A NOTE ON SPERNER'S LEMMA AND ROBUST MACHINES

P. Crescenzi and R. Silvestri

Abstract. Sperner's Lemma states that any admissible coloring of any triangulation of the unit triangle has a 3-colored triangle. In this paper, we first show that any algorithm to find this 3-colored triangle that treats the coloring itself as an oracle must be in the worst case linear in the size of the triangulation. Successively, we apply this lower bound to solve three open questions on robust machines posed by Hartmanis and Hemachandra.

Subject classifications. 68Q05, 68Q10, 68Q25, 68R05.

1. Introduction

It has been recently pointed out that in several well-known instances in mathematics the existence of a mathematical object is established by an inefficient constructive argument (see Megiddo & Papadimitriou 1991, Papadimitriou 1990). These include Brouwer's Fixed Point Theorem, Sperner's Lemma, Chevalley's Theorem, and Smith's Theorem. For example, Brouwer's Fixed Point Theorem states that any continuous function f from the d-dimensional simplex to itself has a fixed point (see Brouwer 1912). Computing such a fixed point, however, is difficult: indeed, it was shown that any algorithm to find an approximate fixed point accurate to p binary digits that treats f as an oracle must be in the worst case exponential in p (see Hirsh & Vavasis 1987, Hirsh *et al.* 1989). Papadimitriou (1990) and Papadimitriou (1994) captured this inefficient-existence-proof phenomenon by complexity classes containing several important complete problems.

Sperner's Lemma (see Sperner 1928, Shashkin 1991) states that any admissible coloring of any triangulation of the unit triangle has a 3-colored triangle (see Figure 3.3). The proof of such a lemma is constructive, albeit by an algorithm that takes $O(n^2)$ steps, where n denotes the size of the triangulation. In this paper we show that any algorithm to find a 3-colored triangle in any admissible coloring that treats the coloring itself as an oracle must be in the worst case linear in n. Such a result is per se maybe not surprising: indeed, the proof is based on techniques similar to those presented by Hirsh & Vavasis (1987). More appealing, instead, is the fact *the lower bound on the complexity of Sperner's Lemma can be usefully applied to study robust machines.*

A robust property of a machine is a property that a machine has with every oracle.[1] For example, if two machines N_0 and N_1 span the entire Σ^* for every oracle (that is, $(\forall A)[L(N_0^A) \cup L(N_1^A) = \Sigma^*]$), we say that the two machines are *robustly* Σ^**-spanning*. Robust machines have been deeply investigated in recent years. For instance, Beigel (August 1988), Ko (1987) and Schöning (1985) used robust machines for discriminating between the oracles that 'help' to solve a problem and those that do not, while robustness is interpreted by Hemachandra (1993) as a radical approach to fault-tolerant database access. Hartmanis & Hemachandra (1990) ask what price a machine pays to have robust properties. In particular, they prove the following three theorems:

1. Machines robustly Σ^*-accepting accept for transparent reasons (a machine is *robustly* Σ^**-accepting* if for every oracle accepts all inputs).

   Theorem 1.1. *Let N be a robustly Σ^*-accepting oracle Turing machine. Then, for all sparse oracles S, a function f computable in $\mathrm{P}^{\mathrm{NP}\oplus S}$ exists such that, for all x, $f(x)$ prints an accepting computation path of $N^S(x)$.*

2. Machines robustly Σ^*-spanning have simple selector functions.

   Theorem 1.2. *Let N_0 and N_1 be two robustly Σ^*-spanning oracle Turing machines. Then, for all sparse oracles S, a function f computable in $\mathrm{P}^{\mathrm{NP}\oplus S}$ exists such that, for all x, $x \in L(N^S_{f(x)})$.*

3. Machines robustly complementary and categorical accept easy languages (two machines are *robustly complementary* if they accept complementary

[1]In this paper, all robust machines are assumed to be *nondeterministic polynomial-time oracle Turing machines.*

languages for every oracle, while a machine is *robustly categorical* if for no oracle and no input it has more than one accepting path).

THEOREM 1.3. *Let N_0 and N_1 be two robustly complementary and categorical oracle Turing machines. Then, for all sparse oracles S, $L(N_0^S) \in$ $\mathrm{P}^{(\mathrm{UP} \cup co\mathrm{UP}) \oplus S}$.*

Hartmanis & Hemachandra (1990) leave as an open problem the question whether the above theorems can be proven with the sparseness condition removed.

