NAIST-IS-DT0361015

Doctoral Dissertation

ALGORITHMS FOR INDEPENDENT SET IN SOME KINDS OF GRAPHS AND ITS APPLICATIONS

Masakuni TAKI

February 6, 2004

Department of Information Systems
Graduate School of Information Science
Nara Institute of Science and Technology

Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
DOCTOR of ENGINEERING

Masakuni TAKI

Thesis committee: Dr. Kunihiro CHIHARA, Professor
Dr. Toshinobu KASHIWAHARA, Professor(Osaka Univ.)
Dr. Minoru ITO, Professor
Dr. Yoshitsugu MANABE, Associate Professor

Abstract

In this dissertation, we discuss algorithms for independent set in some kinds of graphs and its applications.

This dissertation consists of 5 Chapters, and a representation diagram for maximal independent sets of a co-comparability graph is taken up in Chapter 1, and algorithms for generating maximum weight independent sets in intersection graphs are described in Chapter 2, and Chapter 3 and Chaper 4 describe online edge-coloring algorithms for degree bounded bipartite graphs, and application of transitive graph for education support system in Chapter 5.

We are discussing the relation of each chapter from a common problem of independent set respectively. The outline of each chapter is described here.
Chapter 1 : Let $H = (V(H), E(H))$ be a directed graph with distinguished vertices s and t. An st-path in H is a simple directed path starting from s and ending at t. Let $\mathcal{P}(H)$ be defined as $\{S \mid S$ is the set of vertices on an st-path in H (s and t are excluded)$\}$. For an undirected graph $G = (V(G), E(G))$ with $V(G) \subseteq V(H) - \{s, t\}$, if the family of maximal independent sets of G coincides with $\mathcal{P}(H)$, we call H an MIS-diagram for G. In this chapter, we provide a necessary and sufficient condition for a directed graph to be an MIS-diagram for an undirected graph. we also show that an undirected graph G has an MIS-diagram iff G is a cocomparability graph. Based on the proof of the latter result, we can construct an efficient algorithm for generating all maximal independent sets of a cocomparability graph.
Chapter 2 : In this chapter we propose an algorithm for generating maximum weight independent sets in a circle graph, that is, for putting out all maximum weight independent sets one by one without duplication. The time complexity is $O(n^3 + \beta)$, where n is the number of vertices, β output size, i.e., the sum of the cardinalities of the output sets. It is shown that the same approach can be applied for spider graphs and for circular-arc overlap graphs.
Chapter 3 : we consider an edge coloring game of a graph by two persons who are an adversary and an edge-painter. The former strategy is to add or to delete edges in the graph successively. The strategy is called input. The latter colors the edge as soon as an edge is added in the graph. Then it is never changed the color of edges colored. The latter is not given any further information of the adversary's. we call the coloring strategies of the painter online edge-coloring algorithm. In this chapter the painter is allowed arbitrary amount of colors and colors all of edges added in the graph. A

maximum ratio which is a ratio between the number of colors used of an online edge-coloring algorithm A for adversary's family of input and that of offline edge-coloring algorithm is called a competitiveness coefficient of an online edge-coloring algorithm A. we have proved that a competitiveness coefficient of arbitrary randomized online edge-coloring algorithm is greater than $\frac{2k-1}{k}$ where k is maximum degree.

Chapter 4 : A kind of online edge-coloring problems on bipartite graphs is considered. The input is a graph (typically with no edges) and a sequence of operations (edge addition and edge deletion) under the restriction that at any time the graph is bipartite and degree-bounded by k, where k is a prescribed integer. At the time of edge addition, the added edge can be irrevocably assigned a color or be left uncolored. No other coloring or color alteration is allowed. Coloring can be done only at the time of addition of the edge; no color can be assigned afterward (unless the edge is once deleted). The problem is to assign colors as many times as possible using k colors. Two algorithms are presented: one with competitiveness coefficient $\frac{1}{4}$, and one with competitiveness coefficient between $\frac{1}{4}$ and $\frac{1}{2}$ with the cost of requiring more random bits than the former algorithm, also against oblivious adversaries.

Chapter 5 : It is reported an education support system to take required credits from entrance to graduation in a college. This system is applied by a directed graph that subjects are vertices and relations among subjects are edges. we present an st-digraph that entrance and graduation are vertex s and vertex t respectively and subjects from the entrance to the graduation are vertices. These relations among subjects are transitive by adding some edges. Then, if there are minimal st-separators of the st-digraph, those are the required credits from entrance to graduation. we propose a method to decide the required credits for graduation.

keywords:
independent set, comparability graph, intersection graph, competitiveness coefficient, education support system

Acknowledgement

I am deeply indebted to many people for the advice and supports on this work. I would especially like to thank my supervisor Professor Dr. Kunihiro Chihara, Professor Dr. Toshinobu Kashiwahara and Professor Dr. Sumio Masuda for their invaluable suggestions, advice, and discussion throughout the work.

I am obliged to Professor Dr. Minoru Ito and Professor Dr. Yoshitsugu Manabe for their helpful comments and suggestions.

I would also like to express hearty thanks to all the member of Chihara Laboratory and Kashiwahara Laboratory for their kind-supports.

Finally, I would like to thank my parents(Kunio and Yayoi), my wife(Hiroko), my sons(Yoshikuni and Hirokuni), and my daughter(Hatsuka) for their patiently kind-supports.

Contents

0 Introduction

An independent set of a graph is a set of vertices no two elements of which are adjacent. An independent set is called maximal if it is contained in no other independent set (abbreviated as MIS). An independent set with maximum cardinality is called a maximum independent set . For a vertex-weighted graph, a maximum weight independent set (abbr. MWIS) is an independent set which has the maximum weight among all independent sets. A generation problem for a graph is a problem of finding all the subgraphs of a given graph each of which has some prespecified property. Several kinds of generation problems have been considered for various classes of graphs [1]~[7].

Let $H = (V(H), E(H))$ be a directed graph with distinguished vertices s and t. An st-path in H is a simple directed path starting from s and ending at t. Let $\mathcal{P}(H)$ be defined as $\{S \mid S$ is the set of vertices on an st-path in H (s and t are excluded)$\}$. For an undirected graph $G = (V(G), E(G))$ with $V(G) \subseteq V(H) - \{s, t\}$, if the family of maximal independent sets of G coincides with $\mathcal{P}(H)$, we call H an MIS-diagram for G.

In Ref. [4], it is shown that any interval graph has an MIS-diagram that is an acyclic digraph (abbr. a dag). The result is extended to the vertex-weighted case in Ref. [6].

For an undirected graph G, if we can construct an acyclic MIS-diagram for G, the problem of generating all MIS's of G becomes that of generating all the st-paths in a dag, which is a quite easy task. This fact motivates us to study MIS-diagrams for undirected graphs. In chapter 1, we consider the following two problems:
Problem 1: What kind of digraphs can be MIS-diagrams?
Problem 2: What kind of graphs have MIS-diagrams?

In chapter 1, we provide a necessary and sufficient condition for a directed graph to be an MIS-diagram for an undirected graph. we also show that an undirected graph G has an MIS-diagram iff G is a cocomparability graph. Based on the proof of the latter result, we can construct an efficient algorithm for generating all maximal independent sets of a cocomparability graph.

A circle graph is an intersection graph defined on a set of chords in a circle: there exists an edge between two vertices iff their corresponding chords intersect. The circle graphs were introduced by Even and Itai in [13], and since then, many problems have been studied on them. Problems of maximum independent set, maximum clique, Hamiltonian circuit, and recognition can be solved in polynomial time for circle graphs [12, 14, 9, 10]. On the other hand, it is known that coloring and domination problems on circle graphs are NP-complete [16, 18].

