New Results about the List Access Problem

Salvador Roura
Conrado Martínez

Report LSI–96–63–R

New Results about the List Access Problem*

Salvador Roura and Conrado Martínez†

November 27, 1996

Abstract

We consider the list access problem and show that two unrealistic assumptions in the original cost model presented in [13] and subsequent literature allow for several competitiveness results of the move-to-front heuristic (MTF). We present an off-line algorithm for the list access problem and prove that, under a more realistic cost model, no on-line algorithm can be k-competitive for any constant k, MTF included.

1. Introduction

For the last years there has been a growing interest in the competitive analysis of on-line algorithms. Good examples are the study of the list access and update problems, the k-server problem, the memory paging and dynamic allocation problems, etc. (see for instance [13, 1, 4, 3, 6, 5, 6, 10])

Competitive analysis measures the quality of an on-line algorithm by comparing it against an optimal off-line algorithm, that is, one that is provided with full knowledge of the sequence of requests and serves it with minimal cost. In most situations, the performance of an algorithm servicing a request depends not only on the request itself but on some sort of internal state of the algorithm that can be changed (at a certain cost) after each request is attended. Off-line algorithms exploit full knowledge of the request sequence to evolve through a sequence of internal states while servicing each request, in a way that minimizes the total cost of servicing the whole sequence. On-line algorithms are deprived of this knowledge and should try to do their best at guessing future requests and spot regularities and patterns that appear in the sequence, to keep the total cost to a minimum. An on-line algorithm is defined to be k-competitive if the total cost to serve any (sufficiently long) sequence of requests is, at most, k times the cost of serving the same sequence with an optimal off-line algorithm [13].

In contrast, the goal of the so-called distributional analysis (also known as average-case analysis) is to compute the expected time to serve a request from a

*This research was supported by the ESPRIT BRA Program of the EC under contract no. 20244 (Project ALCOM-IT) and by a grant from CIRIT (Comissió Interdepartamental de Recerca i Innovació Tecnològica).

†Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, E-08028 Barcelona, Spain. E-mail: {roura,conrado}@goliat.upc.es

sequence of requests not known in advance to the on-line algorithm, under some distributional hypothesis. For instance, it is often assumed that requests are independently generated, so that a request of type i is generated with probability p_i irrespective of the time the request is made, and irrespective of previous or future requests. Typically, the on-line algorithm is compared against an off-line algorithm which knows the probabilistic properties of the source that generates the sequence of requests.

Many authors have claimed that competitive analysis provides stronger and more valuable results that distributional analysis since the former does not rely upon previous assumptions about the sequences of requests.

One of the earliest and most often cited results in this subject is due to Sleator and Tarjan [13]. One of the on-line algorithms that they studied is the move-to-front heuristic (MTF) for the list access and update problem in linked lists. The MTF heuristic moves the last accessed (the last inserted) item to the front of the linked list; the rationale behind this strategy is that a recently accessed item will be probably requested in the near future. Several previous studies had shown that MTF performed very well in practice and in theory from the distributional point of view [2, 7, 12].

The seminal paper of Sleator and Tarjan provided an strong argument in favor of MTF's case, following a new and original approach. Under their model of costs, they proved that MTF is 2-competitive or 4-competitive depending on the initial assumptions. That means that MTF cannot be beaten by any other algorithm, even an off-line one, by more than a constant factor.

Sleator and Tarjan's work opened a vast new area, namely, the competitive analysis of on-line algorithms. A great number of related papers considered variants of the original list access and update problems, as well as improvements of the MTF heuristic. Particularly successful is the use of randomization, giving 1.75-competitiveness (and even smaller competitiveness ratios) thus beating the lower bound of 2 for the competitiveness of deterministic on-line algorithms [1, 8, 11].

