Contributions of Jayme Luiz Szwarcfiter to Graph Theory and Computer Science

Cláudio Leonardo Lucchesi *
Institute of Computing, UNICAMP,
C. P. 6176, 13083-970 Campinas, SP, Brazil
lucchesi@ic.unicamp.br

Abstract

This is an account of Jayme's contributions to Graph Theory and Computer Science. Due to restrictions in length, it is not possible to provide an in-depth coverage of every aspect of Jayme's extensive scientific activities. Thus, I describe in detail only some of his principal contributions, touch upon some and merely list the other articles.

I found it easier to write the article in the first person, as though it is an account of a previously given lecture.

Mucho gusto en conocerlo

According to Marisa Gutierrez, this is the way people say "nice to meet you" in Argentina.

The first time I heard of Jayme was back in 1974. At that time I was a graduate student at the University of Waterloo, in Canada. In those days I used to become quite upset with the general lack of knowledge about Brazil. The standard joke among the Brazilian students was that every foreigner thought that Buenos Aires was the capital of our country.

A friend of mine came by my office and showed me a copy of one of Jayme's articles, a joint paper with Donald Knuth [30]. A very nice paper. However, my friend's intention was not to contribute to my education, but to irritate me with the title page of the article. It showed Jayme's name, followed by the affiliation, and I quote: *Universidade Federal do Rio de Janeiro, Argentina.*

I must say that the elegance and simplicity of the paper was worth the irritation. There was also the added pleasure of seeing a Brazilian publishing a paper with Knuth. As you know, Knuth is one of the most illustrious researchers in Computer Science. Almost 20 years later, in 1992, Knuth wrote a book entitled *Literate Programming* [29]. Chapter 3 of this book contains a reproduction of the original article.

This paper was one of Jayme's first contributions to make Brazil better known abroad. Since then, Jayme has produced a significant number of papers, some of which, like the paper mentioned above, are milestones in the history of Brazilian science. Jayme's activity has also helped to foster the integration of the scientific community in our country. He also has an important role in the integration of the Latin American scientific communities, through collaboration with many researchers from different countries. And that includes colleagues from Argentina, of course.

Exercise 1 *Determine the list of Latin American researchers who are coauthors of Knuth.*

A Universidade do Brasil

As many of you know, the *Universidade Federal do Rio de Janeiro* is actually the *Universidade do Brasil*. Jayme participated in the creation of the three departments in that University where most of the research in Computer Science and Graph Theory takes place: (i) the *Núcleo de Computação Eletrônica*, created in 1970, (ii) the Department of Computer Science, created in 1971, part of the Institute of Mathematics, and (iii) the Systems Engineering and Computer Science Program, created in 1971.

Jayme taught the first Computer Science undergraduate course at the Universidade do Brasil, "Introduction to Computing", to undergradute Physics students, back in 1971.

He has supervised 17 Master's students and 16 Ph.D. students, from several regions of Brazil and Latin America, who are today university professors in these regions.

*Supported by a grant from CNPq. Supported by the program PRONEX/CNPq (664107/1997-4).

Jayme's influence extends to more than one generation of students and was enhanced by two textbooks, one on Data Structures [55], a joint work with Lilian Markenson, the other on Graphs and Algorithms [42].

He has also written three comprehensive surveys: (i) a joint work with Mónica Villanueva, a survey on chordal graphs [60], (ii) a joint work with Celina Figueiredo, a survey on matchings in graphs [20], and (iii) a survey on clique graphs [50].

Theorems × algorithms

Jayme's work seems to be characterized by the search of mathematical properties that help in determining efficient algorithms for solving problems, or that help in showing that the problem is probably computationally intractable. That is clearly the case of most of his papers, particularly the aforementioned paper with Knuth [30].

There are occasions, however, in which the opposite direction seems to have been followed. There is a very nice little note that he published [46], on the closure of a graph, as defined by Bondy and Chvátal in [9], in the search of sufficient conditions for a graph to be Hamiltonian. Whenever one reads that note, it immediately comes to mind the similarity involving the technique used by the closure algorithm and the technique used to enumerate the topological sorting arrangements in [30]. Indeed, Jayme calls the reader's attention to that similarity at the end of the note, and takes advantage of that similarity in order to define a duality that relates mathematically the two situations, so obviously similar from the algorithmic point of view.

Having studied the many facets of his work, it is unclear to me whether Jayme views his mathematical activity as a vehicle for finding efficient algorithms or he uses the search for efficient algorithms as a means for discovering nice mathematical properties! It might be both. But, whatever his guiding principle might be, it is clear that these two aspects of his work enrich each other.

Let us examine now some of Jayme's work. The standard definitions in graph theory and in algorithm complexity may be found in classical books such as [10] and [21].

Topological sorting

Let us now take a look at the joint work with Knuth [30] on topological sorting arrangements, mentioned at the beginning of this lecture.

A *topological sorting* of a directed graph G is an enumeration $T := (v_1, v_2, \cdots, v_n)$ of the n vertices of G such that for each edge (v_i, v_j) of G, $i < j$.

The authors give a very nice backtracking algorithm for generating all topological sortings of a directed graph. There are two fundamental ideas behind the algorithm. The first observation is that any subsequence $S := (v_1, v_2, \cdots, v_r)$ $(0 \leq r < n)$ of consecutive terms of T is a topological sorting of $G[S]$, the subgraph of G spanned by the vertices in S. The second observation is that vertex v_{r+1} must be a source of $G - S$, the subgraph of G spanned by the vertices not in S; that is, v_{r+1} must have in-degree zero in $G - S$.

With these two observations in mind, it is easy to understand the algorithm for enumerating all topological sortings of G. The algorithm is recursive. It receives (i) a sequence $S := (v_1, v_2, \cdots, v_r)$ $(0 \leq r < n)$ that is a topological sorting of $G[S]$, (ii) a vector d on n entries such that for each vertex v of G, the entry $d[v]$ is equal to zero if v lies in S, otherwise $d[v]$ is the in-degree of v in $G - S$, and (iii) a linked list L of vertices of $G - S$ that have in-degree zero in $G - S$.

The initial call to this algorithm passes as arguments the empty sequence S, the vector d of in-degrees of each vertex in G, and the list of sources of G.

The algorithm then recursively extends the sequence received, in all possible ways. For this, it must extend the sequence with a source of $G - S$: it does that by extracting from the list L its first element, say v. It is then easy to update vector d, by subtracting one from $d[w]$, for each vertex w such that (v, w) is an edge of G. For each such w, if the new value of $d[w]$ is zero then w is added at the end of list L. The algorithm is then ready to call itself recursively. On the return from the recursive call, vector d is restored to its original value, by adding one to $d(w)$, for each edge (v, w) in G. Each vertex added at the end of L, in the set of sources of $G - S - v$ that are not sources of $G - S$, is removed from L. Vertex v is removed from S and added back to the end of list L. The algorithm then proceeds with the next source of $G - S$, until the complete list L is scanned and the first vertex v is again the first vertex in the list. At this point, the algorithm returns.