Generation algorithms have been investigated for many years; cycle generation, cutset generation, maximal independent set generation, clique generation, etc. For maximal-independent-set generation of arbitrary graphs, the fastest algorithm to date takes $O(nm\alpha)$ time, where n is the cardinality of the vertex set, m the cardinality of the edge set, and α the number of maximal independent sets generated [1]. For particular graphs, much more efficient algorithms are known. In chapter 2, we propose an algorithm for generating maximum weight independent sets in a circle graph, that is, for putting out all maximum weight independent sets one by one without duplication. The time complexity is $O(n^3+\beta)$, where n is the number of vertices, β output size, i.e., the sum of the cardinalities of the output sets. It is shown that the same approach can be applied for spider graphs and for circular-arc overlap graphs.

Vertex- or edge- coloring of a graph is to assign colors to vertices or edges in such a way that no two vertices or edges with the same color are adjacent. There seem to exist two types of coloring problems: "assignment maximization type" in which the number of available colors is prespecified and the objective is to maximize the number of vertices or edges that are assigned colors, and "color minimization type" in which the objective is to minimize the number of colors used to color all vertices or edges.

In resent years, online versions of existing problems have been considered, among them are the online versions of these coloring problems.

Competitive analysis of vertex coloring problems of both color minimization type and of assignment maximization type are reported [20], [21]. But in [20], for randomized online edge-coloring algorithm, the outcome was only shown briefly, however the proof was imperfect. In chapter 3, we have proved that the competitiveness coefficient of arbitrary randomized online edge-coloring algorithm is greater than $\frac{2k-1}{k}$ where k is maximum degree. Competitive analysis on edge coloring problems of assignment maximization type seems not to have been reported. In chapter 4, we concentrate on online edge coloring problems of this type in which the graph is subjected to operations of both edge addition and edge deletion such that at any time the graph has some special properties: 1) the graph is bipartite with prespecified bipartition, and 2) maximum degree of vertices is not more than a prespecified integer k. Such a property is related to assignment problems on some switching network. We consider only the case in which the number of available colors is equal to k. The problem is to assign colors as many times as possible using k colors. Two algorithms are presented: one with competitiveness coefficient $\frac{1}{4}$, and one with large competitiveness coefficient (which is between $\frac{1}{4}$ and $\frac{1}{2}$) but needs more random bits than the former one random bit per one edge addition on the fly).

The research of the structure of the graph concerning the independent set is described up to the above-mentioned Chapter 4. In Chapter 5, the applied problem with the research of the above-mentioned is taken up. Here, the above-mentioned graph structure is used for the education support system for students to take required credits from entrance to graduation in the academic education etc. This system is applied by a directed graph that subjects are vertices and relations among subjects are edges. There is an st-digraph that entrance and graduation are vertex s and vertex t respectively and subjects from the entrance to the graduation are vertices. These relations among subjects are transitive by adding some edges. Then, if there are minimal st-separators of the st-digraph, those are the required credits from entrance to graduation. It is proposed the algorithm to which such minimal st-separators were obtained the required credits for graduation.

1 A Representation Diagram for Maximal Independent Sets of a Graph

1.1 Introduction

A generation problem for a graph is a problem of finding all the subgraphs of a given graph each of which has some prespecified property. Several kinds of generation problems have been considered for various classes of graphs [1]~[7].

Let H be a directed graph (a digraph, for short) with distinguished vertices s and t. An st-path in H is a simple directed path starting from s and ending at t. Let $\mathcal{P}(H)$ be defined as $\{S \mid S$ is the set of vertices on an st-path in H (s and t are excluded)$\}$. For an undirected graph $G = (V(G), E(G))$ with $V(G) \subseteq V(H) - \{s, t\}$, if the family of maximal independent sets (abbreviated as MIS's) of G coincides with

$\mathcal{P}(H)$, we call H a representation diagram for MIS's of G, or an MIS-diagram for G.

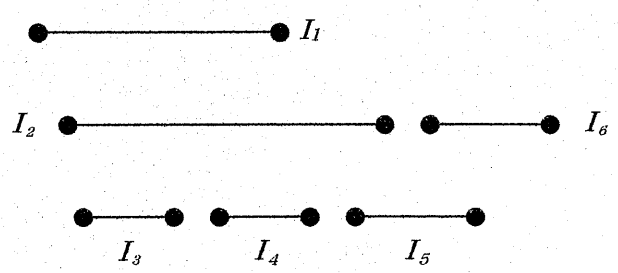
In Ref. [4], it is shown that any interval graph has an MIS-diagram that is an acyclic digraph (abbr. a dag). An example is given in Figure 1. The result is extended to the vertex-weighted case in Ref. [6].

For an undirected graph G, if we can construct an acyclic MIS-diagram for G, the problem of generating all MIS's of G becomes that of generating all the st-paths in a dag, which is a quite easy task. This fact motivates us to study MIS-diagrams for undirected graphs. In this section, we consider the following two problems:
Problem 1: What kind of digraphs can be MIS-diagrams?
Problem 2: What kind of graphs have MIS-diagrams?

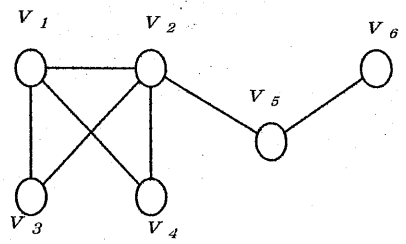
This section is organized as follows: Sect. 1.2 defines several terms and notations. Sect. 1.3 provides necessary and sufficient conditions for both of the above problems. Sect. 1.4 introduces a diagram to represent all the maximum weight independent sets of a vertex-weighted graph. Finally, Sect. 1.5 concludes this section.

1.2 Preliminaries

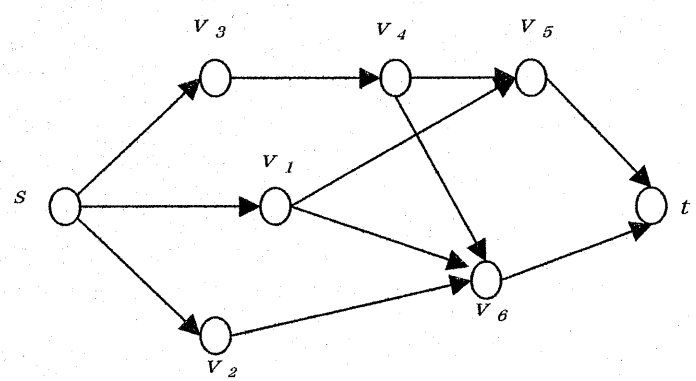
In this section, a directed graph is called a digraph, whereas an undirected graph is simply called a graph. An undirected edge between vertices u and v is denoted by (u, v). A directed edge from u to v is denoted by $(u \rightarrow v)$.



(a)



(b)



(c)

Figure 1: (a)A familyn of intervals, (b)the coresponding interval graph G (vertex v_i corresponds to interval I_i for i =1,2,...,6), and (c)an MIS-diagram for G.

For a graph or digraph G, $V(G)$ and $E(G)$ denote the set of vertices and that of edges, respectively, of G. For graphs or digraphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$, $G \cap H$ denotes the intersection of G and H, i.e., the graph or digraph $(V(G) \cap V(H), E(G) \cap E(H))$. Similarly, $G \cup H$ is the union of G and H.