However, as we will discuss afterwards, we think that the result of Sleator and Tarjan is more often than not misunderstood, and its strenght can be questioned —our impression is that it has not been so in the past. To the best of our knowledge, nobody has provided sound arguments for (nor against) the model of costs for the list update problem proposed by Sleator and Tarjan. To begin with, no justification for this model was given in [13] nor serious attempts have been made to check the model's robustness. Thus, a big misunderstanding seems to have gone unnoticed for these past years. In the original paper [13] and ever since, the statement "MTF is 2-competitive under such and such cost model" has been loosely and inconciously replaced by "MTF is 2-competitive", implicitly assuming that no off-line algorithm can beat MTF by more than a constant factor, for any reasonable cost model. However, as we shall show hereafter, it turns out that under a reasonable and realistic cost model, there cannot exist an on-line algorithm for the list access and update problems which is k-competitive, for any constant k. Furthermore, our feeling is that Sleator and Tarjan's cost model is not realistic.

In short, the seemingly omnipotent off-line algorithm has to pay a prohibitive cost according to the model by Sleator and Tarjan. Although trying to take advantage of its knowledge, it is not worth for it the effort of doing anything but the same type of operations as MTF does. Hence, the conclusion that MTF is competitive follows, but we should be aware that MTF is competitive against a rather weakened off-line algorithm. In other contexts, for instance, in the study of competitive on-line paging algorithms, restrictions on the all-mighty adversary are achieved by means of the so-called access graphs (see for instance [9, 3]), and the strenght and limitations of the results are fairly well-understood.

In next section we point out two reasons that make Sleator and Tarjan's cost model unrealistic, and prove a trivial $\Theta(n)$ lower bound over the worst-case amortized cost per access of any on-line algorithm (either deterministic or randomized) to attend a sequence of n different accesses to a linked list of n items. In Section 3 we present an off-line algorithm that achieves $\Theta(\log n)$ amortized cost per access, when serving any sequence of requests and a more realistic model of costs is assumed. This implies that no on-line algorithm can be k-competitive for any constant k, including MTF.

2. THE COST MODEL

The cost model presented in [13] is as follows (we will only consider the static list model, i.e. no updates are allowed). Accessing the i-th item costs i. Immediately after an access to the item i, we are allowed to move it at no cost to any position closer to the front of the list (this is called a free exchange). Any other update in the list should consist of exchanges of consecutive items, where each one costs 1 and is called a paid exchange.

The first thing one could argue is that the cost of traversing a link can differ from the cost of an exchange, but it is not difficult to see that considering this difference cannot yield significantly different conclusions.

On the other hand, it is clear that there are not such free exchanges under a realistic model. However, while traversing the list in order to find the requested item it is simple to keep a pointer at some nearer position to the beginning of the list, and update the list at small constant cost. Again, considering that the zero cost free exchanges have in fact some constant cost does not affect the conclusions of [13] and many other paper on the subject. So this is not the point, either.

There are two strong reasons that make Sleator and Tarjan's cost model not utterly realistic.

First of all, in practice, there is no way of exchanging two consecutive items if they are not reached before. Hence, we cannot consider that exchanging two consecutive items has cost 1, if they are farther away and after the accessed item.

Secondly, after reaching the item i, any item before it in the list could be moved closer to the beginning of the list by means of a free (or constant time) exchange, for the same reason that we are allowed to use free exchanges with

the i-th item and move it to the front if we wish so. In other terms, the cost of moving such an item j positions closer to the front of the list should not cost j units, but a constant amount of time.

The first of these two reasons does not affect the main conclusion about MFT competitiviness. Considering more realistic costs for paid exchanges, as explained above, actually means that the off-line algorithm would have to pay more than it was charged by the Sleator-Tarjan cost model, while MTF's cost would remain the same as it does not use paid exchanges.

The second reason above turns out to change things dramatically. In next section we will see how a simple off-line algorithm can profit by these generalized free (constant time) exchanges to defeat any on-line algorithm (MTF included).

One simple way to state our objection to Sleator and Tarjan's model is that $\Theta(n^2)$ inversions can be removed with linear cost and not quadratic, once we allow for usual operations on linked lists and their costs are accounted in a realistic manner.

We end up this section deducing a simple linear lower bound on the time of any on-line algorithm (either deterministic or randomized) dealing with a sequence of n different requests over a list with n elements. Thus, we restrict our attention to sequences of accesses that are permutations of the n items in the list. After all the items have been accessed, another sequence of n requests could be served, and so on. This simplified model will be enough for our purposes, and there is no loss of generality.