Whenever the algorithm completes a sequence of length n, it prints the sequence as output. So after the return to the first call to the algorithm, the complete enumeration of all topological sortings has been printed.

Every effort is made in order to avoid any unnecessary searching in the graph. The list L is a very efficient way of having the sources ready for use, with-

out the need to scan vector d. The first vertex v in L is added to S and removed from L. This requires constant time. Vector d is then updated in time linear with respect to the out-degree of v in G. So too is list L, which must now contain the sources of $G - S - v$. On the return from the recursive call, a reverse procedure, also linear in the out-degree of v, restores the initial values for L and d, except that v is now at the end of L.

It is thus easy to see that the time required for each topological sorting is linear on the size of G. Therefore, the algorithm has complexity $O((m+n)\alpha)$, where m is the number of edges of G and α the number of distinct topological sorting arrangements of G.

The authors even have a very elegant way of avoiding repeating the printing of the common prefix S, for each topological sorting arrangement that starts with S. For example, in an acyclic directed graph on 5 vertices, the output could look like this:

1	2	3	4	5
		4	3	5
	3	2	4	5
2	1	3	4	5
		4	3	5
	3	1	4	5
3	1	2	4	5
	2	1	4	5

By the way:

Exercise 2 *Reconstruct graph G, given the topological sortings of G.*

The closure of a graph

Adrian Bondy and Vašek Chvátal proved the following nice result [9]:

Theorem 1 *Let G be a simple graph on n vertices, v and w two nonadjacent vertices of G such that the sum $d(v) + d(w)$ of the degrees of v and w in G is at least n. Then, G is Hamiltonian if and only if $G+vw$ is Hamiltonian.*

The proof of this result is an application of the pigeonhole principle. If G is Hamiltonian then clearly so is $G + vw$. For the converse, get a Hamiltonian circuit C of $G + vw$. If C does not use edge vw, then it is a Hamiltonian circuit of G itself. If C uses edge vw, then $C - vw$ has a path $P := (v = v_1, v_2, \cdots, v_n = w)$ that is a Hamiltonian path in G; moreover, by the pigeonhole principle, there exists i such that $1 < i < n - 1$, v_i is adjacent to w and v_{i+1} is adjacent to v. In that case, $(v = v_1, v_{i+1}, v_{i+2}, \cdots, v_n = w, v_i, v_{i-1}, \cdots, v_1)$ is a Hamiltonian circuit of G.

We may repeatedly add edges satisfying the inequality stated in the assertion of the Theorem, until addition is no longer possible, thereby getting a sequence of graphs $G = G_0, G_1, \cdots, G_r = c(G)$ such that either each graph in the sequence is Hamiltonian or no graph in the sequence is. Moreover, although the sequence is not unique, the last graph $c(G)$ of the sequence, called the *closure* of G, is unique. If $c(G)$ is the complete graph, clearly a Hamiltonian graph, then G is also Hamiltonian.

It is not difficult to have a polynomial algorithm of complexity $O(n^4)$ steps, which computes $c(G)$, given G: for each pair $\{v, w\}$ of vertices of G, determine whether v and w are nonadjacent and $d(v)+d(w) \geq n$. If v and w pass both tests, add vw to G and repeat the algorithm. This verification may be done in time $O(1)$ for each pair, whence the determination of the next graph in the sequence takes $O(n^2)$. The length of the sequence is $O(n^2)$, therefore the algorithm has complexity $O(n^4)$.

Let us now see how Jayme was able to reduce the complexity to $O(n^3)$ in [46], by making the computation of each new graph in the sequence linear. In the *initialization phase*, compute the adjacency matrix A of G, and the *deficiency* matrix D, an $n \times n$ matrix such that for each pair $\{v, w\}$ of distinct nonadjacent vertices of G, $D[v, w] := \max\{0, n - (d[v] + d[w])\}$. Compute also a list L of pairs $\{v, w\}$ of nonadjacent vertices v and w of G such that $D[v, w] = 0$. All this can be clearly done in time $O(n^2)$.

Let me now describe one step of the *iteration phase*, in which a new graph in the sequence is obtained in time $O(n)$, or, alternatively, the algorithm concludes that $G = c(G)$ in time $O(1)$.

If the list L is empty then $G = c(G)$. If L is nonempty, remove from it a pair $\{v, w\}$, add vw to G. Update A in time $O(1)$. For each vertex x of $G-v-w$ such that x is not adjacent to v and $D[v, x] > 0$, subtract one from $D[v, x]$. If $D[v, x]$ becomes equal to zero, add the pair $\{v, x\}$ to the list L. Repeat this, with w in the role of v. One step of the iteration phase can certainly be done in time $O(n)$. Therefore, the computation of $c(G)$ can be done in time $O(n^3)$.

The similarity between the idea of having a deficiency matrix dynamically computed in this algorithm and the idea of having an in-degree vector dynamically computed in the algorithm of topological sorting enumeration prompted Jayme to observe a duality involving the two problems, which I will describe in a

moment.

The *line graph* $L(G)$ of G is the intersection graph of the edges of G. That is, the set of vertices of $L(G)$ is the set of edges of G and two vertices e_1 and e_2 of $L(G)$ are adjacent if and only if edges e_1 and e_2 are adjacent in G. Jayme expressed the duality in the following statement:

Theorem 2 *Let G be an undirected graph. Then $c(G)$ is complete if and only if there is an acyclic orientation of the line graph $L(\overline{G})$ of the complement $\overline{G}$ of G in which the in-degree of each vertex $\{v,w\}$ of $L(\overline{G})$ is at least the deficiency $D[v,w]$ of the pair $\{v,w\}$ in G.*

Hamiltonian paths in grid graphs

Speaking of Hamiltonian paths and circuits, let me talk now about another very important result of Jayme, in a joint work with Alon Itai and Christos Papadimitriou [26].

Let G^∞ denote the infinite graph whose vertices are the points in the plane having integral coordinates and in which two vertices are adjacent if and only if the Euclidean distance between them is equal to one.

A *grid graph* is a finite vertex-induced subgraph of G^∞. Thus, a grid graph is completely specified by its set of vertices. For each vertex v, let v_x and v_y denote the coordinates of v. The *parity* of v is the parity of the sum $v_x + v_y$ of its coordinates. Thus, v is *even* if $v_x + v_y$ is even, and is *odd* otherwise. All grid graphs are bipartite, with the edges connecting an even vertex to an odd vertex.

Denote by $R(m,n)$ the grid graph whose set of vertices is $\{v : 1 \leq v_x \leq m, \quad 1 \leq v_y \leq n\}$. A *rectangular graph* G is a grid graph that is isomorphic to $R(m,n)$, for some integers m and n, called the *dimensions* of G. Note that the dimensions of a rectangular graph completely specify the graph, up to isomorphism.