For a digraph G, the underlying graph of G is the graph obtained from G by replacing each directed edge $(u \rightarrow v)$ with an undirected edge (u, v). Note that parallel edges may result. A digraph is said to be connected if its underlying graph is connected. A connected component of a digraph is similarly defined.

In a digraph G, if the number of edges coming into vertex v is 0, then v is called a source of G. On the other hand, the number of edges going out of v is 0, v is called a sink of G.

A digraph is said to be transitive if the existence of two edges $(u \rightarrow v)$ and $(v \rightarrow w)$ implies the existence of edge $(u \rightarrow w)$. A graph is a comparability graph if the edges can be oriented so that the resultant digraph is transitive. Furthermore, such an orientation is called a transitive orientation. A comparability graph may be called a transitively orientable graph.

The complement G^C of a graph G is the graph with vertex set $V(G)$ and complementary edge set: for two distinct vertices u and v in $V(G)$, G^C has edge (u, v) iff G has not. A graph is a cocomparability graph if its complement is a comparability graph.

A set of vertices in a graph is called an independent set if no two vertices in it are mutually adjacent. An independent set which is not contained in any other independent set is called a maximal independent set, which is abbreviated as MIS in this section.

A set S of vertices is called a clique if any two vertices in it are mutually adjacent. Note that S is a clique of G iff it is an independent set of G^C.

In a digraph, a directed walk is a sequence of vertices and edges $(v_1, e_1, v_2, e_2, \cdots, e_k, v_{k+1})$ such that e_i is an edge from v_i to v_{i+1} for $i = 1, 2, \cdots, k$. A directed walk is called a directed path if no vertex appears more than once. A directed walk is called a directed cycle if no vertex appears more than once except that the first and last vertices are the same. A digraph which has no directed cycle is called an acyclic digraph (abbr. a dag).

We consider that directed paths and directed cycles are subgraphs of the original graph: specifying the set of vertices and that of edges suffices for determining a directed path or a directed cycle. Thus, for a directed path P, we denote by $V(P)$ and $E(P)$ the set of vertices and that of edges, respectively, on P. Similar notations are used for directed cycles.

For a directed path P passing through vertices u and v in this order,

$P(u \to v)$ denotes the directed path from u to v that traverses vertices and edges as in P. Similarly, for a directed cycle C, $C(u \to v)$ represents the portion of C from u to v. In particular, $C(u \to u)$ means a graph with only one vertex u.

A graph G is said to cover another graph G' if G' is a subgraph of G. Moreover, a set F of graphs is said to cover G' if every vertex and edge of G' belongs to some graph in F. In this section, we are mainly concerned with the case in which a graph is covered by a set of directed paths.

1.3 Conditions for MIS-diagrams

In this section, we answer Problems 1 and 2.

We first consider Problem 1. Let H be a digraph with distinguished vertices s and t. As defined in Sect. 1, $\mathcal{P}(H) = \{V(P) - \{s,t\} \mid P$ is an st-path in $H\}$. H is an MIS-diagram for a graph G if the family of MIS's of G coincides with $\mathcal{P}(H)$.

We say that H is st-full if each vertex is on some st-path and also each edge is on some st-path. A maximal st-full subgraph of H is the subgraph of H which is st-full and is maximal with respect to this property. Figure 2 shows an example, where the dotted lines indicate the edges not in the maximal st-full subgraph.

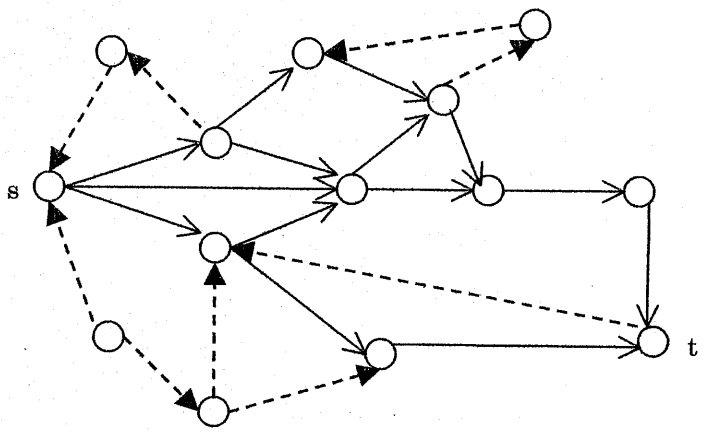


Figure 2: An example of maximal st-full subgraph

Note that the maximal st-full subgraph of H is unique.

The following lemma is trivial.

Lemma 1.1 *H is an MIS-diagram for a graph G iff the maximal st-full subgraph of H is an MIS-diagram for G.* □

Therefore, in what follows, we assume that H itself is *st*-full.

In order to show a condition for an *st*-full digraph to be an MIS-diagram, I start with the following lemma.

Lemma 1.2 *Suppose that H has a directed cycle. Then H has a directed cycle that is covered by a set of two st-paths.*

Proof *Let C be one of the directed cycles of H which is covered by the minimum number of st-paths $P_1, P_2, \cdots, P_k$ in H. we assume on the contrary that $k \geq 3$. Clearly, for $i = 1, 2, \cdots, k$, the set of directed paths $\{P_1, P_2, \cdots, P_k\} - \{P_i\}$ does not cover any cycle in H. Let i be an arbitrary integer such that $1 \leq i \leq k$. In the following, I make use of the expression "the first vertex of P_i on C" to mean the vertex on C which is met first when traversing P_i from s. "The last vertex of P_i on C" is similarly defined. Let the connected components of digraph $C \cap P_i$ be $C_i^1, C_i^2, \cdots, C_i^{m_i}$. For $j = 1, 2, \cdots, m_i$, C_i^j is a directed path and may be an isolated vertex in $C \cap P_i$. Without loss of generality, assume that C_i^1 contains the first vertex of P_i on C and that $C_i^1, C_i^2, \cdots, C_i^{m_i}$ appear on C in this order when tracing C in the direction of the edges.*
Now we have the next "fact".
Fact: *$C_i^1, C_i^2, \cdots, C_i^{m_i}$ appear in this order when traversing P_i from s to t.*
Proof of Fact: *Suppose on the contrary that $C_i^1, C_i^2, \cdots, C_i^{m_i}$ do not appear in this order on P_i. By the definition of C_i^1, C_i^1 appears first. Thus, there exist integers j and l such that $C_i^1, C_i^2, \cdots, C_i^j$ appear in this order and C_i^l ($l \geq j+2$) appears next. Let u be the last vertex of C_i^{l-1} and let v be the first vertex of C_i^l. Because P_i passes v before u and C contains a directed path from u to v, $P_i \cup C(u \rightarrow v)$ contains a directed cycle, say C'. Since any edge on $C(u \rightarrow v)$ is not on P_i, the union of $P_1, P_2, \cdots, P_{i-1}, P_{i+1}, P_{i+2}, \cdots, P_k$ contains a directed walk, say P', from s to t which covers $C(u \rightarrow v)$. P' should be an st-path because $\{P_1, P_2, \cdots, P_k\} - \{P_i\}$ does not have any directed cycle as mentioned before. Therefore, the directed cycle C' is covered by $\{P_i, P'\}$, contradicting that $k \geq 3$.* *(end of proof of Fact)*
From this "fact", by changing the portion of P_i between C_i^j and C_i^{j+1} with the portion of C between C_i^j and C_i^{j+1}, we obtain another st-path. By performing such an operation as long as possible, we can assume that $C \cap P_i$ is connected for $i = 1, 2, \cdots, k$.