Let $\mathcal{A}$ be any on-line algorithm that deals with a sequence of n distinct requests to access the n elements of a linked list. Consider the initial state from the point of view of an adversary trying to induce worst-case behaviour. Even if we only know the probability that each permutation has of being the current configuration of the list, it is very easy to see that we can always force, at least, $(n+1)/2$ expected comparisons/pointers traversed to serve a request —this happens when every item has the same expected search time; for instance, when all the permutations are equally likely. After that, the best thing that can happen to $\mathcal{A}$ is that it moves the requested item at the end of the list, since this item will not be accessed any more. Now we are in the same situation as before, except that, from the point of view of the adversary, the list has $n-1$ elements. This argument shows that it is always possible to find a sequence of requests whose expected cost is $(n+1)/2 + n/2 + \cdots + 1 = n(n+3)/4$. Therefore, $\mathcal{A}$ requires $\Theta(n)$ expected time per access to serve the sequence of requests.

3. THE OFF-LINE ALGORITHM

Roughly speaking, our off-line algorithm works as follows. To serve the first request, the algorithm visits all the nodes of the list, irrespective of the requested item. Although the algorithm pays n at this step, it can save much of the future work by means of a simple strategy, which consists in splitting the original list into three sublists, one with the requested item, and two additional sublists with $(n-1)/2$ elements each. The first one of these sublists (*first*) contains the items that will be requested first, the other (*second*) the items that will be requested

Stage	State of the list	Cost
1	10, 8, 3, 6, 15, 5, 1, 14, 11, 2, 9, 4, 7, 13, 12*	15
	8, 3, 6, 5, 2, 4, 7*, 10, 15, 14, 11, 9, 13, 12, 1	7
	3, 5, 4*, 8, 6, 7, 2, 10, 15, 14, 11, 9, 13, 12, 1	3
	4*, 5, 3, 8, 6, 7, 2, 10, 15, 14, 11, 9, 13, 12, 1	1
5	4, 5*, 3, 8, 6, 7, 2, 10, 15, 14, 11, 9, 13, 12, 1	2
	5, 4, 3, 8, 6, 7*, 2, 10, 15, 14, 11, 9, 13, 12, 1	6
	7*, 8, 6, 5, 4, 3, 2, 10, 15, 14, 11, 9, 13, 12, 1	1
	7, 8*, 6, 5, 4, 3, 2, 10, 15, 14, 11, 9, 13, 12, 1	2
	8, 7, 6, 5, 4, 3, 2, 10, 15, 14, 11, 9, 13, 12*, 1	14
10	10, 11, 12*, 15, 14, 13, 9, 8, 7, 6, 5, 4, 3, 2, 1	3
	11*, 12, 10, 15, 14, 13, 9, 8, 7, 6, 5, 4, 3, 2, 1	1
	11, 12*, 10, 15, 14, 13, 9, 8, 7, 6, 5, 4, 3, 2, 1	2
	12, 11, 10, 15, 14, 13*, 9, 8, 7, 6, 5, 4, 3, 2, 1	6
	14*, 15, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1	1
15	14, 15*, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1	2
	15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1	

Figure 1: An execution of the off-line algorithm. An asterisk (*) marks the farthest element to the right reached during the corresponding stage. The sequence of requests is $\sigma = 1, 2, \ldots, n$ with $n = 15$.

afterwards. The new list is build by joining *first*, *second* and the requested item in this order. This ensures that the following $(n-1)/2$ requests will hit elements in the first half of the list. Therefore, these requests can be recursively served by the same algorithm, but now over a sublist half the size of the original one. This is the point of the algorithm.

When all the elements in the first sublist have been already accessed, the next requests will be for the items in the second sublist. The first of these requests is served by

1. skipping *previous*, the sublist with the $(n-1)/2$ previously accessed items;
2. acting exactly as described for the general case but over the next $(n-1)/2$ items (that is, splitting this sublist into three sublists and joining them), and
3. joining the result of the step above with *previous*.

