Matching Algorithms for Bipartite Graphs

Herbert Alexander Baier Saip

Cláudio Leonardo Lucchesi

Relatório Técnico DCC–03/93

Março de 1993

Matching Algorithms for Bipartite Graphs

Herbert Alexander Baier Saip*

Cláudio Leonardo Lucchesi†

Abstract

This report is a rather concise summary of the first-named author's M. Sc. Dissertation, written under the second-named author's supervision.

The matching problem in graphs consists in determining matchings, that is, vertex disjoint sets of edges of the graph. In particular, we are interested in finding maximum matchings, that is, matchings of maximum cardinality. There are many variations around this problem, the graph can be: bipartite or not, weighted or not.

In this work we describe briefly the most important algorithms for solving the problem of maximum matching, weighted or not, in bipartite graphs. At the end of the article we give a table which describes briefly the most important algorithms for solving the general problem, in which the graph is not necessarily bipartite.

*DCC - IMECC - UNICAMP. Supported by CNPq, FAPESP and FAEP.

†DCC - IMECC - UNICAMP. lucchesi@dcc.unicamp.br. Supported in part by a grant from CNPq.

1 Introduction

This report is a rather concise summary of the first-named author's M. Sc. Dissertation, written under the second-named author's supervision [4].

Let $G(V, E)$ be an undirected graph with vertex set V and edge set E (or, simply, G, if both V and E are understood), where $|V| = n$ and $|E| = m$.

A graph G is *bipartite* if V can be partitioned into two sets, V^+ and V^-, so that each edge of G has one end in V^+ and the other end in V^-.

A subset M of E is called a *matching* in G if no vertex is incident to more than one edge in M. A matching of maximum cardinality is called a *maximum matching*. If every vertex of G is incident to some edge of M, then M is called a *perfect matching*; clearly, every perfect matching is maximum. A matching M in graph G is *maximal* if it is not properly contained in any other matching (see figure 1).[1] A vertex $v \in V$ is *M-free* if it is incident with no edge in M.

(a) (b)

Figure 1: (a) Maximal matching. (b) Perfect Matching.

A graph G is weighted if we give a *cost function* c, that associates each edge with a real value, that is, $c : E \rightarrow \Re$. Let X be a subset of E.

[1] Dashed lines represent edges not in M and full lines represent edges in M.

The cost of set X is

$$c(X) := \sum_{\alpha \in X} c(\alpha).$$

Suppose that E' is a subset of E. The subgraph of G whose vertex set is the set of vertices incidents to edges in E' and whose edge set is E' is called the subgraph of G *induced* by E' and is denoted by $G[E']$.

Let M be a matching in G. An *M-alternating path* in G is a path whose edges are alternately in $E \setminus M$ and M. An *M-augmenting path* is a nondegenerate M-alternating path whose origin and terminus are both M-free (see figure 1a). A proof of the following theorem can be found in [8, page 70].

Theorem 1 (Berge [5], 1957) *A matching M in G is maximum if and only if G contains no M-augmenting path.* □

The next lemma shows how we can augment the matching, given an M-augmenting path.

Lemma 2 *Let M be a matching in G and P an M-augmenting path. Set $N := E \oplus EP$ is a matching in G and $|N| = |M| + 1$, where EP is the edge set of P and $E \oplus EP$ denotes the symmetric difference of sets E and EP.* □

In Section 2 we describe briefly the most important algorithms for finding maximum matchings in non weighted bipartite graphs (Table 1 presents the described algorithms[2]).

In Section 3 we describe briefly the most important algorithms for finding maximum matchings in weighted bipartite graphs (Table 2 presents the described algorithms).

In Section 4 we give a table which describes briefly the most important algorithms for solving the general problem, in which the graph is not necessarily bipartite.

[2] In all tables we use p to represent the number of processors used by parallel algorithms and U to denote the greatest absolute value among edge cost values.

