problem for proximity-free word patterns be denoted by PFWP-SAT. There is an obvious connection of PFWP-SAT and SAT, the satisfiability problem for Boolean logic [20]. Any instance of PFWP-SAT can be considered to be an instance of SAT and vice versa (this is a trivial reduction where the roles of words and Boolean variables are interchanged). Thus we only have to consider the complications arising in our framework due to proximity word patterns.

In what follows, we will need the binary operation of *concatenation* of two text values.

Definition 11. *Let s_1 and s_2 be text values over vocabulary $\mathcal{V}$. Then the* concatenation *of s_1 and s_2 is a new text value denoted by $s_1 s_2$ and defined by the following:*

1. $|s_1 s_2| = |s_1| + |s_2|$
2. $s_1 s_2(x) = s_1(x)$ *for all* $x \in \{1, \ldots, |s_1|\}$, *and*
3. $s_1 s_2(x) = s_2(x)$ *for all* $x \in \{|s_1| + 1, \ldots, |s_2| + |s_1|\}$

We will also need the concept of the *empty text value* which is denoted by ϵ and has the property $|\epsilon| = 0$. The following properties of concatenation are easily seen:

1. $(s_1 s_2)s_3 = s_1(s_2 s_3)$, for all text values s_1, s_2 and s_3.
2. $s\epsilon = \epsilon s = s$ for every text value s.

The associativity of concatenation allows us to write concatenations of more than two text values without using parentheses.

The following variant of the concept of satisfaction captures the notion of a set of positions in a text value containing *only* words that contribute to the satisfaction of a proximity-free word pattern. This concept is used in the results that follow.

Definition 12. *Let $\mathcal{V}$ be a vocabulary, s a text value over $\mathcal{V}$, wp a proximity-free word pattern over $\mathcal{V}$, and P a subset of $\{1, \ldots, |s|\}$. The concept of s* satisfying *wp* with set of positions *P (denoted by $s \models_P wp$) is defined as follows:*

1. *If wp is a word of $\mathcal{V}$ then $s \models_P wp$ iff there exists $x \in \{1, \ldots, |s|\}$ such that $P = \{x\}$ and $s(x) = wp$.*
2. *If wp is of the form $wp_1 \wedge wp_2$ then $s \models_P wp$ iff there exist sets of positions $P_1, P_2 \subseteq \{1, \ldots, |s|\}$ such that $s \models_{P_1} wp_1$, $s \models_{P_2} wp_2$ and $P = P_1 \cup P_2$.*
3. *If wp is of the form $wp_1 \vee wp_2$ then $s \models_P wp$ iff $s \models_P wp_1$ or $s \models_P wp_2$.*
4. *If wp is of the form (wp_1) then $s \models_P wp$ iff $s \models_P wp_1$.*

We also need the following notation. Let P be a subset of the set of natural numbers $\mathbb{N}$, and $x \in \mathbb{N}$. We will use the notation $P + x$ to denote the set of natural numbers $\{p + x : \ p \in P\}$.

The following lemma is now easy to see.

Lemma 1. *Let s and s' be text values, wp a proximity-free word pattern and $P \subseteq \{1, \ldots, |s|\}$. If $s \models_P wp$ then $ss' \models_P wp$ and $s's \models_{P+|s'|} wp$.*

The following proposition shows that positive proximity-free word patterns are always satisfiable (its proof can be done by induction on the structure of the word pattern).

Proposition 5. *If wp is a positive proximity-free word pattern then wp is satisfiable. In fact, there exists a text value s_0 such that*

1. *$|s_0| \leq words(wp) \cdot ops(wp)$ where $words(wp)$ is the number of words of wp (multiple occurrences of the same word are multiply counted) and $ops(wp)$ is the number of operators of wp (or 1 if wp has no operators).*
2. *Every word of s_0 is a word of wp.*
3. *$s_0 \models_{\{1,\ldots,|s_0|\}} wp$.*

We can easily show that proximity word patterns are also always satisfiable.