In this paper we solve this problem. In particular, by making use of the lower bound on Sperner's Lemma, we show that the answer is negative in the case of Theorems 1.1 and 1.2, while, by making use of techniques similar to those used by Hartmanis & Hemachandra (1990), we prove that the answer is affirmative in the case of the third theorem.

A preliminary version of this paper was presented at *Structure in Complexity Theory, Eighth Annual IEEE Conference*, 1993.

1.1. Notations. Throughout this paper, we will use standard notations in complexity theory (see, for example, Balcázar *et al.* 1988).

A *query-algorithm* is an algorithm which is based on queries about the bits of its input. Such an algorithm starts with a bit-position i_0 (which depends on the length n of the input) and asks for its value b_0. Based on n, i_0, and b_0, it computes a new bit-position i_1, asks for its value b_1, and continues until it returns an output. We permit a query-algorithm unlimited computational power in terms of operating on the queries i_j and their values b_j. The limit on the computational power of the algorithm comes from the amount of information it has about the input, that is, the *number of queries* made before it halts. A query-algorithm is said to be *$f(n)$-bounded* if, for every input of length n, it halts after at most $f(n)$ queries.

2. Robustly Σ^*-spanning machines

In this section we shall prove modulo the existence of suitable functions that Theorem 1.2 cannot be proven with the sparseness condition removed.

Let $\Sigma = \{0, 1\}$. A total function $f : \Sigma^* \to \Sigma$ is said to have *inefficient short witnesses* if two polynomial-time decidable predicates w_0 and w_1 and three constants h, k, and c exist such that:

1. For any x of length n, if $f(x) = 1$ then a *1-witness for* x exists, that is, a k-tuple of bit-positions $i_1, \ldots, i_k$ exists such that $w_1(n, i_1, x_{i_1}, \ldots, i_k, x_{i_k})$ is true.

2. For any x of length n, if $f(x) = 0$ then a *0-witness for* x exists, that is, a h-tuple of bit-positions $j_1, \ldots, j_h$ exists such that $w_0(n, j_1, x_{j_1}, \ldots, j_h, x_{j_h})$ is true.

3. For any $c\sqrt{n}$-bounded query algorithm and for any n, a word x of length n exists such that the algorithm on input x either outputs 1 and no 1-witness for x exists or outputs 0 and no 0-witness for x exists. Note that this condition intuitively justifies the term 'inefficient': indeed, it states that many bits have to be known in order to find a witness.

(Actually, the above conditions are stronger than necessary in order to prove the next theorem: we preferred to trade optimality for clarity.)

By assuming the existence of functions with inefficient short witnesses, which will be proved in the next section, we are now in a position to prove the main result of this section.

THEOREM 2.1. *Two robustly Σ^*-spanning machines N_0 and N_1 and an oracle E exist such that, for any (0,1)-function f computable in* $\mathrm{P}^{\mathrm{NP}\oplus E}$, *a word x exists such that:*

$$x \notin L(N^E_{f(x)}).$$

PROOF. Let f be a function with inefficient short witnesses and let w_0, w_1 and h, k, c be the corresponding two polynomial-time decidable predicates and the corresponding three constants, respectively. Note that, for any oracle A and for any $n \geq 0$, the characteristic function of $A \cap \Sigma^n$ can be viewed as an input of length 2^n for f which will be denoted as σ^A_n.

The two machines N_0 and N_1 are defined as follows:

1. On input x and for any oracle A, each computation path of $N^A_1(x)$ guesses a k-tuple $i_1, \ldots, i_k$ of bit-positions of $\sigma^A_{|x|}$, queries the oracle A about the values $b_1, \ldots, b_k$ of these bit-positions and tests whether $w_1(2^{|x|}, i_1, b_1, \ldots, i_k, b_k)$ is true. If so, then it accepts, otherwise it rejects.

2. On input x and for any oracle A, each computation path of $N^A_0(x)$ guesses a h-tuple $j_1, \ldots, j_h$ of bit-positions of $\sigma^A_{|x|}$, queries the oracle A about the values $b_1, \ldots, b_h$ of these bit-positions and tests whether $w_0(2^{|x|}, j_1, b_1, \ldots, j_h, b_h)$ is true. If so, then it accepts, otherwise it rejects.