Let s and t be distinct vertices of a grid graph G. The *Hamilton path problem* (G,s,t) consists in determining whether there is in G a Hamiltonian path from s to t.

The problem of determining whether or not an instance (G,s,t) of a Hamilton path problem has a solution is NP-complete. This result was proved by Jayme and his coauthors in the paper I just mentioned. This result is the most important known NP-complete restriction of the Traveling Salesman Problem. Indeed, the book *The Traveling Salesman Problem* [31], edited by Lawler et al., contains a chapter written by D. S. Johnson and Papadimitriou where the authors mention the importance of the NP-completeness result and transcribe its proof as originally published in [26].

The proof of the NP-completeness is too technical to be presented here. But it uses grid graphs G that have "holes", that is, $G^\infty - G$ is not connected. In the paper, the authors indicate that it is not known whether or not the Hamilton path problem is NP-complete for grid graphs without holes.

The paper also presents a very nice positive result for rectangular graphs: for every rectangular graph $G := R(m,n)$, there is an algorithm of complexity $O(mn)$ that either determines that the instance (G,s,t) of the Hamilton path problem has no solution or determines a solution. As usual, the algorithm is a by-product of the proof of theorems. I will not present the details here, but let me at least tell you the characterization of the instances that do have a solution.

Consider a rectangular graph G of dimensions m and n. We have seen that G is bipartite. If m and n are both odd, then the number mn of the vertices of G is odd, one of the parts of the bipartition has one vertex more than the other part. In that case, every Hamiltonian path in G must start and end in vertices that lie in the majoritarian part. On the other hand, if at least one of m and n is even, then G has an even number of vertices, and each part of the bipartition of G has precisely half of the vertices of G. In that case, every Hamiltonian path has its origin and terminus in distinct parts of the bipartition.

Without loss of generality, let us assume that $G = R(m,n)$, $m \geq n$. A necessary condition for (G,s,t) to have a solution is that the parities of s and t coincide if and only if (i) mn is odd and (ii) each of s and t is even. The authors call this the *color compatibility condition* of the (G,s,t) problem.

The color compatibility condition is not sufficient in general. It is sufficient for "large" rectangles, in which both dimensions are at least four. It is also sufficient for rectangles in which m is odd and $n = 3$. (Under the hypothesis that $m \geq n$.)

The problems appear when either $n \leq 2$ or when $n = 3$ and m is even. If $n = 1$ then the problem has a solution if and only if

$$\{s_x, t_x\} = \{1, m\}. \tag{1}$$

If $n = 2$ then the problem has a solution if and only if

$$s_x = t_x \in \{1, m\}. \tag{2}$$

The description of the necessary and sufficient condition for the case in which $n = 3$ and m is even is more elaborate: we have seen that the color compatibility condition implies that in this case one of s and t must

be even, the other odd. Adjust notation so that s is odd and t is even. The additional condition is then the following:

Under the hypothesis that m is even, $n = 3$, s is odd and t is even, the instance (G, s, t) has a solution if and only if

$$s_x \geq t_x - 1, \quad \text{with equality only if } t_y \neq 2. \qquad (3)$$

In sum, the color compatibility condition is necessary. Assume that $m \geq n$. The condition is sufficient for the cases in which $n \geq 4$. It is also sufficient for the case in which $n = 3$ and m is odd. For the cases in which (i) $n = 1$, or (ii) $n = 2$, or (iii) $n = 3$ and m is even, the condition has the additional requirement of (1), (2) and (3) , respectively.

As I said, there are too many details to describe the proof and the algorithm here. The following exercises may prompt you to try to prove the result.

Exercise 3 *Assume that $s_x, t_x \leq m - 2$. Let $G' := R(m - 2, n)$. Show that if the problem (G', s, t) has a solution then so too does (G, s, t). Prove also that if $n = 3$ and (G, s, t) has a solution then (G', s, t) has a solution.*

Exercise 4 *Prove that the additional conditions* (1), (2) *and* (3) *are necessary for small rectangles.*

Optimal multiway search trees

The most popular data structure used in large databases is the B-tree, or a variation thereof. B-trees were invented by Bayer and McCreight in 1971 [5], in order to minimize the number of input operations in secondary storage. In fact, in the worst case this number is the height of the tree, and a B-tree may hold millions of keys with a height equal to three. This means that in order to find a key in a database of that size it suffices to perform at most three input operations from disk.

I am now going to describe a solution found by Jayme to an open problem posed by McCreight himself. That solution was published in 1984 [43]. I shall also describe some related work published in that same paper. For this, we need some definitions.

Let $E := (e_1, \cdots, e_n)$ be a sequence of elements called *keys*. Associated with each key e_i there is its *size* s_i and its *value*, an integer y_i. We assume that $y_1 < y_2 < \cdots < y_n$.

A *multiway search tree for* E is an ordered rooted tree T such that each key e in E is assigned to exactly one node $x(e)$ of T, while each node x keeps a subset $E(x)$ of keys that satisfies the following properties:

(i) $E(x)$ is empty if and only if x is a leaf of T.

(ii) Each non-leaf x of T has exactly $|E(x)| + 1$ sons.

(iii) If y is the k-th son of x in the ordering of T and e_j is an arbitrary key of $E(y)$ then exactly $k - 1$ keys e_i of $E(x)$ satisfy the inequality $y_i < y_j$.

The $n + 1$ leaves of T are called *gaps* and denoted $g_0, g_1, \cdots, g_n$. Each gap g_i corresponds to the interval $I_i := \{y : y_i < y < y_{i+1}\}$, where $y_0 := -\infty$ and $y_{n+1} := +\infty$.

Let x be a node of T. The *size* $s(x)$ of x is the sum of the sizes of the keys of $E(x)$. The *height* $h(x)$ of x is the number of nodes in the path from the root to x. The *height* of T is the maximum height of the non-leaves of T. The *space* of T is the number of non-leaf nodes.

Let L be any integer. We say that T has *page limit* L if $s(x) \leq L$, for each node x of T.

Let L_1 and L_2 be integers such that $0 < L_1 \leq L_2$. A *weak B-tree of limits* (L_1, L_2) is a multiway search tree T of page limit L_2 such that:

(i) $s(x) \geq L_1$, for each non-leaf node x of T distinct from the root of T.

(ii) All leaves of T have the same height.

A *B-tree* is a weak B-tree of limits $(\lceil L_2/2 \rceil, L_2)$.

Suppose that there are associated with each key e_i a probability p_i and with each interval I_i a probability q_i such that $\sum_{i=1}^{n} p_i + \sum_{i=0}^{n} q_i = 1$.