Proposition 6. *Let wp be a proximity word-pattern of the form*

$$w_1 \prec_{i_1} \cdots \prec_{i_{n-1}} w_n.$$

Then wp is satisfied by a text value $s = w_1 v_1 \cdots v_{n-1} w_n$ where $v_k,\ k = 1, \ldots, n-1$ are text values of the following form: If $begin(i_k) > 0$ then v_k is formed by $begin(i_k)$ successive occurrences of the special word # which is not contained in wp. Otherwise, v_k is the empty text value ϵ.

Finally, we can show that any positive word pattern is always satisfiable.

Proposition 7. *Let wp be a positive word pattern and $\theta_1 \vee \cdots \vee \theta_k$ be the DNF of wp. Then there exists a $j \in \{1, \ldots, k\}$, and text values $s_{0j}, s_{1j}, \ldots, s_{mj}$ such that $s_{0j} s_{1j} \cdots s_{mj} \models wp$ and*

1. *s_{0j} is a sequence of words appearing as conjuncts of disjunct θ_j, and*

2. for $i = 1, \ldots, m$, s_{ij} is a text value such that $s_{ij} \models_{\{1,\ldots,|s_{ij}|\}} \phi_i$ where $\phi_1, \ldots, \phi_m$ are all the proximity conjuncts of θ_j.

The next theorem shows that when negation is introduced, deciding the satisfiability of a word pattern becomes a hard computational problem. But first we need a lemma that shows that even in the case of arbitrary word patterns, satisfiability implies satisfaction by a text value of a special form.

Lemma 2. *Let wp be a word pattern and $\theta_1 \vee \cdots \vee \theta_k$ be the DNF of wp. Then wp is satisfiable iff there exists $j \in \{1, \ldots, k\}$, and text values $s_{0j}, s_{1j}, \ldots, s_{mj}$ such that $s_{0j}s_{1j} \cdots s_{mj} \models wp$ and*

1. *s_{0j} is a sequence of words appearing non-negated in disjunct θ_j, and*
2. *for $i = 1, \ldots, m$, s_{ij} is a text value such that $s_{ij} \models_{\{1,\ldots,|s_{ij}|\}} \phi_i$ where $\phi_1, \ldots, \phi_m$ are all the proximity conjuncts of θ_j.*

We can now utilize Lemma 2 to show the upper bound in the following result (the lower bound is easy).

Theorem 1. *Let wp be a word pattern. Deciding whether wp is satisfiable is an NP-complete problem.*

A tractable case of the satisfiability problem for word patterns in $\mathcal{WP}$ is given by the following easy theorem.

Theorem 2. *Let wp be a word pattern in DNF. Deciding whether wp is satisfiable can be done in $O(\kappa^2\zeta)$ time where κ is the maximum number of words in a conjunct of wp, and ζ is the number of conjuncts.*

Let us now turn our attention to the entailment problem in $\mathcal{WP}$. The problem is easily seen to be coNP-hard from Proposition 1. Using techniques similar to the ones developed above we can show that to decide whether $\phi \models \psi$ for word patterns ϕ and ψ it is enough to consider text values of a special form. This allows to prove that the entailment problem is in coNP thus we have the following result.

Theorem 3. *Deciding whether a word pattern in $\mathcal{WP}$ entails another is a coNP-complete problem.*

5 Satisfiability and Entailment in $\mathcal{AWP}$

Let us now turn our attention to the satisfiability and entailment problems for queries in $\mathcal{AWP}$. Let $\mathcal{Q}$ denote the class of queries in $\mathcal{AWP}$ with the following property: every positive word pattern *wp* appearing in formulas of the form $A \sqsupseteq wp$ is in DNF or CNF form. Let SAT($\mathcal{Q}$) denote the satisfiability problem for queries in class $\mathcal{Q}$. The following two propositions show that the problems SAT and SAT($\mathcal{Q}$) are equivalent under polynomial time reductions.

Proposition 8. SAT is polynomially reducible to SAT($\mathcal{Q}$).

Proposition 9. SAT($\mathcal{Q}$) is polynomially reducible to SAT.