Sequential Algorithms			
Date	Authors	Ref.	Complexity
	Primitive		$O(mn)$
1973	Hopcroft and Karp	[26]	$O(mn^{1/2})$
1991	Alt, Blum, Mehlhorn and Paul	[3]	$O(n^{3/2}\sqrt{m/\log n})$

Parallel Algorithms				
Date	Authors	Ref.	Complexity	Processors
1987	Kim and Chwa	[27]	$O(n \log n \log\log n)$	$O(n^3/\log n)$
1988	Goldberg, Plotkin and Vaidya	[24]	$O(n^{2/3}\log^3 n)$	$O(n^3/\log n)$
1992	Goldberg, Plotkin, Shmoys and Tardos	[23]	$O(m^{1/2}\log^3 n)$	$O(m^3)$

Table 1: Non weighted bipartite graphs.

Sequential Algorithms			
Date	Authors	Ref.	Complexity
1955	Kuhn	[29]	$O(mn^2)$
1972	Edmonds and Karp	[13]	$O(mn \log n)$
1984	Fredman and Tarjan	[13, 16]	$O(mn + n^2 \log n)$
1985	Gabow	[19]	$O(mn^{3/4} \log U)$
1989	Gabow and Tarjan	[21]	$O(mn^{1/2} \log nU)$

Parallel Algorithms				
Date	Authors	Ref.	Complexity	Processors
1988	Goldberg, Plotkin and Vaidya	[24]	$O(n^{2/3}\log^3 n \log nU)$	$O(n^3/\log n)$
1988	Gabow and Tarjan	[20]	$O((mn^{1/2}/p)\log p \log nU)$	$O(m/(n^{1/2}\log^2 n))$

Table 2: Weighted bipartite graphs.

2 Non Weighted Bipartite Graphs

In this section we describe briefly some of the most important algorithms for finding maximum matchings in non weighted bipartite graphs.

2.1 Primitive Algorithm

- Type: sequential.
- Complexity: $O(mn)$.

This algorithm uses Theorem 1 (Berge). Since the graph is bipartite, a breadth-first search may be used to find an M-augmenting path in the following way:

1. Begin the search from M-free vertices in one bipartition of G, say V^+.
2. In odd iterations of the search use $(E \setminus M)$-edges and in even iterations use M-edges.
3. If the search finds an M-free vertex in V^-, then we have an M-augmenting path.

Remarks:

1. If the search finishes without finding an M-free vertex in V^- then there is no M-augmenting path in G, therefore M is maximum, by Theorem 1.
2. If the search finishes as soon as an M-free vertex in V^- is found, then we have a shortest M-augmenting path in G.
3. The search time bound is $O(m+n)$, which is $O(m)$ if we assume G to be connected.

The primitive algorithm consists of $O(n)$ phases, each of which consists of a breadth-first search for an augmenting path. If any augmenting path exists, the search finds a shortest one and the matching is augmented (Lemma 2). Therefore, the algorithm time bound is $O(mn)$.

2.2 Hopcroft and Karp's Algorithm

- Authors: Hopcroft and Karp [26].
- Date: 1973.
- Type: sequential.
- Complexity: $O\left(mn^{1/2}\right)$.

Unlike the primitive algorithm (Section 2.1), which finds only one M-augmenting path in each phase, Hopcroft and Karp's algorithm searches for a maximal disjoint collection of shortest M-augmenting paths.

This collection can be found using the breadth-first search described in Section 2.1. The search stops as soon as an M-free vertex in V^- is visited, though it completes that iteration, looking for other M-free vertices at the same level. Then, a depth-first search is performed, starting at M-free vertices in V^-, using only vertices visited in the first search, to determine the collection. The search time bound is $O(m+n)$, which is $O(m)$ if we assume G to be connected.

Hopcroft and Karp discovered that if the matching is augmented using a maximal collection in each phase, instead of just one augmenting path, the number of phases is reduced to $O(\sqrt{n})$. Thus, Hopcroft and Karp's algorithm time bound is $O(m\sqrt{n})$. This is the most efficient sequential algorithm known, provided G is not dense. When the graph is dense it is perhaps better to use an algorithm due to Alt, Blum, Mehlhorn and Paul [3], which has complexity $O(n^{3/2}\sqrt{m/\log n})$.

Even and Tarjan [15] observed the similarity between Hopcroft and Karp's algorithm and Dinic's algorithm [10].

2.3 Kim and Chwa's Parallel Algorithm

- Authors: Kim and Chwa [27].
- Date: 1987.
- Type: parallel.
- Complexity: $O(n \log n \log\log n)$.
- Processors: $O(n^3/\log n)$.
- Model: PRAM and CREW.