In a search for a value y, p_i is the probability that $y = y_i$ and q_i is the probability that $y \in I_i$. The *cost* for determining whether a given value y is the value associated with some key is equal to $h(x(e_i))$ if $y = y_i$ for some key e_i; otherwise, the cost is $h(g_i) - 1$, where $y \in I_i$. As you see, the cost is the number of input operations from disk that are necessary in order to have the answer to the problem "does there exist a key e_i such that $y = y_i$?". Taking into account the probabilities mentioned earlier, we deduce that the *average search cost* of T, or simply, the *cost* of T, is the sum

$$\sum_{i=1}^{n} p_i h(x(e_i)) + \sum_{i=0}^{n} q_i (h(g_i) - 1).$$

One of the problems is to minimize the cost of a tree, given the limit L, in the generic case of multiway

trees, or the limits (L_1, L_2), in the case of a weak B-tree.

There is a particular case of the problem that is solvable in polynomial time, using standard dynamic programming techniques. It is the case in which the sizes of the keys are equal (see [22, 25]).

In the paper I mentioned, Jayme showed that the general problem of optimizing the cost of a tree, given the keys, the probabilities and the page limit, is NP-hard. More precisely, he proved the following result:

Theorem 3 *Deciding whether there exists a multiway search tree for E having limit L and cost at most C is NP-complete. It remains so even if all gap probabilities are equal to zero and each key probability is proportional to the size of the key.*

He also proved another similar result in that paper, which implies the previous one:

Theorem 4 *Deciding whether there exists a weak B-tree for E having limits (L_1, L_2) and cost at most C is NP-complete. It remains so even if all gap probabilities are equal to zero and each key probability is proportional to the size of the key.*

Both proofs use a reduction of the partition problem, an NP-complete problem [21]. Given a set $A := \{a_1, a_2, \cdots, a_n\}$ of integers, the *partition problem* consists in determining whether or not there exists a partition of A in two blocks such that the sums of the elements in each block coincide.

Jayme also gave pseudo-polynomial algorithms for the problems considered in Theorems 3 and 4, using dynamic programming. His algorithm determines a cost optimal multiway tree in time $O(n^3 L)$ and a cost optimal weak B-tree in time $O(n^3 L_2)$. He also gave polynomial algorithms for minimizing either the height or the space of a multiway tree.

Let us get now to the problem posed by McCreight and solved by Jayme. The problem is the following: given a sequence E, and limits (L_1, L_2), give an efficient algorithm for determining a weak B-tree that has height two, precisely M keys in the root and minimum root size. As usual, we are assuming that $0 < L_1 \leq L_2$. We are also assuming that $\sum s_i > L_1$, otherwise there is no reason for a tree of height two.

Let us first show how Jayme solved a simpler problem: solve McCreight's problem relaxing the condition on the number of keys in the root. That is, the root may have any number of keys, subject to the upper limit condition on its size.

Here is a description of the very elegant solution for this simpler problem. Define a complete, acyclic directed graph D on the set $\{v_0, v_1, \cdots, v_{n+1}\}$ of vertices, in which for every pair (i, j) such that $0 \leq i < j \leq n+1$, an edge leaves vertex v_i, enters vertex v_j and the cost of that edge is $d[i, j]$, defined as follows:

$$d[i,j] := \begin{cases} s_j, & \text{if } L_1 \leq \sum_{i<k<j} s_k \leq L_2 \\ \infty, & \text{otherwise,} \end{cases}$$

where $s_{n+1} := 0$.

Let the *cost* of a directed path P be the sum of the costs of its edges. For each directed path P from v_0 to v_{n+1} of cost at most L_2, the set of vertices $V(P)$ gives a (possibly not optimal) solution to the simplified problem: the set of keys in the root is precisely the set $\{e_i : v_i \in V(P) - v_0 - v_{n+1}\}$ and the size of the root is the cost of P. If, in addition, the cost of P is minimum, then the tree has minimum root size.

Thus, Jayme reduced the simplified version of the problem to that of finding in D a path of minimum cost from v_0 to v_{n+1}. A straightforward implementation of a minimum cost path algorithm in D takes time $O(n^2)$. Well, Jayme discovered a structure in the costs in the graph that allowed him to decrease the complexity of the algorithm to $O(n \log n)$.

Let us now see how to solve the original problem, in which we would like to have not only minimum root size, but also precisely M keys in the root. The solution described here seems simpler than that originally given by Jayme in the paper, but is somewhat equivalent. In terms of the complete acyclic directed graph D defined above, the problem is equivalent to that of finding in D, among all directed paths of length $M+1$ from v_0 to v_{n+1}, one that has minimum cost.

So it is easy to solve: define an $(n+1) \times (n+1)$ matrix A, initialized with ∞ everywhere, except at entry $A[0,0]$, which is initialized with zero. Then,

```
for r = 1, ··· , M + 1
  for i = 0, ··· , n
    for j = i + 1, ··· , n + 1
      let A[j,r] := min{A[j,r], A[i,r − 1] + d[i,j]}.
```

Of course, $A[i, r]$ is the cost of the minimum cost path from v_0 to v_i that has length r. In particular, if $A[n+1, M+1] \leq L_2$, then $A[n+1, M+1]$ is the size of the root of a weak B-tree for E with limits (L_1, L_2) having precisely M keys in the root and minimum root size. A straightforward implementation of this algorithm takes time $O(n^2 M)$. Jayme was able to lower the complexity to $O(nM \log n)$.

Iterated clique graphs with increasing diameters

Let us now examine a result proved by Jayme and Claudson Bornstein [12]. The result answered a question that had been open for 12 years, it was posed in two articles and also in the book *Graph Dynamics*, by E. Prisner [35].

Let G be a graph. A *clique* of G is a set of pairwise adjacent vertices of G. A clique of G is *maximal* if it is not a proper subset of some other clique of G. The *clique graph* $K(G)$ of G is the intersection graph of the family of maximal cliques of G. That is, the set of vertices of $K(G)$ is the set of maximal cliques of G, and two vertices k_1 and k_2 of $K(G)$ are adjacent if and only if the cliques k_1 and k_2 of G intersect.

For each nonnegative integer i, define $K^i(G) := G$ if $i = 0$ and $K^i(G) := K(K^{i-1}(G))$, if $i > 0$. Likewise, for each nonnegative integer i, define $L^i(G) := G$ if $i = 0$ and $L^i(G) := L(L^{i-1}(G))$, if $i > 0$, where $L(G)$ denotes the line graph of G. For each vertex k of $K(G)$, and each positive integer i, the *i-th inverse image* $K^{-i}(k)$ of k is the clique k of G if $i = 1$, otherwise it is $\cup_{v \in K^{-1}(k)} K^{-(i-1)}(v)$.

The *diameter* $\mathrm{diam}(G)$ of G is the maximum distance between any two vertices of G. That is, if $d(v, w)$ denotes the distance between v and w in G, then $\mathrm{diam}(G) := \max\{d(v, w) : v, w \in V(G)\}$. A pair of vertices of G is *diametrical* if their distance is equal to the diameter of G.