Proof. Let ϕ be a query in $\mathcal{Q}$. Using Proposition 3, ϕ can easily be transformed into a formula θ which is a Boolean combination of atomic queries (see Definition 10) using only operators $\wedge$ and $\vee$. This transformation can be done in time linear in the size of the formula.

The next step is to substitute in θ atomic formulas $A = s$ and $A \sqsupseteq wp$ (where wp is a word or a proximity word pattern) by propositional variables $p_{A=s}$ and $p_{A \sqsupseteq wp}$ respectively to obtain formula θ'. Finally, the following formulas are conjoined to θ' to obtain ψ:

1. If $A = s_1$ and $A = s_2$ are conjuncts of θ' and $s_1 \neq s_2$ then conjoin $p_{A=s_1} \equiv \neg p_{A=s_2}$.
2. If $A = s$ and $A \sqsupseteq wp$ are conjuncts of θ' and $s \models wp$ then conjoin $p_{A=s} \supset p_{A \sqsupseteq wp}$.
3. If $A = s$ and $A \sqsupseteq wp$ are conjuncts of θ' and $s \not\models wp$ then conjoin $p_{A=s} \supset \neg p_{A \sqsupseteq wp}$.
4. If $A \sqsupseteq wp_1$ and $A \sqsupseteq wp_2$ are conjuncts of θ' and $wp_1 \models wp_2$ then conjoin $p_{A \sqsupseteq wp_1} \supset p_{A \sqsupseteq wp_2}$.

The above step can be done in polynomial time because satisfaction and entailment of word patterns in θ' can be done in polynomial time.

It is also easy to see that ϕ is a satisfiable query iff ψ is a satisfiable formula of Boolean logic. Then the result holds. ■

Thus we have the following result.

Theorem 4. *The problem SAT($\mathcal{Q}$) is NP-complete.*

Given the reduction of Proposition 9, one can discover tractable subcases of the problem SAT($\mathcal{Q}$). As an example, an easy reduction from 2-SAT gives us the following result.

Corollary 1. *Let ϕ be a query of $\mathcal{AWP}$ such that each disjunction of ϕ has at most two disjuncts. The problem of deciding whether ϕ is satisfiable can be solved in PTIME.*

Finally, what is important about Proposition 9 is that it gives us a straightforward way of evaluating the satisfiability of a query ϕ by transforming it into a propositional formula ψ and invoking a well-known propositional satisfiability algorithm on ψ (e.g., variations of GSAT [22]). One could also devise filtering algorithms that are based on Boolean logic techniques as it is done for a similar language in [3].

The following theorem can now be proved using similar ideas.

Theorem 5. *The entailment problem for queries of $\mathcal{AWP}$ in class $\mathcal{Q}$ is coNP-complete.*

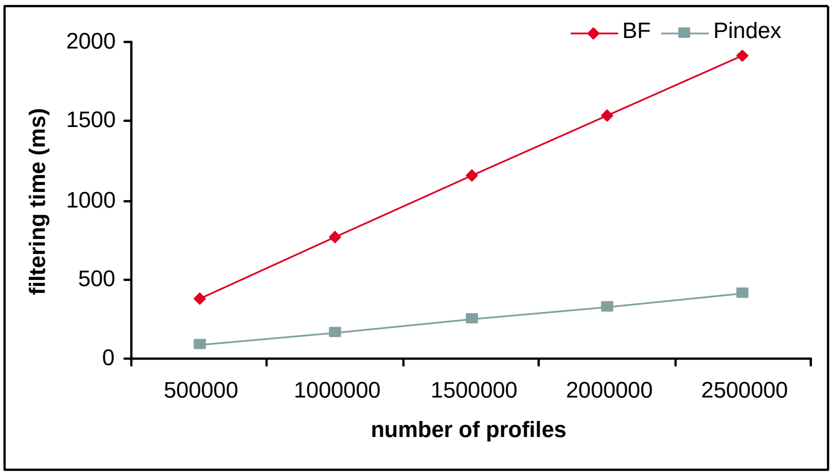

Fig. 2. Matching time vs number of indexed profiles for queries under $\mathcal{AWP}$.