This algorithm is similar to the primitive algorithm (Section 2.1), however the search for shortest M-augmenting paths is done in parallel. Instead of a breadth-first search, this algorithm uses an even simpler mechanism for finding M-augmenting paths, namely, a straightforward parallel computation of the product of adjacency matrices.

2.4 Goldberg, Plotkin and Vaidya's Parallel Algorithm

- Authors: Goldberg, Plotkin and Vaidya [24].
- Date: 1988.
- Type: parallel.
- Complexity: $O\left(n^{2/3} \log^3 n\right)$.
- Processors: $O(n^3/\log n)$.
- Model: PRAM and CRCW.

This algorithm works in two phases. In the first phase, a variation of an algorithm called *preflow* [25] is used. This variation is efficient when there are many M-free vertices incident to shortest M-augmenting paths. In the second phase, Kim and Chwa's algorithm (Section 2.3) is then used. A proper balancing of the two phases leads to a sublinear running time.

3 Weighted Bipartite Graphs

When the graph is weighted, there are several variants of the problem, some of which are outlined below:

(i) Minimum cost perfect matching - among perfect matchings, choose one having minimum cost.

(ii) Minimum cost matching - among all matchings, choose one having minimum cost.

(iii) Minimum cost maximum matching - among all maximum matchings, choose one having minimum cost.

(iv) Maximum cost perfect matching - among all perfects matchings, choose one having maximum cost.

(v) Maximum cost matching - among all matchings, choose one having maximum cost.

(vi) Maximum cost maximum matching - among all maximum matchings, choose one having maximum cost.

Lemma 3 *The following problems are equivalent: (i) and (iv), (ii) and (v), and (iii) and (vi).*

Proof: The reductions are done by simply replacing cost function c by $-c$. □

Problems (ii) and (iii) are reduced to problem (i). The reductions may be found in [4]. Thus, variants (ii)–(vi) are reduced to variant (i).

We use a *dual variable* $y : V \rightarrow \Re$ in order to solve the problem. If

$$y(u) + y(v) \leq c(u, v), \ \forall (u, v) \in E, \tag{1}$$

then y is called *c-independent*.

We denote by $E_{y,c}$ the set of edges satisfying (1) with equality, that is,

$$E_{y,c} := \{(u,v) \in E : y(u) + y(v) = c(u,v)\}.$$

Let M be a matching in G and y be a dual variable. The pair (M, y) is *c-admissible* if y is c-independent and the following properties are satisfied:

(i) $M \subseteq E_{y,c}$.

(ii) For every non M-free vertex v of V^+ (V^-) and for every M-free vertex w in V^+ (respectively, V^-), $y(v) \leq y(w)$.

The next Lemma in fact shows that it is possible to find a minimum cost matching of any given cardinality. This result is presented in [4].

Lemma 4 *If (M, y) is a c-admissible pair, then M has minimum cost among all matchings in G of cardinality $|M|$.*

Proof: Let N be a matching in G such that $|N| = |M|$. Adding (1) over all edges in N yields

$$y(VN) \leq c(N),$$

where VN is the set of vertices incident to edges in N.

Since $M \subseteq E_{y,c}$, each M-edge satisfies (1) with equality. Adding (1) over all edges in M yields

$$y(VM) = c(M),$$

where VM is the set of vertices incident to edges in M.

Let $X := VM \cap V^+ \setminus VN$ and $Y := VN \cap V^+ \setminus VM$. By definition of c-admissible pair (ii), for each vertex v in X and for each vertex w in Y, $y(v) \leq y(w)$. By hypothesis, $|M| = |N|$, whence $|X| = |Y|$. Thus,

$$y(VM \cap V^+) \leq y(VN \cap V^+).$$

Similarly, $y(VM \cap V^-) \leq y(VN \cap V^-)$. Thus,

$$y(VM) \leq y(VN).$$

Therefore,

$$c(M) = y(VM) \leq y(VN) \leq c(N). \ \square$$

Corollary 5 *Let M be a perfect matching in G. If y is c-independent and $M \subseteq E_{y,c}$, then M is a minimum cost perfect matching in G.* $\square$

3.1 Hungarian Search

- Authors: Kuhn [29, 30] and Munkres [34].
- Date: 1955 and 1957.
- Type: sequential.
- Complexity: $O\left(mn^2\right)$.