As $\{P_1, P_2, \cdots, P_k\}$ covers C, we can assume that $C \cap P_1$ and $C \cap P_2$ have a common vertex (change suffix if necessary). By the definition of

$P_1, P_2, \cdots, P_k$, none of $C \cap P_1$ and $C \cap P_2$ contains the other. Thus, without loss of generality, we assume that the last vertex of P_2 on C appears after the last vertex of P_1 on C, when traversing C from the first vertex of P_1 on C.

Let v' be the first vertex of P_1 on C and let v'' be the last vertex of P_2 on C. Due to the assumption that $k \geq 3$, $P_1 \cup P_2$ does not contain directed cycles. Thus, combining $P_1(s \rightarrow v')$, $C(v' \rightarrow v'')$ and $P_2(v'' \rightarrow t)$, we have an st-path, say P'. Obviously, $\{P', P_3, P_4, \cdots, P_k\}$ covers C, which is a contradiction to the definition of k. □

Theorem 1.1 *An st-full digraph H is an MIS-diagram only if H has no directed cycle.*

Proof *Suppose that H has a directed cycle and at the same time is an MIS-diagram of a graph G. From Lemma 1.2, there exist a directed cycle C and two st-paths P_1 and P_2 such that $\{P_1, P_2\}$ covers C. By the same argument as the one in the proof of Lemma 1.2, we can assume that $C \cap P_1$ is connected. Let v_1 be the first vertex of P_1 on C and let v_2 be the last vertex of P_1 on C (the meanings of these terms "first" and "last" are the same as those in the proof of Lemma 1.2). Also let v_3 be the last vertex of P_2 on C. Since P_2 covers $C(v_2 \rightarrow v_1)$, $v_3 \in V(P_1(v_1 \rightarrow v_2)) - \{v_2\}$. Combining $P_1(s \rightarrow v_1)$, $C(v_1 \rightarrow v_3)$ and $P_2(v_3 \rightarrow t)$ results in a directed walk from s to t, each of whose vertices and edges belongs to P_1 or P_2. Obviously, this directed walk contains an st-path, say P'. As H is an MIS-diagram for G, P' must correspond to an MIS of G. In what follows, we show that this is not the case.*
Each of P_1 and P_2 corresponds to an MIS of G. Furthermore, v_2 is on both P_1 and P_2. Thus, G has no edge that connects v_2 to a vertex in $V(P_1) \cup V(P_2) - \{s, t\}$. As mentioned above, $V(P') \subseteq V(P_1) \cup V(P_2)$. Since $v_2 \notin V(P')$, P' cannot correspond to an MIS of G. □

Here we define two terms for an st-full digraph H. An edge $(u \rightarrow v)$ in H is called a redundant edge if H has a directed path from u to v with more than one edge. The transitive closure of H, which we denote by H^T, is the graph such that $V(H^T) = V(H)$ and $E(H^T) = \{(u \rightarrow v) \mid H$ has a directed path from u to $v\}$. Clearly H^T is a transitive graph.

Theorem 1.2 *Suppose that H is an st-full digraph that has no directed cycles. Then H is an MIS-diagram iff H has no redundant edges.*

Proof *(only if part)*
If H has a redundant edge, there exist two st-paths P_1 and P_2 such that $V(P_2)$ is a proper subset of $V(P_1)$. Then P_2 cannot correspond to a maximal set.

(if part)
Suppose that H has no redundant edges. Let G' be the underlying (undirected) graph of H^T and let G'' be the graph obtained from G' by deleting s and t. Moreover, let G be the complement of G''. Figure 3 shows an examlpe of these graphs or digraphs.

Note that $V(G) = V(H) - \{s, t\}$.
Let $\mathcal{F}(G)$ be the family of all MIS's of G. we show below that H is an MIS-diagram for G, that is, $\mathcal{F}(G) = \mathcal{P}(H)$.
For any two distinct st-paths P_1 and P_2 in H, $V(P_1) \neq V(P_2)$ because H is acyclic. Let P be any st-path in H. Let $V_{st}(P)$ denote $V(P) - \{s, t\}$. By the definition of G', $V_{st}(P)$ is a clique of G', and thus it is an independent set of G. Assume that $V_{st}(P)$ is not an MIS of G. Then there exists a vertex v in $V(G) - V_{st}(P)$ such that $\{v\} \cup V_{st}(P)$ is an independent set of G. This means that, in G', v is adjacent to every vertex in $V_{st}(P)$: every vertex on P (including s and t) has an edge directed to v or directed from v in H^T. Thus there exist two vertices u and u' on P such that u' is next to u on P and there are two edges $(u \rightarrow v)$ and $(v \rightarrow u')$ in H^T (u or u' may be s or t). By the definition of transitive closure, H has a directed path from u to v and a directed path from v to u'. As a whole H has a directed path from u to u' with more than one edge. Thus edge $(u \rightarrow u')$ in H is a redundant edge, a contradiction. Consequently, for any st-path P in H, $V_{st}(P)$ is an MIS of G. Therefore, $\mathcal{F}(G) \supseteq \mathcal{P}(H)$.

Let D be any MIS of G. D is a clique of G'. As is known [8], a complete directed graph has a Hamiltonian path. So H^T has a directed path passing through all vertices of D. By the definition of transitive closure, this implies that H has a directed path passing through all vertices in D. Because H is st-full and acyclic, all vertices of D is on an st-path in H, say P'. As described above, $V_{st}(P')$ is an MIS of G, and hence $D = V_{st}(P')$. Thus, $\mathcal{F}(G) \subseteq \mathcal{P}(H)$. □

We are now ready to answer Problem 1. Combining Lemma 1.1 and Theorems 1.1 and 1.2, we obtain the following Theorem.

Theorem 1.3 *A digraph H is an MIS-diagram iff the maximal st-full subgraph of H has neither directed cycles nor redundant edges.* □

As for Problem 2, we have the following Theorem.