6 Filtering algorithms for $\mathcal{AWP}$

In this section we briefly describe two main memory algorithms suitable for the filtering problem in P2P agent systems for textual information dissemination. The algorithms presented are designed for the following class of queries. A query can be of the form $A_1 = s_1 \ \wedge \ ... \ \wedge \ A_n = s_n \ \wedge \ B_1 \sqsupseteq wp_1 \ \wedge \ ... \ \wedge \ B_m \sqsupseteq wp_m$, where A_i, B_i are attributes, s_i is a text value and wp_i is a word pattern containing conjunctions of words and proximity formulas with only words as subformulas.

The Brute Force algorithm (BF) is a very simple one and was implemented for comparison purposes. BF maintains a linked list where all the profiles are stored and each time a new profile arrives it is inserted at the end of this list. When an incoming document arrives, BF sequentially scans all the profiles to find those that match it.

The Profile Index algorithm (PIndex) utilises a two level index over profiles under the model $\mathcal{AWP}$. Each profile of the form $A_1 = s_1 \wedge ... \wedge A_n = s_n \wedge B_1 \sqsupseteq wp_1 \ \wedge \ ... \ \wedge \ B_m \sqsupseteq wp_m$ is indexed under all its attributes $A_1, ..., A_n, B_1, ..., B_m$ and m words selected randomly from $wp_1, ..., wp_m$. To match an incoming document against a set of profiles, PIndex utilises the two level index to retrieve quickly all matching profiles. To facilitate the matching process several data structures are used; some of them for indexing the profiles, some others for representing the incoming document, and there are also auxiliary data structures that are used in order to improve performance. The details of this algorithm are omitted for space reasons.

The experiments were carried out using documents downloaded from *ResearchIndex*[5] and realistic profiles, consisting of terms extracted from the documents. The same documents were previously used in [8]. Both the profiles and the documents are stored in main memory and the time measured is the mean matching time for one hundred incoming documents. The experiments were run on a standard PC with Pentium III 1.6GHz processor and 1GB RAM, running Linux. As it is shown in Figure 2, PIndex deals with 2.5 million profiles in less than 500 milliseconds.

7 Conclusions

We discussed the models $\mathcal{WP}$ and $\mathcal{AWP}$ especially designed for the selective dissemination of textual information by peer-to-peer agent systems. We briefly presented the syntax and semantics of these models and concentrated on the complexity of query satisfiability, satisfaction, filtering and entailment (these four problems arise naturally in the intended application domain). We also outlined two algorithms for the problem of filtering and presented some preliminary performance evaluation results based on corpus documents and realistic profiles. We are currently working on more sophisticated filtering algorithms which exploit similarities among profiles. We are currently also implementing a complete version of our P2P agent architecture on top of the DIET core software presented in [14].

Acknowledgements

This work was carried out as part of the DIET (Decentralised Information Ecosystems Technologies) project (IST-1999-10088), within the Universal Information Ecosystems initiative of the Information Society Technology Programme of the European Union. We would like to thank the other participants in the DIET project, from Departmento de Teoria de Senal y Comunicaciones, Universidad Carlos III de Madrid, the Intelligent Systems Laboratory, BTexact Technologies and the Intelligent and Simulation Systems Department, DFKI, for their comments and contributions.

We would also like to thank Evangellos Milios and his group at Dalhousie University for providing us the original Neural Network Corpus.

Special thanks go to Thodoros Koutris and Paraskevi Raftopoulou for their help in the processing of the corpus and for reading previous versions of this paper and providing useful comments on it.

References

1. M. Altinel and M.J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *Proceedings of the 26th VLDB Conference*, 2000.