Again, the new list has at its front a sublist with the $(n-1)/2$ elements that are going to be immediately requested (except for the farthest of them, which is the one just accessed). Hence, we can also use the algorithm recursively to serve the second sublist and end up servicing all the requests.

Let L be the original list to be accessed. For the sake of simplicity, let us assume that $n = 2^m - 1$ for some $m \geq 1$.

We give a recursive version of the off-line algorithm serve(*list*, *sz*, *previous*). The meaning of the parameters is as follows: *list* is a pointer to the beginning of the linked list, $sz = 2^k - 1$ is the number of items to be serviced by the current recursive call to serve, for some $1 \leq k \leq m$, *previous* is a pointer to a list of already serviced items that will be skipped and added to the end of *list*. The first call to serve is thus serve(L, n, **NIL**).

We assume that the sequence of requests σ is globally accesible to all the procedures below. Requests are attended one at a time and the global variable t, which is initialized to 1, indicates the next request (σ_t) to be attended.

A possible execution of this algorithm is given in Figure 3. Each row reflects the order of the items in the linked list in each stage, the initial row being the initial state of the list and each successive row showing a step of the process. In the example, we assume that $\sigma = 1, 2, 3, \ldots, n$; with $n = 15$. The asterisks mark the farthest reached item in each step, and the last column keeps track of the cost of each stage, in this case, the number of visited items.

The analysis of the cost of this algorithm is quite simple. Let $T(k)$ be the number of visited items while serving a sequence with $2^k - 1$ requests. By definition, $T(1) = 1$. Furthermore, for every $k \geq 2$ we have that

$$T(k) = (2^k - 1) + T(k-1) + (2^{k-1} - 1) + T(k-1) = 3 \cdot 2^{k-1} - 2 + 2T(k-1).$$

A simple proof by induction yields $T(k) = 3k \cdot 2^{k-1} - 2^{k+1} + 2$. We can use it to compute the amortized time to serve a request in the whole list as

$$\frac{T(m)}{n} = \frac{3m}{2} - 2 + \frac{3m}{2^{m+1} - 2} = \frac{3}{2}\log_2 n + \mathcal{O}(1) = \Theta(\log n).$$

Algorithm 1 The off-line algorithm.

```
serve(list, sz, previous) {

  classify(list, sz, first, second, current, rest);
  // Traverse the first sz items in list.
  // Build three new lists by adding elements at their ends:
  //      first:   contains the (sz - 1) / 2 items which,
  //               according to σ, will be accessed first
  //      second:  contains the (sz - 1) / 2 items that
  //               will be accessed afterwards
  //      current: contains the currently requested item
  // Also, return rest, a pointer to the sz + 1-th item in list.

  attend(current); t = t + 1;
  // Service the request to the item pointed to by current.

  join(first, second, current, previous, rest, list);
  // Concatenate the lists first, second, current, previous and rest,
  // in this order, and return list, a pointer to the head of the so
  // constructed linked list.

  if (sz == 1) return;

  serve(list, (sz - 1) / 2, NIL);

  chop(list, (sz - 1)/2 , previous);
  // Traverse (sz - 1)/ 2 items in list.
  // These items form a new linked list, called previous.
  // After the call, list points to the beginning of the remaining
  // elements.

  serve(list, (sz - 1)/2, previous);
}
```

Finally and to be completely rigorous, we should mention that we have not shown which is the cost of deciding whether an item must be attached to *first* or *second*, while we are classifying the list.

Let $\hat{\sigma}$ denote the inverse permutation of σ. It may be computed in time $\mathcal{O}(n)$. Assume we have to decide wheter some item $k \neq \sigma_t$ goes either to *first* or to *second*, while traversing the *list* to classify it. Recall that t denotes the step or index of the item to be serviced. Then, k should be added to the list *first* if $\hat{\sigma}_k \leq t + (sz - 1)/2$ and to *second* otherwise. The linear cost of building $\hat{\sigma}$ at the beginning, plus the constant cost to classify one item, does not change the conclusion that the amortized cost per access is $\mathcal{O}(\log n)$. Furthermore, we are taking here into account costs which are not usually considered in the literature when computing the cost of servicing the sequence σ by the optimal off-line algorithm.