This algorithm finds a minimum cost maximum matching in G. It has two important variables, namely, M, a matching, and y, a dual variable. During the execution of the algorithm, the following property is preserved: pair (M, y) is c-admissible. The Hungarian search consists of phases, each of which is composed of two steps:

1. Usage of the primitive algorithm (Section 2.1) in order to extend M to a maximum matching in $G[E_{y,c}]$.
2. If M is maximum, then by Lemma 4 it has minimum cost among all maximum matchings in G; otherwise, maintaining pair (M, y) c-admissible, an adjustment in y is done in order to enlarge $VG[E_{y,c}]$. There is no guarantee that an M-augmenting path will then be found but in that case new adjustments will be done.

3.2 Edmonds and Karp's Algorithm

- Authors: Edmonds and Karp [13].
- Date: 1972.
- Type: sequential.
- Using heap [1, 31], complexity: $O(mn \log n)$.
- Using Fibonacci heap [16], complexity: $O(mn + n^2 \log n)$.

This algorithm finds a minimum cost maximum matching in G.

Dijkstra's algorithm [9] is used to determine the adjustment of the dual variable, in such a way that each run of Dijkstra's algorithm corresponds to one adjustment of the dual variable. Moreover, every adjustment is guaranteed to yield at least one augmenting path in $G[E_{y,c}]$. Thus, this algorithm requires $O(n)$ phases of the Hungarian Search (Section 3.1).

Fredman and Tarjan, in 1984, developed a new heap implementation, called *Fibonacci heap* [16], that improves the running time to

$$O\left(mn + n^2 \log n\right).$$

3.3 Algorithm with Scaling – Gabow

- Author: Gabow [19].
- Date: 1985.
- Type: sequential.
- Complexity:[3] $O\left(mn^{3/4} \log U\right)$.
- Restriction: integrality of cost function $c : E \rightarrow \mathcal{Z}$.

[3] U is the greatest absolute value among edge costs.

This algorithm finds a minimum cost perfect matching in G. By *scaling* the following technique is meant: for each edge, a cost equal to half of the original cost is taken, and a solution is found recursively; based on this solution the algorithm determines a solution for the original problem. It is clear that this technique requires integrality of cost function c.

This technique was introduced, in this context, by Edmonds and Karp in 1972 [13].

3.4 Scaling and Approximation Algorithm of Gabow and Tarjan

- Authors: Gabow and Tarjan [21].
- Date: 1989.
- Type: sequential.
- Complexity: $O\left(mn^{1/2}\log nU\right)$.
- Restriction: integrality of cost function $c : E \to \mathcal{Z}$.

This algorithm finds a minimum cost perfect matching in G, using scaling and approximation. By *approximation* we mean the following: cost function c is uniformly multiplied by a sufficiently large constant K (in this case, $K = \left\lceil \frac{n+1}{2} \right\rceil$); thus any matching M whose cost $c(M)$ is sufficiently near the optimum cost C (that is, $c(M) < C + K$) is in fact optimal.

3.5 Scaling and Approximation Parallel Algorithm of Goldberg, Plotkin and Vaidya

- Authors: Goldberg, Plotkin and Vaidya [24].
- Date: 1988.
- Type: Parallel.
- Complexity: $O\left(n^{2/3}\log^3 n \log nU\right)$.
- Processors: $O\left(n^3/\log n\right)$.
- Model: PRAM and CRCW.
- Restriction: integrality of cost function $c : E \rightarrow \mathcal{Z}$.

This algorithm finds a minimum cost perfect matching in G. It is a parallel algorithm which uses scaling and approximation. The constant used in the approximation is $\left\lceil \frac{n+1}{2} \right\rceil$.

3.6 Scaling and Approximation Parallel Algorithm of Gabow and Tarjan

- Authors: Gabow and Tarjan [20].
- Date: 1988.
- Type: Parallel.
- Complexity: $O\left(\left(mn^{1/2}/p\right)\log p \log nU\right)$.
- Processors: p, where p is at most $\frac{m}{n^{1/2}\log^2 n}$.
- Model: PRAM and EREW.
- Restriction: integrality of cost function, that is, $c : E \rightarrow \mathcal{Z}$.