From the first two properties of f it follows that

$$(\forall A)[L(N_0^A) \cup L(N_1^A) = \Sigma^*].$$

The oracle E will be derived by diagonalization. Let $T_1, T_2, \ldots$ be an enumeration of deterministic oracle Turing machines so that the running time of T_i is bounded by the polynomial $p_i(n) = n^i + i$. The diagonalization process consists in associating with each T_i in the enumeration an integer n_i such that either $T_i^{\mathrm{NP}\oplus E}(0^{n_i}) = 1$ and $\sigma_{n_i}^E$ has no 1-witness or $T_i^{\mathrm{NP}\oplus E}(0^{n_i}) = 0$ and $\sigma_{n_i}^E$ has no 0-witness. This, in turn, implies that no polynomial-time deterministic Turing machine with oracle $\mathrm{NP} \oplus E$ exists which is able to select the right machine between N_0 and N_1.

The oracle E is constructed in stages. Let $E(i)$ denote the finite set of words included in E after the ith stage and let n_i be an upper bound on the length of the words of $E(i)$. At the beginning we set $E(0) = \emptyset$ and $n_0 = 0$. The set $E(i)$ is then computed as follows:

> Let $n_i = \min\{m : p_{i-1}(n_{i-1}) < m \wedge p_i(m) < c\sqrt{2^m}\}$. Let $H(i)$ be a set of words of length n_i such that either $T_i^{\mathrm{NP}\oplus(E(i-1)\cup H(i))}(0^{n_i}) = 1$ and $\sigma_{n_i}^{H(i)}$ has no 1-witness or $T_i^{\mathrm{NP}\oplus(E(i-1)\cup H(i))}(0^{n_i}) = 0$ and $\sigma_{n_i}^{H(i)}$ has no 0-witness. We then set $E(i)$ equal to $E(i-1) \cup H(i)$.

Since $p_i(|0^{n_i}|) < c\sqrt{2^{n_i}}$, the number of queries made by the computation $T_i^{\mathrm{NP}\oplus(E(i-1)\cup H(i))}(0^{n_i})$ is less than $c\sqrt{2^{n_i}}$. From the third property of f, it then follows that a set $H(i)$ always exists. □

Note that, by similar techniques, it is possible to prove the following stronger version of the above theorem.

THEOREM 2.2. *Two robustly Σ^*-spanning machines N_0 and N_1 exist such that, for any oracle H, an oracle E exists such that, for any (0,1)-function f computable in* $(\mathrm{NP} \cap co\mathrm{NP})^{H\oplus E}$, *a word x exists such that:*

$$x \notin L(N_{f(x)}^E).$$

Note also that from Theorem 2.1 it follows that Theorem 1.1 cannot be proven with the sparseness condition removed. Moreover, in this case it is possible to prove that sparseness is optimal, that is, for any superpolynomial density, Theorem 1.1 fails for some oracle of that density. We leave as an open problem whether sparseness is optimal also in the case of Theorem 1.2.

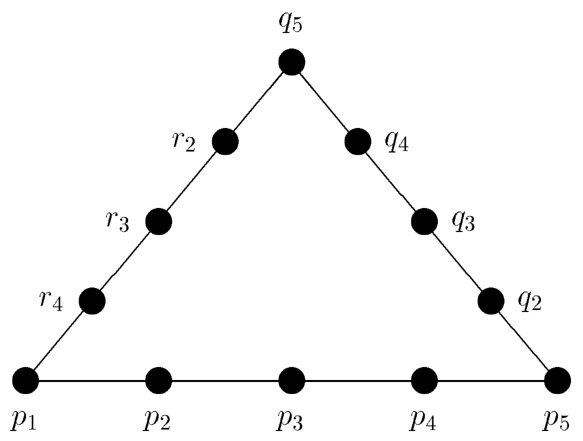

Figure 3.1: Sperner's triangle of size 5

3. The complexity of Sperner's Lemma

In this section we prove that functions with inefficient short witnesses exist by making use of the Sperner's Lemma. Before going on, let us remark that the difficulty in finding these functions is clearly due to the third condition in their definition, that is, the inefficiency of the witnesses. Indeed, the main result of this section (see Lemma 3.2) faces this problem.

The *Sperner's triangle of size* n is a cycle of $3n-3$ *border vertices* $p_1, \ldots, p_n$, $q_2, \ldots, q_n$, and $r_2, \ldots, r_{n-1}$ (see Figure 3.1).