Theorem 1.4 *A graph G has an MIS-diagram iff G is a cocomparability graph.*

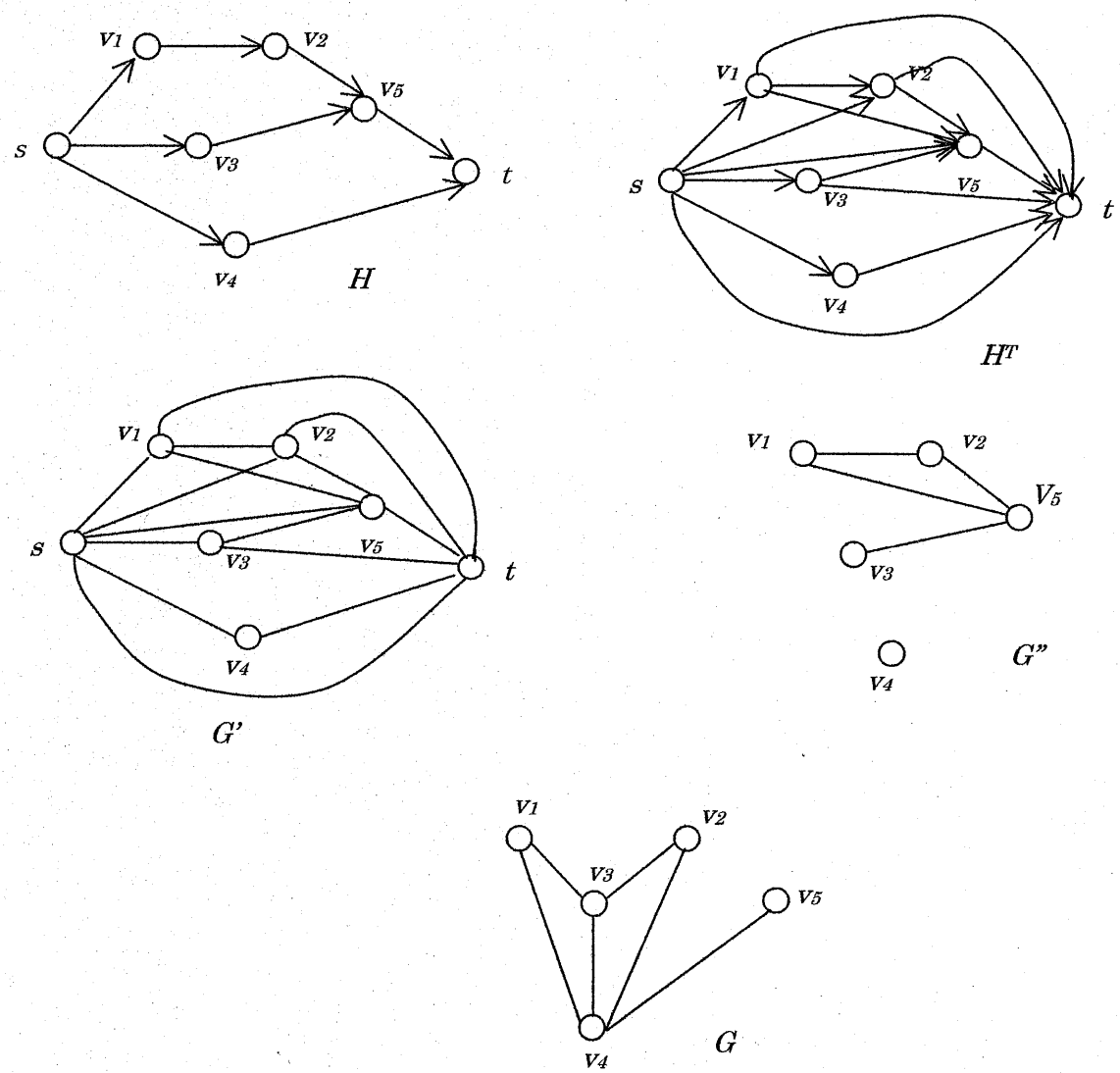


Figure 3: An example of H, H^T, G', $G"$, G.

Proof *(only if part)*
Suppose that G has an MIS-diagram H. By Lemma 1.1, we may assume that H is st-full. Let G' and G'' be defined as in the proof of Theorem 1.2. It is clear that $V(H) \subseteq V(G) \cup \{s,t\}$. Furthermore, since any vertex $v \in V(G)$ is an element of at least one MIS of G, v is contained in $V(H)$. Therefore, $V(H) = V(G) \cup \{s,t\}$, and hence $V(G) = V(G'')$.
For any two vertices $u, v \in V(G)$, if $(u,v) \notin E(G)$ then G has an MIS that contains both u and v. In this case, H^T has $(u \rightarrow v)$ or $(v \rightarrow u)$, and hence $(u,v) \in E(G'')$. On the other hand, if $(u,v) \in E(G)$, then H^T has neither $(u \rightarrow v)$ nor $(v \rightarrow u)$, and hence $(u,v) \notin E(G'')$. Thus, G'' is the complement of G. G' and G'' are comparability graphs because they are made from the transitive closure H^T. Therefore, G is a cocomparability graph.
(if part)
If G is a cocomparability graph, we can construct an MIS-diagram for G in the following manner:
step 1: Make the complement G^C of G.
step 2: Give a transitive orientation to the edges of G^C.
step 3: Eliminate all redundant edges. Let V_{source} and V_{sink} be the set of sources and sinks, respectively, of the resultant digraph.
step 4: Create a new source s and add edge $(s \rightarrow x)$ for each $x \in V_{source}$. Similarly, create a new sink t and add edge $(y \rightarrow t)$ for each $y \in V_{sink}$. □

Since any interval graph is a cocomparability graph [9], this result is an extension of a portion of Ref. [4]. Note that the latter half of the proof of Theorem 4 is constructive. It is known that the computation of the transitive orientation (step 2) and the elimination of redundant edges (step 3) both can be executed in polynomial time [10, 11]. Therefore, an MIS-diagram of a cocomparability graph can be created in polynomial time. Then, we can generate efficiently all MIS's of the graph by finding all st-paths in the MIS-diagram.

1.4 Diagrams for maximum weight independent sets

Let G be a graph such that each vertex $v \in V(G)$ is given a positive weight $w(v)$. An independent set of G is called a maximum weight independent set (abbreviated as an MWIS) if the sum of weights of its vertices is the largest among all of the independent sets of G. We are to construct a diagram to represent all MWIS's (not MIS's) of G. We call such a diagram an MWIS-diagram for G. An example is depicted in Figure 4.

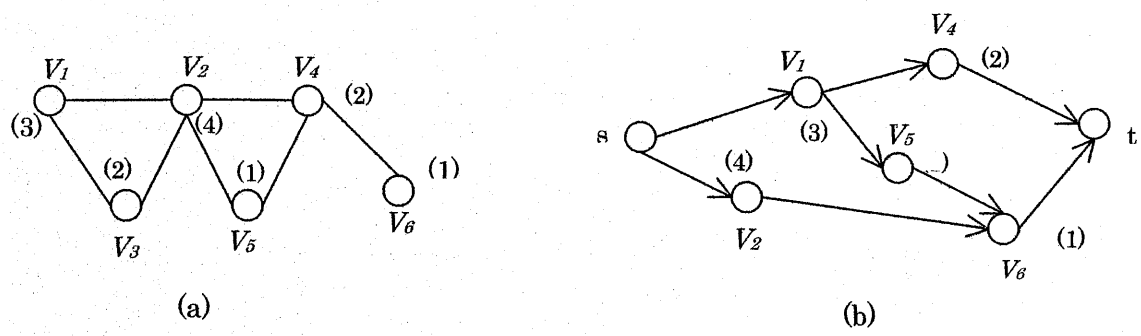


Figure 4: (a)A weighted graph G (the numbers in parentheses show vertex weights), and (b)an MWIS-diagram for G.

An MWIS-diagram does not always contain all vertices in $V(G)$.

As weights are all positive, any MWIS is an MIS. If an st-full MIS-diagram H is at hand, an MWIS-diagram can be obtained as follows.

We set the weights of s and t to be 0. For each vertex v, let $L(v)$ be the maximum among the sum of weights of vertices on a directed path from v to t in H.

Beginning with $L(t) = 0$, $L(\cdot)$ can be calculated easily because H is acyclic. Then eliminate all edges $(v \to u)$ such that $L(v) > L(u) + w(v)$. It is straightforward to prove that the resultant digraph is an MWIS-diagram. Therefore, the next Theorem follows from Theorem 1.4.

Theorem 1.5 *Suppose that each vertex in graph G has a positive weight. If G is a cocomparability graph, then it has an MWIS-diagram.* □

It should be noted that a graph which has an MWIS-diagram is not necessarily a cocomparability graph. See Figure 5.