This algorithm finds a minimum cost perfect matching in G. It is a parallel algorithm which uses scaling and approximation. The constant used in the approximation is $\left\lceil \frac{n+1}{2} \right\rceil + n$.

4 General Graphs

Tables 3 and 4 describe briefly some of the most important algorithms for solving the general problem, in which the graph is not necessarily bipartite. The algorithm of Mulmuley, Vazirani and Vazirani [33], showed in Table 3, is probabilistic (Monte Carlo).

Sequential Algorithms			
Date	Authors	Ref.	Complexity
1965	Edmonds	[11]	$O(mn^2)$
1975	Even and Kariv	[14]	$O(\min\{n^{5/2}, mn^{1/2}\log n\})$
1976	Gabow	[18]	$O(n^3)$
1980	Micali and Vazirani	[32]	$O(mn^{1/2})$
1990	Blum	[7]	$O(mn^{1/2})$

Parallel Algorithm				
Date	Authors	Ref.	Complexity	Processors
1987	Mulmuley, Vazirani and Vazirani	[33]	$O(\log^2 n)$	$O(n^{7/2}m)$

Table 3: Non weighted graphs.

Sequential Algorithms			
Date	Authors	Ref.	Complexity
1973	Edmonds and Johnson	[12]	$O(mn^2U)$
1982	Galil, Micali and Gabow	[22]	$O(mn\log n)$
1990	Gabow	[17]	$O(mn + n^2\log n)$

Table 4: Weighted graphs.

References

[1] A. Aho, J. Hopcroft, and J. Ullman. *The Design And Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, Inc., Massachusetts, 1974.

[2] S. Akl. *The Design and Analysis of Parallel Algorithms*. Prentice-Hall International, Inc., New Jersey, 1989.

[3] H. Alt, N. Blum, K. Mehlhorn, and M. Paul. Computing a maximum cardinality matching in a bipartite graph in time $o(n^{1.5}\sqrt{m/\log n})$. *Inf. Proc. Letters*, 37:237–240, Feb 1991.

[4] H. Baier S. Matching algortihms in bipartite graphs. Master's thesis, UNICAMP, Campinas-SP, Brazil, Mar 1993.

[5] C. Berge. Two theorems in graph theory. In *Proc. Nat. Acad. Sci. USA*, pages 842–844, 1957.

[6] D. P. Bertsekas. A new algorithm for the assignment problem. *Math. Prog.*, 21:152–171, 1981.

[7] N. Blum. A new approach to maximum matching in general graphs. Technical Report 8546-CS, Universitat Bonn, Feb 1990.

[8] J. A. Bondy and U. S. R. Murty. *Graph Theory With Applications*. MacMillan, New York, 1976.

[9] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[10] E. A. Dinic. Algorithm for solution of a problem of maximum flow in a network with power estimation. *Soviet Math. Dokl.*, 11(5):1277–1280, Sep 1970.

[11] J. Edmonds. Path, trees and flowers. *Canadian J. Math.*, 17:449–467, 1965.

[12] J. Edmonds and E. J. Johnson. Euler tours and the chinese postman. *Math. Prog.*, 5:88–124, 1973.

[13] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. of the Assoc. for Comput. Mach.*, 19(2):248–264, Apr 1972.

[14] S. Even and Kariv. An $O(n^{2.5})$ - algorithm for maximum matching in general graphs. In *16th FOCS*, pages 100–112, 1975.

[15] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM J. Comput.*, 4(4):507–518, Dec 1975.

[16] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. In *25th FOCS*, pages 338–346, 1984.

[17] H. Gabow. Data structures for weigthed matching and nearest common ancestors with linking. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 434–443, 1990.

[18] H. N. Gabow. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *J. of the Assoc. for Comput. Mach.*, 23(2):221–234, Apr 1976.

[19] H. N. Gabow. Scaling algorithms for network problems. *J. of Comput. and Syst. Sci.*, 31(2):148–168, Oct 1985.

[20] H. N. Gabow and R. E. Tarjan. Almost-optimal speed-ups of algorithms for matching and related problems. In *20th STOC*, pages 514–527, 1988.