The *standard triangulation* (of the Sperner's triangle) of size n is obtained by joining the following pairs of vertices (see Figure 3.2):

1. p_i and $q_{n-(i-1)}$, for any i with $2 \leq i \leq n-1$,

2. p_i and $r_{n-(i-1)}$, for any i with $2 \leq i \leq n-1$, and

3. q_i and $r_{n-(i-1)}$, for any i with $2 \leq i \leq n-1$.

The crossings between the above straight-line segments are called *crossing vertices*.

Let N denote the set of border and crossing vertices of the standard triangulation of size n. A vertex coloring $c : N \to \{0,1,2\}$ is said to be *admissible* if:

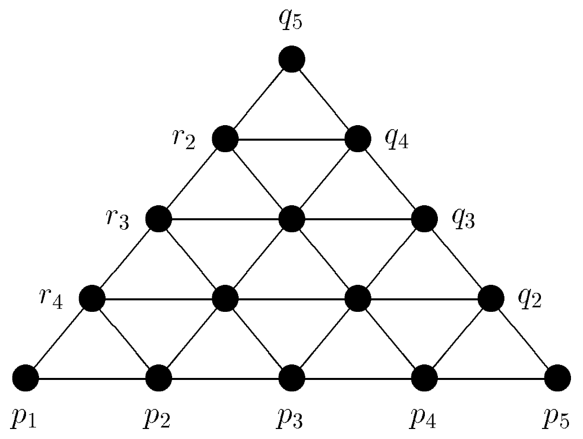

Figure 3.2: Standard triangulation of size 5

1. $c(p_1) = 0$, $c(p_n) = 1$, and $c(q_n) = 2$, and
2. $c(p_i) \neq 2$, for any i with $2 \leq i \leq n-1$,
3. $c(q_i) \neq 0$, for any i with $2 \leq i \leq n-1$, and
4. $c(r_i) \neq 1$, for any i with $2 \leq i \leq n-1$ (see Figure 3.3).

The Sperner's Lemma allows to state the following result.

Lemma 3.1. *Any admissible coloring of the standard triangulation of size n has a 3-colored triangle.*

(A proof of this lemma is given, for instance, by Papadimitriou 1990.)

On the ground of this lemma, we can define a function f which on input a direct encoding of a coloring of the standard triangulation of size n outputs 1 if and only if this coloring has a 3-colored triangle. We shall now prove that f has inefficient short witnesses.

Clearly, f admits 1-witnesses (that is, a 3-colored triangle). Moreover, Lemma 3.1 states that f also admits 0-witnesses (that is, a single border vertex which is colored in a non-admissible way). It remains to show that these witnesses are inefficient. In the following, we denote by m_n the length of a direct encoding of a coloring of the standard triangulation of size n.

Lemma 3.2. *For any $\sqrt{m_n}/3$-bounded query algorithm and for any n, a coloring of the standard triangulation of size n exists such that the algorithm on input this coloring either outputs 1 and no 3-colored triangle exists or outputs 0 and the coloring is admissible.*

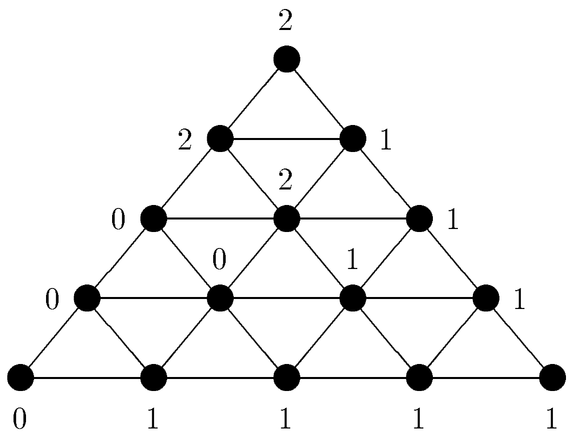

Figure 3.3: An admissible coloring of the standard triangulation of size 5

Proof. The coloring requiring the stated number of queries will be derived by an adversary argument similar to that used by Hirsh & Vavasis (1987). We construct a sequence $P_0, P_1, \ldots, P_t$ of partial paths starting from the border vertex $r_{\lceil n/2 \rceil - 1}$ to fool the algorithm. Each P_i is an extension of P_{i-1} and is a sequence of alternating right- and down-straight subpaths: it is constructed from P_{i-1} either by setting $P_i = P_{i-1}$ or by the addition of one more subpath. The P_i will be chosen to force the algorithm to make a lot of queries.