4. CONCLUSIONS

We have considered the list access problem and proved that, under a realistic cost model, no on-line algorithm to deal with it can be k-competitive. In particular, we have shown that the result about the 2-competitiveness of MTF follows from the high costs associated to those operations that would enable an off-line algorithm to take advantadge of its full knowledge of the sequence of requests.

We think that our contribution is twofold. On the one hand, we can conclude that the knowledge of the future sequence of requests indeed improves the average time per access. On the other, although competitive analysis is a worthwhile technique for the study of on-line problems, more attention should be paid to the assumptions upon which competitiviness results are based.

The new open problem is thus if there exists any on-line algorithm that using non-local exchanges would do much better than MTF. Our conjecture is that such an algorithm does not exist. Basically, we conjecture that no on-line algorithm could take advantage of moving non-requested items far away from their positions towards the front of the list. Since only local exchanges between consecutive elements and moving the last accessed item towards the front of the list seem to make sense for list access heuristics, the conclusion should be that MTF is "as good as" any other on-line algorithm and would follow from Sleator and Tarjan's result.

REFERENCES

[1] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1):2–14, 1994.

[2] J.L. Bentley and C. C. McGeoch. Amortized analyses of self-organizing sequential search heuristics. *Communications of the ACM*, 28(4):404–411, 1985.

[3] A. Borodin, S. Irani, Raghavan, and Schieber. Competitive paging with locality of reference. In *ACM Symposium on Theory of Computing (STOC)*, 1991.

[4] A. Borodin, N. Linial, and M. Saks. An optimal on-line algorithm for metrical task systems. *Journal of the ACM*, 39(4):743–763, 1992.

[5] A. Fiat, R.M. Karp, M. Luby, L.A. McGeoch, D.D. Sleator, and N.E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12:685–699, 1991.

[6] A. Fiat, Rabani, and Ravid. Competitive k-server algorithms. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, 1990.

[7] G. Gonnet, J.I. Munro, and Suwanda. Exegesis of self-organizing linear search. *SIAM Journal on Computing*, 1981.

[8] S. Irani. Two results on the list update problem. *Information Processing Letters*, 38:301–306, 1991.

[9] S. Irani, A.R. Karlin, and S. Phillips. Strongly competitive algorithms for paging with locality of reference. *SIAM Journal on Computing*, 25(3):477–497, 1996.

[10] M.S. Manasse, L.A. McGeoch, and D.D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230, 1990.

[11] N. Reingold, J. Westbrook, and D.D. Sleator. Randomized competitive algorithms for the list update problem. *Algorithmica*, 11(1):15–32, 1994.

[12] Rivest. On self-organizing sequential search heuristics. *Communications of the ACM*, 19, 1976.

[13] Sleator and Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Research Reports – 1996

LSI–96–1–R "(Pure) Logic out of Probability", Ton Sales.

LSI–96–2–R "Automatic Generation of Multiresolution Boundary Representations", C. Andújar, D. Ayala, P. Brunet, R. Joan-Arinyo, and J. Solé.

LSI–96–3–R "A Frame-Dependent Oracle for Linear Hierarchical Radiosity: A Step towards Frame-to-Frame Coherent Radiosity", Ignacio Martin, Dani Tost, and Xavier Pueyo.

LSI–96–4–R "Skip-Trees, an Alternative Data Structure to Skip-Lists in a Concurrent Approach", Xavier Messeguer.

LSI–96–5–R "Change of Belief in SKL Model Frames (Automatization Based on Analytic Tableaux)", Matías Alvarado and Gustavo Núñez.

LSI–96–6–R "Compressibility of Infinite Binary Sequences", José L. Balcázar, Ricard Gavaldà, and Montserrat Hermo.

LSI–96–7–R "A Proposal for Word Sense Disambiguation using Conceptual Distance", Eneko Agirre and German Rigau.

LSI–96–8–R "Word Sense Disambiguation Using Conceptual Density", Eneko Agirre and German Rigau.