[21] H. N. Gabow and R. E. Tarjan. Faster scaling algorithms for network problems. *SIAM J. Comput.*, 18(5):1013–1036, Oct 1989.

[22] Z. Galil, S. Micali, and H. Gabow. Priority queues with variable priority and an $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs. In *23rd FOCS*, pages 255–261, 1982.

[23] A. Goldberg, S. Plotkin, D. Shmoys, and E. Tardos. Using interior-point methods for fast parallel algorithms for bipartite matching and related problems. *SIAM J. Comput.*, 21(1):140–150, Feb 1992.

[24] A. V. Goldberg, S. A. Plotkin, and P. M. Vaidya. Sublinear-time parallel algorithms for matching and related problems. In *29th FOCS*, pages 174–185, 1988.

[25] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. of the Assoc. for Comput. Mach.*, 35(4):921–940, Oct 1988.

[26] J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, Dec 1973.

[27] T. Kim and K. Y. Chwa. An $O(n \log n \log\log n)$ parallel maximum matching algorithm for bipartite graphs. *Inf. Proc. Letters*, 24(1):15–17, Jan 1987.

[28] D. Knuth. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms.* Addison-Wesley Publishing Company, Inc., Massachusetts, 1973.

[29] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart.*, 2:83–97, 1955.

[30] H. W. Kuhn. Variants of the Hungarian method for assignment problems. *Naval Res. Logist. Quart.*, 3:253–258, 1956.

[31] U. Manber. *Introduction To Algorithms - A Creative Approach.* Addison-Wesley Publishing Company, Inc., Massachusetts, 1989.

[32] S. Micali and V. Vazirani. An $O(|V|^{0.5} \cdot |E|)$ algorithm for finding maximum matchings in general graphs. In *21th FOCS*, pages 17–27, 1980.

[33] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. In *19th STOC*, pages 345–354, 1987.

[34] J. Munkres. Algorithms for the assignment and transportation problems. *J. Soc. Indust. Appl. Math.*, 5:32–38, 1957.

Relatórios Técnicos – 1992

01/92 **Applications of Finite Automata Representing Large Vocabularies,** *C. L. Lucchesi, T. Kowaltowski*

02/92 **Point Set Pattern Matching in d-Dimensions,** *P. J. de Rezende, D. T. Lee*

03/92 **On the Irrelevance of Edge Orientations on the Acyclic Directed Two Disjoint Paths Problem,** *C. L. Lucchesi, M. C. M. T. Giglio*

04/92 **A Note on Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams,** *W. Jacometti*

05/92 **An (l, u)-Transversal Theorem for Bipartite Graphs,** *C. L. Lucchesi, D. H. Younger*

06/92 **Implementing Integrity Control in Active Databases,** *C. B. Medeiros, M. J. Andrade*

07/92 **New Experimental Results For Bipartite Matching,** *J. C. Setubal*

08/92 **Maintaining Integrity Constraints across Versions in a Database,** *C. B. Medeiros, G. Jomier, W. Cellary*

09/92 **On Clique-Complete Graphs,** *C. L. Lucchesi, C. P. Mello, J. Szwarcfiter*

10/92 **Examples of Informal but Rigorous Correctness Proofs for Tree Traversing Algorithms,** *T. Kowaltowski*

11/92 **Debugging Aids for Statechart-Based Systems,** *V. G. S. Elias, H. Liesenberg*

12/92 **Browsing and Querying in Object-Oriented Databases,** *J. L. de Oliveira, R. de O. Anido*

Relatórios Técnicos – 1993

01/93 **Transforming Statecharts into Reactive Systems,** *A. G. Figueiredo Filho, H. Liesenberg*

02/93 **The Hierarchical Ring Protocol: An Efficient Scheme for Reading Replicated Data,** *N. das C. Mendonça, R. de O. Anido*

03/93 **Matching Algorithms for Bipartite Graphs,** *H. A. Baier Saip, C. L. Lucchesi*

Departamento de Ciência da Computação — IMECC
Caixa Postal 6065
Universidade Estadual de Campinas
13081-970 – Campinas – SP
BRASIL

`reltec@dcc.unicamp.br`