Each P_i stops at a vertex and tentatively begins a new subpath S_i: S_i is the path from the last vertex l_i of P_i either horizontal-rightward or down-right to a border vertex b_i. In both cases, we denote as R_i the parallelogram determined by l_i, b_i, and p_n (see Figure 3.4).

At the beginning, P_0 includes only vertex $r_{\lceil n/2 \rceil - 1}$ and S_0 is the horizontal-rightward path from $r_{\lceil n/2 \rceil - 1}$ to the q-border.

Let v_i be the vertex whose color c_i has been asked by the algorithm at the ith query. We then distinguish the following cases:

1. v_i is a border vertex. In this case we answer according to the following rules (see Figure 3.3):

   (a) $c(p_1) = c(r_i) = 0$, for any i with $\lceil n/2 \rceil \leq i \leq n-1$,

   (b) $c(p_i) = c(q_j) = 1$, for any i with $2 \leq i \leq n$ and for any j with $2 \leq j \leq n-1$,

   (c) $c(q_n) = c(r_i) = 2$, for any i with $2 \leq i \leq \lceil n/2 \rceil - 1$.

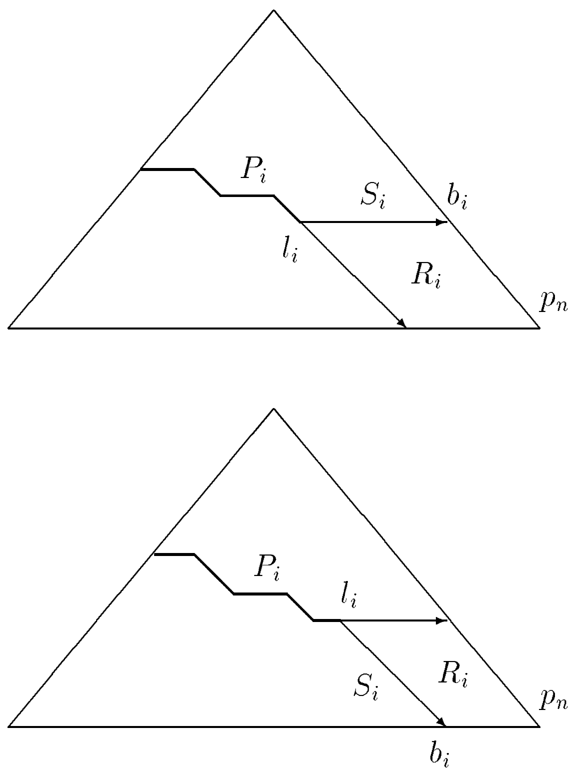

Figure 3.4: Partial paths and tentative extensions

Furthermore we let P_{i+1} be P_i and S_{i+1} be S_i.

2. v_i is a crossing vertex not included in R_i. In this case, if v_i is above or on P_i then we answer $c_i = 2$, otherwise we answer $c_i = 0$. Furthermore we let P_{i+1} be P_i and S_{i+1} be S_i.

3. v_i is a crossing vertex included in R_i but neither on S_i nor adjacent to a vertex of S_i. In this case, we answer $c_i = 1$: v_i becomes *forbidden*, that is, no P_j can pass through it for $j > i$. Furthermore we let P_{i+1} be P_i and S_{i+1} be S_i.

4. v_i is a crossing vertex either on S_i or adjacent to a vertex of S_i and S_i is an horizontal-right path. Let l'_i and l''_i be the first two adjacent vertices of S_i (with l''_i successive to l'_i) such that both the down-right paths from l'_i and l''_i to the border do not contain either a forbidden vertex or v_i. In

this case, we let P_{i+1} be P_i united with the path from l_i to l_i'', and we let S_{i+1} be the down-right path from l_i'' to the border. Furthermore, if v_i is in R_{i+1} then we answer $c_i = 1$, otherwise we answer $c_i = 2$ or $c_i = 0$ depending on v_i being on S_i or adjacent to a vertex of S_i.

5. v_i is a crossing vertex either on S_i or adjacent to a vertex of S_i and S_i is a down-right path. This case is treated similarly to the previous one.