Hedman [24] showed that

$$\mathrm{diam}(G) - 1 \leq \mathrm{diam}(K(G)) \leq \mathrm{diam}(G) + 1.$$

The proof of the above inequalities is not difficult. In fact, if you want to warm up for this subject, here is a nice problem:

Exercise 5 *Let G be a connected graph with at least one edge. Let F be a family of nonnull cliques of G such that for every edge e of G, both ends of e lie in some member of F. Let I denote the intersection graph of G induced by F, that is the set of vertices of I is F, two vertices f_1 and f_2 of I are adjacent if and only if the cliques f_1 and f_2 of G intersect. Prove that*

$$\mathrm{diam}(G) - 1 \leq \mathrm{diam}(I) \leq \mathrm{diam}(G) + 1.$$

Prove also that $\mathrm{diam}(I) = \mathrm{diam}(G) + 1$ *if and only if there exist two cliques k_1 and k_2 in F such that for each vertex v_1 of k_1 and each vertex v_2 of k_2, vertices v_1 and v_2 are diametrical in G.*

Note that the inequality proved by Hedman is a particular case of Exercise 5. It suffices to define F to be the family of maximal cliques of G: graph I will be the clique graph of G in that case. Another application of Exercise 5 is to define F to be set of pairs of vertices of G that are adjacent: in that case, graph I will be the line graph of G. The inequality for line graphs was proved by Knor et al. [28]

Hedman also described a family of graphs G for which $\mathrm{diam}(K(G)) = \mathrm{diam}(G)+1$, and asked if graphs G exist with $\mathrm{diam}(K^i(G)) = \mathrm{diam}(G) + i$ for each positive integer $i \geq 2$.

The existence of such graphs G for $i = 2$ was established by R. Balakrishnan and P. Paulraja [2] and independently by C. Peyrat, D. F. Rall and P. J. Slater [34], who also proved the existence in the cases $i = 3$ and $i = 4$.

Jayme and Claudson were able to prove the existence of such a G for all positive integers i. In order to do that, they defined a graph $H(d, G)$, where d is a positive integer and G is a graph. Graph $H(d, G)$ was defined as follows: (i) take two copies G' and G'' of G; (ii) take a new vertex v, and join it to each vertex of each of G' and G'' by a path of length d. Then, they showed the following result:

Theorem 5 *Let G be a graph whose diameter is at most $2d$, v_1 and v_2 two vertices of $K^i(H(d, G))$. If the inverse i-th images of v_1 and v_2 are contained in G' and G'', respectively, then v_1 and v_2 are diametrical with distance* $\mathrm{diam}(H(d, G)) + i$.

Thus, Hedman's problem was reduced to finding a suitable graph G. They were able to use a relatively complicated argument to show that $L^i(K_n)$ is a good choice. More specifically, they proved the following result:

Theorem 6 *Let i, d and n be positive integers such that $2d > i+2$, $n \geq 4$ and $i \leq \lfloor n/2 \rfloor - 1$. Let G be the graph $H(d, L^i(K_n))$. Then,*

$$\mathrm{diam}(K^{i+1}(G)) = \mathrm{diam}(G) + i + 1.$$

With the above theorem, they solved completely the question, which had remained open for 12 years.

Clique graphs of directed path graphs and of rooted path graphs

Let us now examine another important contribution of Jayme. It is a joint work with Erich Prisner [36]. The paper was selected to be part of the

Editors' Choice 1999 of the *Discrete Applied Mathematics*, an indication that it was one the best articles published by the periodical in 1999.

In that paper, the authors characterize the clique graphs of two families of graphs, the directed path graphs and the rooted path graphs.

A *directed path graph* (or a *DV graph*) is the intersection graph of the family of directed paths of a directed tree. A *dually directed path graph* (or *dually DV graph*) is a graph G that admits a spanning directed tree T such that, for each edge (v, w) of G, T contains a directed $v-w$ path or a directed $w-v$ path whose vertices form a clique in G.

In order to describe their characterization of a dually DV graph, we need some definitions.

A family $\mathcal{F}$ of sets has the *Helly property* if, for every nonnull subcolletion $\mathcal{G}$ of $\mathcal{F}$, either $\mathcal{G}$ contains two disjoint sets or all the sets in $\mathcal{G}$ have a common element. A graph is *clique-Helly* if its family of maximal cliques has the Helly property.

For any graph G, let G' denote the graph obtained from G by adding, for each vertex v of G, a new vertex v' and a new edge joining v to v'.

Here is the characterization of dually DV graphs:

Theorem 7 *A graph G is a dually DV graph if and only if G is clique-Helly and $K(G')$ is a DV graph.*

The authors also derive an algorithm of complexity $O(|E(G)|^4)$ to determine whether a given graph G is dually DV.

Let us now describe their characterization of clique graphs of rooted path graphs. A *rooted tree* is a directed tree having precisely one vertex with in-degree zero (a rooted tree is sometimes called a *branching*). A *rooted path graph* (or *RDV graph*) is the intersection graph of the family of directed paths of a rooted tree. A *dually rooted path graph* (or *dually RDV graph*) is a graph G that admits a spanning rooted tree T such that, for each edge (v, w) of G, T contains a directed $v - w$ path or a directed $w - v$ path whose vertices form a clique in G.

A *chordal graph* is a graph that contains no induced cycle of length four or greater. A *strong chord* of a cycle of a graph is a chord that joins two vertices of the cycle with an odd distance in the cycle. A *strongly chordal* graph is a graph in which every cycle on six or more vertices contains a strong chord.

Here is the characterization for dually RDV graphs:

Theorem 8 *The following statements are equivalent for a graph G:*

(i) *G is a dually RDV graph.*

(ii) *G is clique-Helly and $K(G')$ is an RDV graph.*

(iii) *G is strongly chordal and $K(G')$ is an RDV graph.*

The authors also derive an algorithm $O(|V(G)^{2.38}|)$ to determine whether or not a given graph G is dually RDV.

Many other results

We now examine very briefly other results obtained by Jayme.

Comparability graphs

Let D be an acyclic orientation of a graph G. Then, D is *transitive* if, for each pair (u, v) and (v, w) of edges of D, edge (u, w) also lies in D. A graph is a *comparability graph* if it admits a transitive orientation.

For any two vertices v and w of D, let $\langle v, w\rangle$ denote the set consisting of those vertices that are simultaneously descendants of v and ancestors of w. Orientation D is *locally transitive* if $G[\langle v, w\rangle]$ is transitive, for each edge (v, w) of D. Graph G is *local comparability* if it admits a locally transitive orientation.

A graph G is *P_4-comparability* if it admits an orientation D such that the restriction of D to the subgraph of G spanned by the set of vertices of each path of length three is transitive.

A *circle graph* is the intersection graph of chords of a circle, in which no two chords have a common point in the circle.

A pair $\{v, w\}$ of vertices of G is *even* if every induced path from v to w has even length. A pair $\{v, w\}$ is *odd* if v and w are nonadjacent and each induced path from v to w has odd length.