LSI–96–9–R "Towards Learning a Constraint Grammar from Annotated Corpora Using Decision Trees", Lluis Màrquez and Horacio Rodríguez.

LSI–96–10–R "POS Tagging Using Relaxation Labelling", Lluís Padró.

LSI–96–11–R "Hybrid Techniques for Training HMM Part-of-Speech Taggers", Ted Briscoe, Greg Grefenstette, Lluís Padró, and Iskander Serail.

LSI–96–12–R "Using Bidirectional Chart Parsing for Corpus Analysis", A. Ageno and H. Rodríguez.

LSI–96–13–R "Limited Logical Belief Analysis", Antonio Moreno.

LSI–96–14–R "Logic as General Rationality: A Survey", Ton Sales.

LSI–96–15–R "A Syntactic Characterization of Bounded-Rank Decision Trees in Terms of Decision Lists", Nicola Galesi.

LSI–96–16–R "Algebraic Transformation of Unary Partial Algebras I: Double-Pushout Approach", P. Burmeister, F. Rosselló, J. Torrens, and G. Valiente.

LSI–96–17–R "Rewriting in Categories of Spans", Miquel Monserrat, Francesc Rosselló, Joan Torrens, and Gabriel Valiente.

LSI–96–18–R "On the Depth of Randomly Generated Circuits", Tatsuie Tsukiji and Fatos Xhafa.

LSI–96–19–R "Learning Causal Networks from Data", Ramon Sangüesa i Solé.

LSI–96–20–R "Boundary Generation from Voxel-based Volume Representations", R. Joan-Arinyo and J. Solé.

LSI–96–21–R "Exact Learning of Subclasses of CDNF Formulas with Membership Queries", Carlos Domingo.

LSI–96–22–R "Modeling the Thermal Behavior of Biosphere 2 in a Non-Controlled Environment Using Bond Graphs", Angela Nebot, François E. Cellier, and Francisco Mugica.

LSI–96–23–R "Obtaining Synchronization-Free Code with Maximum Parallelism", Ricard Gavaldá, Eduard Ayguadé, and Jordi Torres.

LSI–96–24–R "Memoisation of Categorial Proof Nets: Parallelism in Categorial Processing", Glyn Morrill.

LSI–96–25–R "Decision Trees Have Approximate Fingerprints", Víctor Lavín and Vijay Raghavan.

LSI–96–26–R "Visible Semantics: An Algebraic Semantics for Automatic Verification of Algorithms", Vicent-Ramon Palasí Lallana.

LSI–96–27–R "Massively Parallel and Distributed Dictionaries on AVL and Brother Trees", Joaquim Gabarró and Xavier Messeguer.

LSI–96–28–R "A Maple package for semidefinite programming", Fatos Xhafa and Gonzalo Navarro.

LSI–96–29–R "Bounding the expected length of longest common subsequences and forests", Ricardo A. Baeza-Yates, Ricard Gavaldà, and Gonzalo Navarro.

LSI–96–30–R "Parallel Computation: Models and Complexity Issues", Raymond Greenlaw and H. James Hoover.

LSI–96–31–R "ParaDict, a Data Parallel Library for Dictionaries (Extended Abstract)", Joaquim Gabarró and Jordi Petit i Silvestre.

LSI–96–32–R "Neural Networks as Pattern Recognition Systems", Lourdes Calderón.

LSI–96–33–R "Semàntica externa: una variant interessant de la semàntica de comportament" (written in Catalan), Vicent-Ramon Palasí Lallana.

LSI–96–34–R "Automatic verification of programs: algorithm ALICE", V.R. Palasí Lallana.

LSI–96–35–R "Multiresolution Approximation of Polyhedral Solids", D. Ayala, P. Brunet, R. Joan-Arinyo, I. Navazo.

LSI–96–36–R "Algebraic Transformation of Unary Partial Algebras II: Single-Pushout Approach", P. Burmeister, M. Montserrat, F. Rosselló, and G. Valiente.

LSI–96–37–R "Probabilistic Conditional Independence: A Similarity-Based Measure and its Application to Causal Network Learning", Ramon Sangüesa Solé, Joan Cabós Fabregat, and Ulises Cortés García.