Clearly, the construction of P_i can run for a large number of queries before either case 4 or case 5 fails: indeed, this cannot happen until the algorithm has made at least $n/2$ queries. Thus the above construction shows that we can keep going with hiding a 3-colored triangle to the algorithm for $n/2$ queries. Observe that $n/2$ is at least $\sqrt{m_n}/3$.

Moreover, the partial coloring determined by the first $n/2$ queries of the algorithm can be always extended so that either it is admissible (thus containing a 3-colored triangle) or it does not contain any 3-colored triangle (thus being not admissible). □

4. One more result on robustness

The last theorem we present shows that, contrary to Theorems 1.1 and 1.2, Theorem 1.3 can be proven with the sparseness condition removed.

Theorem 4.1. *Let N_0 and N_1 be two robustly complementary and categorical oracle Turing machines. Then, for all oracles A, $L(N_0^A) \in \mathrm{P}^{(\mathrm{UP}\cup co\mathrm{UP})\oplus A}$.*

Proof. Let H be the set of tuples $\langle\langle F\rangle, x, i, b\rangle$, where $\langle F\rangle$ denotes the encoding of a finite set F, such that an accepting path of either $N_0^F(x)$ or $N_1^F(x)$ exists so that the ith bit of its encoding is b. Since N_0 and N_1 are robustly complementary and categorical, it follows that, for each x and F, one and only one accepting path of either $N_0^F(x)$ or $N_1^F(x)$ exists. Thus, $H \in \mathrm{UP} \cup co\mathrm{UP}$. By using such a set H, the proof can then be carried out almost identically to that of Theorem 2.1 by Hartmanis & Hemachandra (1990). □

Acknowledgements

This research was partially supported by the MURST Project 'Efficienza di Algoritmi e Progetto di Strutture Informative'.

References

J.L. Balcázar, J. Díaz, and J. Gabarró, *Structural complexity I.* Springer-Verlag, 1988.

R. Beigel, NP-hard sets are P-superterse unless R=NP. Technical Report Technical Report 88-04, Johns Hopkins Department of Computer Science, August 1988.

L.E.J. Brouwer, Über abbildung von mannigfaltigkeiten. *Math. Ann.* **71** (1912), 97–115.

J. Hartmanis and L. Hemachandra, Robust machines accept easy sets. *Theoret. Comput. Sci.* **74** (1990), 217–225.

L. Hemachandra, Fault-tolerance and complexity. In *Proc. Twentieth Int. Coll. Autom., Lang. and Prog.*, 1993, 189–202.

M.D. Hirsh and S. Vavasis, Exponential lower bounds for finding Brouwer's fixed points. In *Proc. 28th Ann. IEEE Symp. Found. Comput. Sci.*, 1987, 401–410.

M.D. Hirsh, C.H. Papadimitriou, and S. Vavasis, Exponential lower bounds for finding Brouwer's fixed points. *J. Complexity* **5** (1989), 379–416.

K. Ko, On helping by robust oracle machines. In *Proc. Second Ann. Structure in Complexity Theory Conf.*, 1987, 182–190.

N. Megiddo and C.H. Papadimitriou, On total functions, existence theorems and computational complexity. *Theoret. Comput. Sci.* **81** (1991), 317–324.

C.H. Papadimitriou, On graph-theoretic lemmata and complexity classes. In *Proc. 31st Ann. IEEE Symp. Found. Comput. Sci.*, 1990, 794–801.

C.H. Papadimitriou, On the complexity of the parity argument and other inefficient proofs of existence. *J. Comput. System Sci.* **48** (1994), 498–532.

U. Schöning, Robust algorithms: a different approach to oracles. *Theoret. Comput. Sci.* **40** (1985), 57–66.

Yu. A. Shashkin, Fixed points. *Amer. Math. Soc.* (1991).

E. Sperner, Neuer beweis für die invarianz der dimensionszahl und des gebietes. *Abh. Math. Sem. Hamburg. Univ.* **6** (1928), 265–272.

Manuscript received November 4, 1993

Pierluigi Crescenzi
Dip.to di Scienze dell'Informazione
Università di Roma "La Sapienza"
Via Salaria 113
00198 Roma, Italy
`piluc@dsi.uniroma1.it`

Riccardo Silvestri
Dip.to di Scienze dell'Informazione
Università di Roma "La Sapienza"
Via Salaria 113
00198 Roma, Italy
`silver@dsi.uniroma1.it`