There are four papers on this subject that should be mentioned. In the first one [48], Jayme introduces the concept of local comparability graphs, as a generalization of comparability graphs. The class of local comparability graphs includes the comparability graphs and the circle graphs.

The first main result in that paper is that every local comparability graph is a difference of two comparability graphs. The second main result is that the class of local comparability graphs of dimension 1 is precisely the class of connected interval graphs that correspond to a set of totally noncomparable intervals of the real line. Circle graphs are similarly but less concisely characterized.

The next three papers show a beautiful evolution of thought, culminating with a nice characterization

of source and sink sets and also a characterization of even and odd pairs in a comparability graph.

The first of these three papers considers the problem of determining whether a comparability graph has a transitive orientation with specified sources and sinks. It is a joint work with Célia Mello and Celina Figueiredo [56]. They consider clique partitions of a comparability graph and determine some necessary conditions for the existence of a solution to the problem. This condition turns out to be sufficient for graphs with at most three maximal cliques. In particular, if only sources are specified, then the set is a source set if and only if each pair of vertices in S is an even pair and each vertex of S is a source of some transitive orientation.

In the second paper of the series, a new author joins the team: John Gimbel [19]. The authors find a condition that is necessary and sufficient for the problem to have a solution. For a specified set S of sources and a specified set T of sinks, the authors construct a graph $G(S,T)$ that is trivially obtained from G, S and T and has size linear on the size of G. Then, they show that the problem has a solution if and only if $G(S,T)$ is a comparability graph. So, not only they solve the problem from a mathematical point of view, but they also give a polynomial algorithm for deciding whether the problem has a solution.

Finally, in the third paper of the series [18], they characterize even and odd pairs in comparability and in P_4 -comparability graphs. The characterizations lead to simple algorithms for deciding whether a given pair of vertices forms an even or odd pair in these classes of graphs. The complexities of the proposed algorithms are $O(n+m)$ for comparability graphs and $O(n^2m)$ for P_4 -comparability graphs. The former represents an improvement over a recent algorithm of complexity $O(nm)$.

Cliques

There is an enormous number of significant results involving cliques. Some of these have already been described. Here are some more.

Clique graphs free of K_3 and K_4

This is joint work with Fábio Protti [38]. The authors characterize the graphs whose clique graphs are free of triangles in terms of forbidden induced subgraphs: $K_{1,3}$, the 4-fan and K_4. The *4-fan* is the graph obtained from the 4-wheel by deleting an edge from the rim. They give a similar characterization for graphs whose clique graphs are free of K_4.

Clique-inverse graphs of bipartite graphs

This is also joint work with Fábio Protti [40]. The authors characterize the families of graphs whose clique graphs are bipartite, in terms of forbidden configurations: the clique graph of a graph G is bipartite if and only if G is free of induced subgraphs in the following list: $K_{1,3}$, the 4-fan, the 4-wheel, C_{2n+5} $(n \geq 0)$. They also characterize two more classes: (i) those graphs whose clique graphs are chordal bipartite graphs and (ii) those graphs whose clique graphs are a tree.

Clique graphs with linear size

Another joint work with Fábio Protti [39]. Let G be a graph. By examining $K(G)$, the authors describe some sufficient conditions for the number of maximal cliques of G to be bounded by $O(|V(G)|)$. These conditions are then applied to analyze the complexity of recognizing clique-inverse graphs of various classes of graphs. In some cases, polynomial time algorithms are obtained, such as in the case of $K^{-1}(K_r$-free). In other cases, the bound is used to show that certificates may be verified in polynomial time, within a proof of NP-completeness.

Clique-Helly graphs

In this paper [49], Jayme describes a characterization of clique-Helly graphs, leading to a polynomial time algorithm for recognizing them.

Clique-complete graphs

This is a joint work with Cláudio Lucchesi and Célia Mello [32]. At the time, Célia had just completed her doctoral thesis, under the supervision of Jayme. Some years prior to that, Célia had written her Master's dissertation under my supervision. So, it was a very pleasant opportunity to be a coauthor with Jayme and a former student of both of us.

For a natural number n, a graph G is *n-convergent* if $K^n(G)$ is isomorphic to K_1, the one-vertex graph. A graph G is *convergent* if it is n-convergent for some natural number n. A 2-convergent graph is called *clique-complete*. A *universal vertex* is a vertex adjacent to every vertex of the graph.

The authors describe the family of minimal graphs which are clique-complete but have no universal vertices. The minimality used there refers to induced subgraphs. In addition, they show that recognizing clique-complete graphs is Co-NP complete.

Clique convergent graphs

This is a joint work with Claudson Bornstein [11]. The *index* of a convergent graph G is the smallest n such that G is n-convergent, while its *Helly defect* is the smallest n such that $K^n(G)$ is clique-Helly. H. Bandelt and E. Prisner [3] proved that the Helly defect of a chordal graph is at most one and asked whether there is a graph whose Helly defect exceeds the difference of its index and diameter by more than one. In this paper an affirmative constructive answer to the above question is given: for any arbitrary finite integer $n \geq 0$ a graph is exhibited in which the Helly defect exceeds by n the difference of its index and diameter.

Clique graphs of chordal graphs and of path graphs

Another joint work with Claudson Bornstein [52], where the authors characterize the clique graphs of chordal graphs and the clique graphs of path graphs.

Computing all maximal cliques distributedly

This is joint work with Fábio Protti and Felipe França [37]. The authors present a parallel algorithm for generating all maximal cliques of a graph. The time complexity of the algorithm is restricted to the induced neighborhood of a vertex and the communication complexity is $O(M\Delta)$, where M is the number of connections and Δ the maximum degree in the graph.

Enumeration of maximal cliques of a circle graph

This is joint work with Magali Barroso [51]. The authors apply the notion of locally edge transitive orientations of an undirected graph and obtain an algorithm for generating all maximal cliques of a circle graph G in time $O(n(m+\alpha))$, where n, m and α are the number of vertices, edges and maximal cliques of G. In addition, they show that the actual number of such cliques can be computed in $O(nm)$ time.

Maximal cliques in circle graphs

This is joint work with Edson Cáceres and Siang Song [14]. A *Coarse Grained Multicomputer* (CGM) consists of a set of p processors with $O(N/p)$ local memory per processor and an arbitrary communication network (or a shared memory). A CGM algorithm consists of alternating local computation and global communication rounds. At each communication round, each processor sends and receives $O(N/p)$ data.

In this paper, the authors present a parallel algorithm for finding the maximal cliques of a circle graph using the CGM model. The proposed algorithm requires $O(\log p)$ communication rounds. In a regular, sequential depth search, normally each edge is visited a constant number of times. The authors devised a new technique, called the *unrestricted depth search*, in which each edge may be visited an unbounded (but finite) number of times. The authors regard this technique as the main contribution of the paper. The three authors also have another paper on unrestricted depth search in parallel [15].