1.5 Conclusion

In this section, we provided a necessary and sufficient condition for a digraph to be an MIS-diagram for an undirected graph, and then showed that an undirected graph G has an MIS-diagram iff G is a cocomparability graph. We also presented a simple method for obtaining MWIS-diagrams from MIS-diagrams for positively weighted cases. Based on these results, we can generate efficiently all MIS's or MWIS's of cocomparability graphs. As mentioned

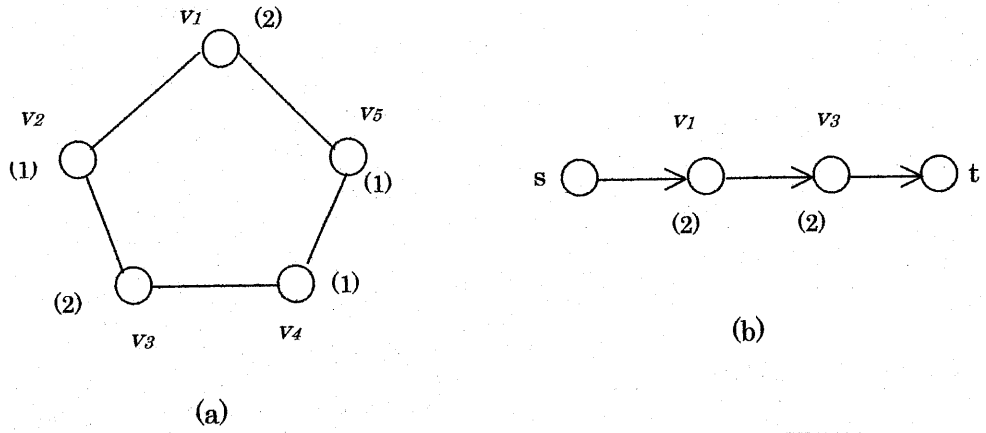


Figure 5: (a)A graph that is not a cocomparability graph, and (b)its MWIS-diagram.

before, every interval graph is a cocomparability graph. Thus, our results are extensions of the pertinent parts of Refs. [4] and [6].

The following problems are still open:

· What kind of digraphs can be MWIS-diagrams?

· What is a necessary and sufficient condition for a weighted graph to have an MWIS-diagram?

2 Algorithms for Generating Maximum Weight Independent Sets in Circle Graphs, Circular-Arc Overlap Graphs, and Spider Graphs

2.1 Introduction

An independent set of a graph is a set of vertices no two elements of which are adjacent. An independent set is called maximal if it is contained in no other independent set. An independent set with maximum cardinality is called a maximum independent set. For a vertex-weighted graph, a maximum weight independent set (abbreviated as MWIS) is an independent set which has the maximum weight among all independent sets. Generation of a certain family of sets is to put out all the sets in the family one by one without duplication.

A circle graph is an intersection graph defined on a set of chords in a circle: there exists an edge between two vertices iff their corresponding chords intersect. The circle graphs were introduced by Even and Itai in [13], and since then, many problems have been studied on them. Problems of maximum independent set, maximum clique, Hamiltonian circuit, and recognition can be solved in polynomial time for circle graphs [12, 14, 9, 10]. On the other hand, it is known that coloring and domination problems on circle graphs are NP-complete [16, 18].

Generation algorithms have been investigated for many years; cycle generation, cutset generation, maximal independent set generation, clique generation, etc. For maximal-independent-set generation of arbitrary graphs, the fastest algorithm to date takes $O(nm\alpha)$ time, where n is the cardinality of the vertex set, m the cardinality of the edge set, and α the number of maximal independent sets generated [1]. For particular graphs, much more efficient algorithms are known. Generation of maximal independent sets of an interval graph, of a circular-arc graph, and of a chordal graph can be done in time $O(n^2+\beta)$, $O(n^2+\beta)$, $O((n+m)\alpha)$, respectively, where n, m and α are the same as that mentioned above and β is the sum of the cardinalities of the maximal independent sets, i.e., the output size [4]. The former two algorithms run in time $O(\textit{polynomial of input size} + \textit{output size})$. As any generation algorithm takes $\Omega(\beta)$ time, such algorithms are, in a sense, optimal. we call such algorithms as pseudo-linear or output-sensitive. For interval graphs, an output-sensitive generation algorithm for MWIS's is also known[6]. For circle graphs, no output-sensitive generation algorithm has been known for maxmal independent sets. Furthermore, although an MWIS can be found efficiently by the same method given in [14], no efficient generation algorithm for MWIS's has been reported.

In this section we show that a simmple combination of the MWIS generation algorithm for interval graphs in [6] and the algorithm in [14] for finding a maximum independent set of a circle graph leads to an $O(n^3 + \beta)$ time algorithm to generate MWIS's of a circle graph with positive vertex weights. Spider graphs and circular-arc overlap graphs are super classes of circle graphs [7, 1]. We show that output-sensitive generation algorithms for MWIS's of these graphs can be obtained in a similar way.

2.2 Preliminaries

For a graph G with vertex weight attached to every vertex, the weight of a set of vertices is defined to be the sum of weights of vertices in the set. An induced subgraph of G defined on a set S of vertices is the subgraph of G obtained by deleting vertices other than S (and also deleting pertinent edges).

For a family F of sets, a graph G is called the intersection graph with (or of or on) model F if there exists a one-to-one correspondence between the vertex set of G and the family F such that two vertices are adjacent iff the corresponding sets have non-empty intersection. Similarly, for a family F of sets, a graph G is called the overlap graph of F if there exists a one-to-one correspondence between the vertex set of G and the family F such that two vertices are adjacent iff the corresponding sets overlap, i.e., have non-empty intersection and none contains the other.

An intersection graph of a family of chords in a circle is called a circle graph. The corresponding family of chords is called the circle model.

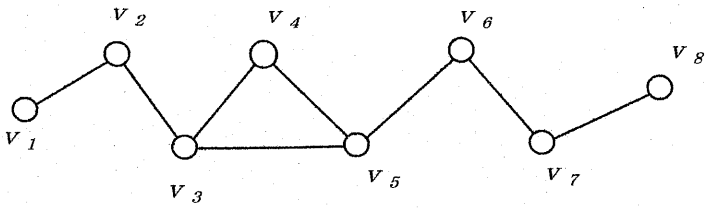
An intersection graph of a family of intervals on the real line (i.e., an interval intersection graph, so to say) is called an interval graph. Similarly, an overlap graph of a family of intervals on the real line is simply called an overlap graph. It is known that circle graphs are equivalent to overlap graphs [14]. It is shown in [14] that the size of a maximum independent set of an overlap graph is equal to the weight of an MWIS of an interval graph defined on the same interval model, with suitably defined vertex weight. Utilizing this fact, an MWIS of a circle graph, given its circle model, can be obtained by using an MWIS algorithm for interval graphs.