LSI–96–38–R "Analysing the Process of Enforcing Integrity Constraints", Enric Mayol and Ernest Teniente.

LSI–96–39–R "Reducció de l'equivalència inicial visible a teoremes inductius" (written in Catalan), Vicent-Ramon Palasí Lallana.

LSI–96–40–R "A Compendium of Problems Complete for Symmetric Logarithmic Space", Carme Àlvarez and Raymond Greenlaw.

LSI–96–41–R "Semàntica algebraica del llenguatge AL: l'algorisme α" (written in Catalan), V.R. Palasí Lallana.

LSI–96–42–R "Partial Occam's Razor and its Applications", Carlos Domingo, Tatsuie Tsujiki, and Osamu Watanabe.

LSI–96–43–R "Transparent Distributed Problem Resolution in the MAKILA Multi-Agent System", Karmelo Urzelai.

LSI–96–44–R "The Intensional Events Method for Consistent View Updating", Dolors Costal, Ernest Teniente, and Toni Urpí.

LSI–96–45–R "Extending Eiffel as a Full Life-cycle Language", Alonso J. Peralta and Joan Serras.

LSI–96–46–R "Analysis of Methods for Generating Octree Models of Objects from Their Silhouettes", Marta Franquesa and Pere Brunet.

LSI–96–47–R "Learning nearly monotone k-term DNF", Jorge Castro, David Guijarro, and Víctor Lavín.

LSI–96–48–R "Learning Monotone Term Decision Lists", David Guijarro, Víctor Lavín, and Vijay Raghavan.

LSI–96–49–R "Coding Complexity: The Computational Complexity of Succinct Descriptions", José L. Balcázar, Ricard Gavaldà, and Osamu Watanabe.

LSI–96–50–R "Algorithms for Learning Finite Automata from Queries: A Unified View", José L. Balcázar, Josep Díaz, Ricard Gavaldà, and Osamu Watanabe.

LSI–96–51–R "Mètodes de validació d'esquemes de bases de dades deductives" (written in Catalan), Carles Farré.

LSI–96–52–R "The Parallel Complexity of Positive Linear Programming", Luca Trevisan and Fatos Xhafa.

LSI–96–53–R "Polynomial-Time Algorithms for Some Self-Duality Problems", Carlos Domingo.

LSI–96–54–R "Patterns in Random Binary Search Trees", Philippe Flajolet, Xavier Gourdon, and Conrado Martínez.

LSI–96–55–R "El llenguatge ROSES. Part I" (written in Catalan), M. Barceló, P. Costa, D. Costal, A. Olivé, C. Quer, A. Roselló, M.R. Sancho.

LSI–96–56–R "Double-Pushout Hypergraph Rewriting through Free Completions", P. Burmeister, F. Rosselló, G. Valiente.

LSI–96–57–R "On User-Defined Features", Christoph M. Hoffmann and Robert Joan-Arinyo.

LSI–96–58–R "Light Transfer Equations for Volume Visualization", Francesc Sala i Valcàrcel and Daniela Tost i Pardell.

LSI–96–59–R "Computing the Medial Axis Transform of Polygonal Objects by Testing Discs", Josep Vilaplana.

LSI–96–60–R "Linear Lower Bounds and Simulations in Frege Systems with Substitutions", M. Bonet and N. Galesi.

LSI–96–61–R "Prototipado de programas usando especificaciones funcionales y no funcionales" (written in Spanish), Xavier Franch and Pere Botella.

LSI–96–62–R "Transformación de restricciones de integridad dinámicas definidas hacia el futuro en una forma definida hacia el pasado", Maria Amélia Pacheco Silva and Maria Ribera Sancho i Samsó.

LSI–96–63–R "New Results about the List Access Problem", Salvador Roura and Conrado Martínez.

Hardcopies of reports can be ordered from:

Nuria Sánchez
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Pau Gargallo, 5
08028 Barcelona, Spain
secrelsi@lsi.upc.es

See also the Department WWW pages, http://www-lsi.upc.es/www/