Edge clique graphs

The *edge clique graph* $K_e(G)$ of a graph G is the graph whose set of vertices is the set of edges of G, two vertices of $K_e(G)$ are adjacent if and only if the corresponding edges lie in a (common) clique of G.

An *edge component* of a graph G is a component of its edge clique graph.

Characterization of edge clique graphs

This is joint work with Márcia Cerioli [17]. A *k-labeling* of a graph G with n vertices is an assignment of a set $l(v) \subset \{1, 2, \cdots, n\}$ to each vertex v of G, such that $|l(v)| = k$ and all label sets are distinct. A set S of vertices is *triangular* if $|S| = \binom{r}{2}$ for some integer r. Set S of vertices is *strongly triangular*, with respect to a 2-labeling l, if $|S| = \binom{|l(S)|}{2}$. The authors show that a graph G is an edge clique graph if and only if it has a 2-labeling that satisfies the following two properties: (i) every maximal clique is strongly triangular and (ii) every strongly triangular set is a clique.

Starlike graphs

Denote by $N(v)$ the set of vertices that are adjacent to v in a graph G and by $N[v]$ the set $\{v\} \cup N(v)$. A graph G is *starlike* if there exists a partition $C, D_1, \cdots, D_s$ ($s \geq 0$) of the set of veretices of G such that C is a maximal clique and, for $u \in D_i$, $v \in D_j$, $i \neq j$ implies that $\{u, v\} \notin E(G)$, whereas $i = j$ implies that $N[u] = N[v]$. It follows that each D_i is included by precisely one maximal clique C_i, and $D_i = C_i - C$. If, in addition, $C \cap C_i \subset C \cap C_{i+1}$ for $1 \leq i < s$, then G is a *starlike-threshold graph*.

A *generalized starlike graph* is a graph G such that precisely one of its edge components is a starlike graph,

the others complete graphs.

A *generalized starlike-threshold graph* is a graph G such that precisely one of its edge components is a starlike-threshold graph, the others complete graphs.

A *split graph* is a graph that admits a partition C, I of its set of vertices such that C is a clique and I an independent set of vertices. Thus, a split graph is a particular case of a starlike graph, in which each D_i is a singleton, for $1 \leq i \leq s$.

This is also joint work with Márcia Cerioli [16]. In this paper, the authors show that the class of starlike (starlike-threshold) graphs contains the class of edge clique graphs of generalized starlike (starlike-threshold) graphs. In addition, every starlike (starlike-threshold) graph which is an edge clique graph is an edge clique graph of a generalized starlike (starlike-threshold) graph. They also prove that a starlike-threshold graph is an edge clique graph if and only if its maximal cliques and intersections of maximal cliques are triangular sets.

Directed graphs

Jayme published several papers related to efficient algorithms for directed graphs. Let us take a brief look at each one of them.

Enumeration of directed circuits

This is a joint work with P. E. Lauer [54]. The authors give an $O(n + mc)$ algorithm for enumerating all the directed circuits of a directed graph on m edges, n vertices and c directed circuits.

Enumeration of Kernels

A *kernel* N of a directed graph D is an independent set of vertices of D such that for every $w \in V(D) - N$ there is an edge from w to N. The existence of a kernel in an directed graph with no odd directed cycles was proved by M. Richardson [41].

This is a joint work with G. Chaty [53]. The authors give an algorithm for generating all distinct kernels in a directed graph D with no odd directed circuits. The complexity of the algorithm is $O(nm(k+1))$, where n, m and k are the number of vertices, edges and kernels of D. Also, they show that the problem of determining the number of kernels in a directed graph D is $\#P$-complete, even if the longest directed circuit of D has length two.

A minimax equality

The problem of finding the minimum set of vertices that intersects all circuits in a directed graph is NP-complete [21]. Jayme published a paper [47] in which he introduces the class of *connectively reducible digraphs* and shows that it contains two classes known to admit polynomial solutions: the class of *fully reducible subgraphs* and the class of *cyclically reducible digraphs*. He also describes an algorithm $O(n^2(n+m))$ that recognizes connectively reducible directed graphs and determines a (minimum) set T of vertices that intersects all directed circuits for those graphs and a (maximum) vertex-disjoint set of directed circuits having cardinality equal to that of T.

Orientations with single source and sink

This is joint work with Ronaldo Persiano and Antonio Oliveira [58]. Given an undirected graph G, possibly with multiple edges, and distinct vertices s and t of G, the authors consider several orientations D of G. One of these orientations is acyclic and has s and t as the only source and sink of D, respectively. They show that this is possible if and only if graph $G + st$ is 2-connected. For each of the problems considered, they use depth-first search to give linear time algorithms for finding the orientations or determine that they do not exist.

Generation of acyclic orientations

This is joint work with Valmir C. Barbosa [4]. The authors describe an algorithm for finding all the acyclic orientations of a graph G in overall time $O((n+m)\alpha)$ and delay complexity $O(n(n+m))$, where G has n vertices, m edges and α acyclic orientations. The space required is $O(n + m)$.

Rooted tree structure

A directed graph $D = D(V, E)$ with a given root vertex s is *reducible* if every depth-first search tree with root s has the same set B of back edges. Thus, for a reducible directed graph D, the associated dag (the subgraph with vertex set V and edge set $E - B$) is uniquely defined. A *tree reducible graph* is a reducible subgraph for which the transitive reduction (a smallest directed graph with the same reachability) of the associated dag is an arborescence (outdirected tree) with root s.

In this paper [44], Jayme gives polynomial algorithm for (1) recognizing, (2) finding isomorphisms

between, and (3) finding minimum equivalent directed graphs for, tree reducible graphs.

Split-indifference graphs

This is a joint work with Carmen Ortiz and Nelson Maculan [33]. An *indifference graph* is an intersection graph on a set of unit intervals on the real line. A *split-indifference graph* is a split graph that is also an indifference graph. The authors give the following characterization of split-indifference graphs:

Theorem 9 *A connected graph G is split-indifference if and only if*

(i) *G is complete, or*

(ii) *G is the union of two cliques G_1 and G_2 such that $G_1 - G_2 = K_1$, or*

(iii) *G is the union of three cliques G_1, G_2, G_3 such that $G_1 - G_2 = K_1$, $G_2 - G_3 = K_1$ and*

$$V(G_1) \cap V(G_3) = \emptyset \quad or \quad V(G_1) \cup V(G_3) = V(G).$$

Using that characterization, they determine the chromatic index $\chi'(G)$ of split-indifference graphs. In order to do that, they construct an edge coloring of K_{2n}, $n \geq 3$, using $2n-1$ colors such that K_{2n} has a perfect matching without color repetitions.

The resulting algorithm is very simple. It determines in linear time an optimum edge coloring of a split indifference graph

Other results

There are many other results that I should describe, but length restrictions force me to be very concise.

Task scheduling

Jayme has four papers in this area, three of them with J. Błażewicz and W. Kubiak [6, 7, 8, 45].