For a graph G, $V(G)$ denotes the set of vertices of G and $E(G)$ the set of edges. we call a digraph an st-dag (directed acyclic graph) if it is acyclic and has only one source s and only one sink t, where a source is a vertex with no incoming edges and a sink with no outgoing edges. An st-path is a directed path starting from s and ending at t. For an st-dag D, let $\mathcal{P}(D)$ denotes the set of st-paths on D, and let $\mathcal{S}(D) \stackrel{\text{def}}{=} \{$*the set of vertices on* $P -$

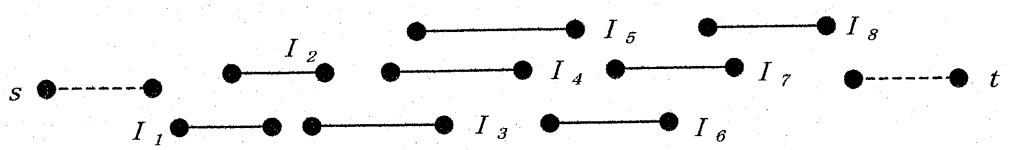
$\{s,t\} \mid P \in \mathcal{P}(D)\}$. we call D the diagram for $\mathcal{S}(D)$. With diagram D at hand, generation of $\mathcal{S}(D)$ is a quite easy task; note that $\mathcal{S}(D)$ can be generated efficiently by usual backtracking method, in time proportional to the output size. It is shown in [6] that, for an interval graph G, there exists a diagram for the family of MWIS's of G, and such a diagram can be obtained in $O(n^2)$ time, where n is the cardinality of $V(G)$. Below we briefly describe the method of [6] for convenience. Given a weighted interval graph G with model $\mathcal{I}$, first find a diagram for maximal independent sets of G by the method of [4] as follows: For each vertex u of G, let I_u denote the corresponding interval in $\mathcal{I}$. We add new interval I_s at the far left of $\mathcal{I}$ and new interval I_t at the far right, and temporarily consider s and t are vertices of the interval graph. For two vertices u and v in G, give a directed edge from u to v iff a) u and v are not adjacent, b) I_v is placed to the right of I_u in $\mathcal{I}$, and c) no interval which dose not intersect I_u nor I_v is placed between I_u and I_v.The resulting st-dag is a diagram for maximal independent sets of G. Then eliminate unnecessary edges so as to make the resulting directed graph become a diagram for MWIS's of G. This can be done because all weights are positive. For example, let G be the interval graph shown Figure 6 (a); the weight of v_5 is 2, weights of other vertices are all 1. Figure 6 (b) is a model for G, with imaginary intervals I_s and I_t added. Figure 6 (c) shows a diagram for maximal independent sets of G. And finally, Figure 6(d) shows a diagramfor MWIS's of G. Note that source s and sink t are not introduced in [14] and [6]. In what follows, every diagram we describe is that for MWIS's of some interval graph (say H). Thus we call such a diagram simply a diagram for H (omitting the family of sets it represents).

2.3 Generation of maximum weight independent sets of a circle graph

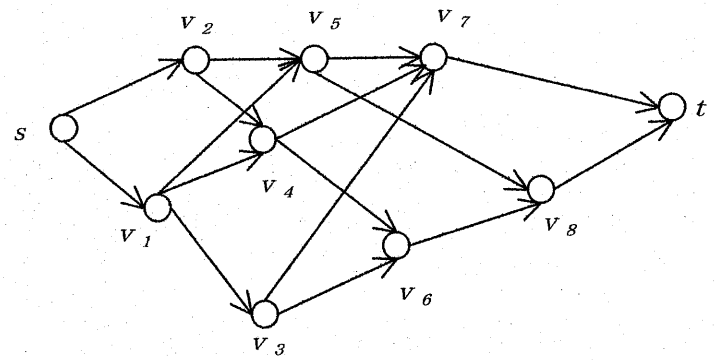
Given a circle graph, a corresponding circle model can be efficiently obtained [10]. As is stated before, a circle graph is an overlap graph. Given a circle model for the circle graph, an interval model for the corresponding overlap graph can be obtained efficiently [14]. In the following we are concerned mainly with interval models, so we will use the term overlap graph rather than circle graph. Thus, in this section, we assume that we are given an overlap graph G with positive vertex weight $wt(\cdot)$ and its interval model $\mathcal{I}$. Also we assume that all the end points of intervals are distinct. For convenience, we use "vertices" and the corresponding "intervals" interchangeably. For



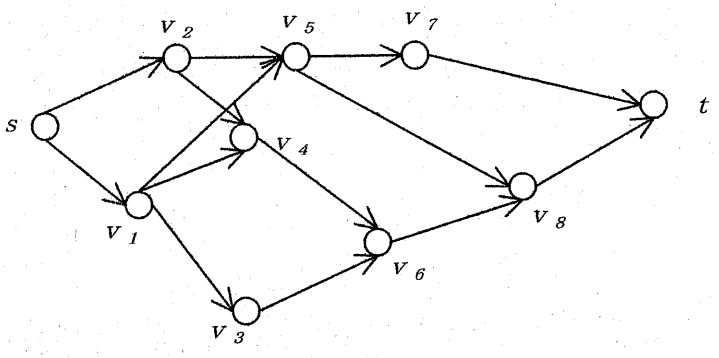
(a)



(b)



(c)



(d)

Figure 6: (a)An interval graph G, weight of v_5 is 2 and weights of other vertices are all 1, (b)model for G, with I_s and I_t added, (c)a diagram for maximal independent sets in G, and (d)a diagram for MWIS's of G.

example, the interval corresponding to vertex v will be called interval v, and for a set of vertices X, interval u in X means the interval corresponding to vertex u in X. The same applies for interval graphs.

An induced subgraph H of G is also an overlap graph. By deleting, from the model for G, intervals other than that corresponding to vertices in H, we obtain a model for H. In the following, by the term "model for an induced subgraph of G" we mean the model obtained in this way. Generally, for an induced subgraph H of G, let $\tilde{H}$ denote the interval graph defined on the same set of intervals that H has as its model. In particular, $\tilde{G}$ is the interval graph with model $\mathcal{I}$.

For a vertex v, let

$$cover(v) \stackrel{\text{def}}{=} \{u \in V(G) \mid \textit{interval } u \textit{ is contained in interval } v, u \neq v\}.$$

Let G_v denote the induced subgraph of G defined on $cover(v)$. Let $wt'(v)$ be a new weight of v defined as follows:

$$wt'(v) \stackrel{\text{def}}{=} wt(v) + \textit{the weight of an MWIS of } G_v.$$

Thus there are two kinds of weights, and MWIS's for these two weights will be considered in the following. To avoid confusion, MWIS with respect to weight $wt'(\cdot)$ is denoted by MWIS(wt'); MWIS for $wt(\cdot)$ is simply denoted by MWIS as before. For an independent set Q of G, the top-layer set of Q is defined to be the set $\{v \in Q \mid \textit{no interval } x \textit{ in } Q \textit{ properly contains interval } v\}$. Then the following two lemmas are immediate consequences of the definitions of $wt'(\cdot)$ and the top-layer set.

Lemma 2.1 *Suppose that $\{v_1, v_2, \cdots, v_k\}$ is an independent set of $\tilde{G}$. Then, $\{v_1, v_2, \cdots, v_k\} \bigcup \bigcup_i (\textit{an MWIS of } G_{v_i})$ is an independent set of G whose weight (with respect to $wt(\cdot)$) is $\sum_{i=1}^{k} wt'(v_i)$.*

Lemma 2.2 *The weight of an independent set Q of G (with respect to $wt(\cdot)$) does not exceed the sum of the weight $wt'(\cdot)$ of vertices in the top-layer set of Q.*

By these lemmas, we have the following Theorems. The proofs are rather straightforward and so are omitted. Unweighted version of Theorem 2.1 is in [14].

Theorem 2.1 *Suppose that $\{v_1, v_2, \cdots, v_r\}$ is an MWIS(wt') of $\tilde{G}$. Then, $\{v_1, v_2, \cdots, v_r\} \bigcup \bigcup_{i=1}^{r} (\textit{an MWIS of } G_{v_i})$ is an MWIS of G.*

Theorem 2.2 *Let Q be any MWIS of G, and let A be the top-layer set of Q. Then A is an MWIS(wt') of $\tilde{G}$, and, for any v in A, $Q \bigcap G_v$ is an MWIS of G_v.*

Theorem 2.3 *For two distinct MWIS(wt')'s X and Y of $\tilde{G}$, where $X = \{x_1, x_2, \cdots, x_r\}$ and $Y = \{y_1, y_2, \cdots, y_q\}$,*
$X \bigcup \bigcup_{i=1}^{r}$(an MWIS of G_{x_i}) cannot coincide with $Y \bigcup \bigcup_{j=1}^{q}$(an MWIS of G_{y_j}) for any choices of MWIS's of G_{x_i} and of G_{y_j}.