[5] www.researchindex.org

2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, 1999.
3. A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith. Efficent filtering in publish-subscribe systems using binary decision diagrams. In *Proceedings of the 23rd International Conference on Software Engineering*, Toronto, Ontario, Canada, 2001.
4. A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing (PODC'2000)*, pages 219–227, 2000.
5. C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. Boolean Query Mapping across Heterogeneous Information Sources. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):515–521, 1996.
6. C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System. *ACM Transactions on Information Systems*, 17(1):1–39, 1999.
7. K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of IJCAI-97*, 1997.
8. L. Dong. Automatic term extraction and similarity assessment in a domain specific document corpus. Master's thesis, Department of Computer Science, Dalhousie University, Halifax, Canada, 2002.
9. M. Koubarakis et. al. Project DIET Deliverable 7 (Information Brokering), December 2001.
10. F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *Proceedings of ACM SIGMOD-2001*, 2001.
11. D. Faensen, L. Faulstich, H. Schweppe, A. Hinze, and A. Steidinger. Hermes – A Notification Service for Digital Libraries. In *Proceedings of the Joint ACM/IEEE Conference on Digital Libraries (JCDL'01), Roanoke, Virginia, USA*, 2001.
12. M. J. Franklin and S. B. Zdonik. "Data In Your Face": Push Technology in Perspective. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 516–519, 1998.
13. A. Galardo-Antolin, A. Navia-Vasquez, H.Y. Molina-Bulla, A.B. Rodriquez-Gonzalez, F.J. Valvarde-Albacete, A.R. Figueiras-Vidal, T. Koutris, A. Xiruhaki, and M. Koubarakis. I-Gaia: an Information Processing Layer for the DIET Platform. In *Proceedings of the 1st International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS 2002)*, September 15–19 2002.
14. C. Hoile, F. Wang, E. Bonsma, and P. Marrow. Core specification and experiments in DIET: a decentralised ecosystem-inspired mobile agent system. In *Proceedings of the 1st International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS 2002)*, September 15–19 2002.
15. M. Koubarakis. Boolean Queries with Proximity Operators for Information Dissemination. Proceedings of the workshop on Foundations of Models and Languages for Information Integration (FMII-2001), Viterbo, Italy , 16-18 September, 2001. In LNCS (forthcoming).
16. M. Koubarakis. Textual Information Dissemination in Distributed Event-Based Systems. Proceedings of the International Workshop on Distributed Event-Based systems (DEBS'02), July 2-3, 2002, Vienna, Austria.
17. M. Koubarakis, T. Koutris, C. Tryfonopoulos, and P. Raftopoulou. Information Alert in Distributed Digital Libraries: The Models, Languages and Architecture

of DIAS. In *Proceedings of the 6th European Conference on Digital Libraries (ECDL2002)*, September 2002. Forthcoming.

18. M. Koubarakis, C. Tryfonopoulos, P. Raftopoulou, and T. Koutris. Data models and languages for agent-based textual information dissemination. In *Proceedings of the 6th International Workshop on Cooperative Information Agents (CIA'2002)*, September 2002. Forthcoming.
19. P. Marrow, M. Koubarakis, R.H. van Lengen, F. Valverde-Albacete, E. Bonsma, J. Cid-Suerio, A.R. Figueiras-Vidal, A. Gallardo-Antolin, C. Hoile, T. Koutris, H. Molina-Bulla, A. Navia-Vazquez, P. Raftopoulou, N. Skarmeas, C. Tryfonopoulos, F. Wang, and C. Xiruhaki. Agents in Decentralised Information Ecosystems: The DIET Approach. In M. Schroeder and K. Stathis, editors, *Proceedings of the AISB'01 Symposium on Information Agents for Electronic Commerce, AISB'01 Convention*, pages 109–117, University of York, United Kingdom, March 2001.
20. C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
21. J. Pereira, F. Fabret, F. Llirbat, and D. Shasha. Efficient matching for web-based publish/subscribe systems. In *Proceedings of COOPIS-2000*, 2000.
22. B. Selman, H. Levesque, and D. Mitchell. GSAT: A new method for solving hard satisfiability problems. In *Proceedings AAAI-92*, pages 440–446, San Jose, CA, USA, 1992.