Euler tours

A joint work with Edson Cáceres, Narsingh Deo and Shivakumar Sastry [13] describes an alternative implementation of Atallah and Vishkin's parallel algorithm for finding an Euler tour of a graph [1].

Search

I should mention here three papers. The first paper is a joint work with L. B. Wilson, on ternary trees [59]. The second paper is joint work with G. Navarro et al., on optimal binary search trees with costs depending on the access paths [57].

The third paper is a joint work with Marina Moscarini and Rossella Petreschi, *Node Searching and Starlike Graphs*. It is a very interesting paper. Let G be a graph whose vertices are *contaminated*. Assigning a *searcher* to a contaminated vertex makes it become *guarded*. Removing the searcher of a guarded vertex turns it *clear*. However, a clear vertex becomes contaminated again if it has a contaminated neighbor. The *node-search number* of G is the least number of searchers needed to clear all its vertices. J. Gustedt [23] has shown that the problem of determining the node search number of G is NP-hard for uniform k-starlike graphs. These graphs are generalizations of split graphs, obtained when each vertex of the independent set of the bipartition of the split graph is replaced by a k-vertex clique. The authors describe necessary and sufficient conditions for finding the node-search number of a uniform k-starlike graph. The characterization described extends a corresponding result for split graphs by T. Kloks [27]. In addition, it leads to a new algorithm for finding the node-search number for graphs of this class.

Acknowledgements I would like to thank Celina M. H. de Figueiredo and Valmir C. Barbosa for having invited me to write this article. Their kind invitation gave me the opportunity to really grasp the extent of Jayme's work over the years. I hope that this article will help others to appreciate the breadth of Jayme's work.

I would also like to thank Valmir C. Barbosa, Márcia Cerioli, Celina M. H. de Figueiredo, Sulamita Klein and U. S. R. Murty for reading an earlier draft and giving me many helpful suggestions.

References

[1] M. Atallah and U. Vishkin. Finding Euler tours in parallel. *J. Comput. System Sci.*, 29(3):330–337, 1984.

[2] R. Balakrishnan and P. Paulraja. Self-clique graphs and diameters of iterated clique graphs. *Utilitas Math.*, 29:263–268, 1986.

[3] H.-J. Bandelt and E. Prisner. Clique graphs and Helly graphs. *J. Combin. Theory Ser. B*, 51(1):34–45, 1991.

[4] V. C. Barbosa and J. L. Szwarcfiter. Generating all the acyclic orientations of an undirected graph. *Inform. Process. Lett.*, 72(1-2):71–74, 1999.

[5] R. Bayer and E. McCreight. Organization and maintenance of logic ordered indexes. *Acta Informatica*, 1:173–189, 1971.

[6] J. Błażewicz, W. Kubiak, H. Röck, and J. Szwarcfiter. Minimizing mean flow-time with parallel processors and resource constraints. *Acta Inform.*, 24(5):513–524, 1987.

[7] J. Błażewicz, W. Kubiak, and J. Szwarcfiter. Scheduling unit-time tasks on flow-shops under resource constraints. *Ann. Oper. Res.*, 16(1-4):255–266, 1988. Multi-attribute decision making via O.R.-based expert systems (Passau, 1986).

[8] J. Błażewicz, W. Kubiak, and J. Szwarcfiter. Scheduling independent fixed-type tasks. In *Advances in project scheduling*, pages 225–236. Elsevier, Amsterdam, 1989.

[9] J. A. Bondy and V. Chvátal. A method in graph theory. *Discrete Math.*, 15(2):111–135, 1976.

[10] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. MacMillan, 1976.

[11] C. F. Bornstein and J. L. Szwarcfiter. On clique convergent graphs. *Graphs Combin.*, 11(3):213–220, 1995.

[12] C. F. Bornstein and J. L. Szwarcfiter. Iterated clique graphs with increasing diameters. *J. Graph Theory*, 28(3):147–154, 1998.

[13] E. N. Cáceres, N. Deo, S. Sastry, and J. L. Szwarcfiter. On finding Euler tours in parallel. *Parallel Process. Lett.*, 3(3):223–231, 1993.

[14] E. N. Cáceres, S. W. Song, and J. L. Szwarcfiter. A coarse-grained parallel algorithm for maximal cliques in circle graphs. In V. Alexandrov, J. Dongarra, B. Juliano, R. Renner, and C. Tan, editors, *Proceedings of Computational Science - ICCS 2001 - Part II*, volume 2074 of *Lecture Notes in Computer Science*, pages 638–647. Springer, 2001.

[15] E. N. Cáceres, S. W. Song, and J. L. Szwarcfiter. A parallel unrestricted depth search algorithm. In H. R. Arabnia, editor, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Appl*

[16] M. R. Cerioli and J. L. Swzarcfiter. Edge clique graphs and some classes of chordal graphs. *Discrete Mathematics*, 242:31–39, 2002.

[17] M. R. Cerioli and J. L. Szwarcfiter. A characterization of edge clique graphs. *Ars Combin.*, 60:287–292, 2001.

[18] C. M. H. de Figueiredo, J. Gimbel, C. P. Mello, and J. L. Szwarcfiter. Even and odd pairs in comparability and in P_4-comparability graphs. *Discrete Appl. Math.*, 91(1-3):293–297, 1999.

[19] C. M. H. de Figueiredo, J. Gimbel, C. P. Mello, and J. L. Szwarcfiter. Sources and sinks in comparability graphs. *Order*, 14(1):75–83, 1997.

[20] C. M. H. de Figueiredo and J. L. Szwarcfiter. Emparelhamentos em grafos. In *Anais do XIX Congresso nacional da Sociedade Brasileira de Computação*, Jornada de Atualização em Informática, pages 127–161, 1999. In Portuguese.

[21] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[22] E. N. Gilbert and E. F. Moore. Variable-length binary encodings. *Bell System Tech. J.*, 38:933–967, 1959.

[23] J. Gustedt. On the pathwidth of chordal graphs. *Discrete Appl. Math.*, 45(3):233–248, 1993.

[24] B. Hedman. Clique graphs of time graphs. *J. Combin. Theory Ser. B*, 37(3):270–278, 1984.

[25] A. Itai. Optimal alphabetic trees. *SIAM J. Comput.*, 5(1):9–18, 1976.

[26] A. Itai, C. H. Papadimitriou, and J. L. Szwarcfiter. Hamilton paths in grid graphs. *SIAM J. Comput.*, 11(4):676–686, 1982.

[27] T. Kloks. *Treewidth*. Springer-Verlag, Berlin, 1994.

[28] M. Knor, L. Niepel, and L. Šoltés. Centers in line graphs. *Math. Slovaca*, 43(1):11–20, 1993.

[29] D. E. Knuth. *Literate Programming*, volume 27 of *CSLI Lecture Notes*. Center for the Study of Language and Information, Stanf