For a set X of vertices, let G_X denote the induced subgraph of G defined on the set $\{v \in V(G) \mid$ *some interval in X contains interval v*$\}$. Actually, in what follows, we only encounter G_X's such that X is an independent set in the interval graph $\tilde{G}$. Thus G_X will be some disjoint union of G_v's.

The above Theorems lead to the following algorithm, a rough sketch of which is as follows.

Procedure $M(H, S)$
    if $H = \emptyset$ then output S.
    else
        for each MWIS(wt') X of $\tilde{H}$
            $M(H_X, S \bigcup X)$

$M(G, \emptyset)$ generates the desired output.

Note That, in the procedure, $\tilde{H}$ is the interval graph corresponding to circle graph H.

To show the validity of the algorithm, we give one more trivial lemma.

Lemma 2.3 *Suppose that $X = \{v_1, v_2, \cdots, v_k\}$ is an independent set of $\tilde{G}$. If M is an MWIS of G_X, $M \cap V(G_{v_i})$ is an MWIS of G_v for $i = 1, 2, \cdots, k$. Conversely, $\bigcup_{i=1}^{k}$(an MWIS of G_{v_i}) is an MWIS of G_X.*

Now, recursive use of Theorem 2.1 and Lemma 2.3 ensures that each output is an MWIS of G. Recursive use of Theorem 2.2 and Lemma 2.3 ensures that every MWIS of G is put out by the algorithm. Finally, Theorem 2.3 ensures that there are no duplicates.

Next,we describe some implementation detail for efficient execution of the algorithm. Let D be the diagram for $\tilde{G}$ (with respected to $wt'(\cdot)$). Similarly, let D_v be the diagram for $\tilde{G}_v$ (with respect to $wt'(\cdot)$). $wt'(\cdot)$ for all vertices are computed, and diagrams D and D_v's are constructed prior

to the execution of the algorithm, in preprocessing phase. The sentence in the algorithm for each MWIS(wt') X of $\tilde{H}$ is replaced by the following three sentences:

construct a diagram for $\tilde{H}$
for each st-path P on $\tilde{H}$
$X \leftarrow$ (vertex set on P) $\cap\, V(G)$

Construction of a diagram for $\tilde{H}$ is done as follows. When $H = G$, D is already constructed. Otherwise, $H = G_X$ for some X that is an independent set in G. Consequently, $\tilde{H}$ is the disjoint union of interval graphs $\{\ \tilde{G}_v \mid v \in X\ \}$. Thus, a diagram for $\tilde{H}$ can be obtained by combining diagrams D_v's ($v \in X$) in series. The time complexity of making such a series combination is dominated by the size of output set. With these devices, the time needed to generate all MWIS's of G is $O(\beta)$ where β is the output size, except for preprocessing phase. To compute $wt'(\cdot)$ and to construct D and D_v's, $O(n^3)$ time suffices [14], [6]. The overall time complexity is $O(n^3 + \beta)$, space complexity $O(n^3)$.

2.4 For spider graphs and circular-arc overlap graphs

A hyperchord in a circle is an arbitrary connected figure in a circle with more than one touching points (i.e., intersecting points with the enclosing circle). Thus a chord is also a hyperchord. we assume that unnecessary intersection between hyperchords has been eliminated: two hyperchords intersects iff their touching points interleave (as in a circle graph). A spider graph is the intersection graph of such hyperchords in a circle.

A circular-arc overlap graph is an overlap graph of arcs on a circle. Without loss of generality, we assume that no arc covers the whole circle. Spider graphs and circular-arc overlap graphs are super classes of circle graphs. MWIS generation algorithm for circle graphs described so far can be utilized to obtain output-sensitive MWIS-generation algorithms for spider graphs and for circular-arc overlap graphs. we assume that the graphs are given with the corresponding models. Algorithm $M(G, \emptyset)$ can be thought of to be an MWIS-generation algorithm for a circular-arc overlap graph, with a slight modification: MWIS(wt') generation of $\tilde{G}$ (executed in the first call for procedure $M(H, S)$, i.e., when $H = G, S = \emptyset$) is done by the MWIS-generation algorithm for circular-arc graphs described in [6] rather than that for interval graphs.

An MWIS-generation algorithm for circular-arc graphs is reported in [6]. Unfortunately, the description is ambiguous and seems to have a flaw. The

algorithm seems to construct one single directed graph (diagram, in our term) and generate all paths. This cannot lead to a correct generation algorithm: for example, consider a cycle of five vertices with all weights 1. Instead, we cite [5] which describes an efficient algorithm for generating maximum cardinality independent sets of a circular-arc graph, which can be easily converted to an MWIS-generation algorithm for a circular-arc graph, whose running time is $O(n^2+\beta)$. Thus, the resulting algorithm is of $O(n^3+\beta)$ time.

In [14] a method is described for converting a circle model of a circle graph to a (linear) interval model of an overlap graph: put a light source on the (enclosing) circle and project each chord onto the tangent line (on the opposite side of the circle) by the light. Apply the same method to the hyperchord model for the spider graph to obtain a linear model. Note that a hyperchord v becomes an interval which may contain several "inner points" which are projected-images of touching points of v in the original model (two of the images of the touching points become the endpoints of the interval). Then, consider $\tilde{H}$ to be the interval graph defined on the intervals (neglecting inner points), and for a set of vertices X, H_X to be the induced subgraph of G on the vertex set
$\{v \in V(H) \mid X$ *has* x *such that interval* x *contains interval* v *and interval* v *does not contain any "inner points" of interval* $x\}$.

By this modification, algorithm $M(G, \emptyset)$ certainly generates MWIS's of the spider graph G. The running time is $O(n^3 + \beta)$, i.e.,output-sensitive.

2.5 Conclusions

In this section we have shown that the DP algorithm in [14] can be extended to be an output-sensitive generation algorithm for MWIS's of a circle graph. We have extended the result further to obtain output-sensitive generation algorithm for MWIS's of a spider graph and of a circular-arc overlap graph. Chordal graphs admit similar output-sensitive generation algorithm for MWIS's, which will be described in separate paper.
No output-sensitive generation algorithm have been obtained so far for maximal independent sets of a circle graph and of a chordal graph. These problems deserve further study.

3 A Competitiveness Coefficient of Online Edge-Coloring Algorithm

3.1 Introduction

Given an undirected graph $G(V, E)$, an edge-coloring is an assignment of indices $1, 2, \cdots, n$ to the edges of G such that no two edges incident on a vertex have the same label. The indices are referred as colors, and the smallest value of n for which such a coloring can be achieved is called the chromatic index of the graph. The online coloring algorithm is defined as follows. It is not given all of input data in advance. As soon as it is given an input data, it colors the input data one after another. It is not given any further information of input after it colors. In general they use a competitive analysis to analyze the online algorithm, because an absolute performance for the online algorithm dose often not make sense. This is to compare cost of online algorithm with that of offline algorithm for the same family of input data. The offline algorithm is an algorithm that deals with a family of input data all of which are known in advance. In this section I consider an edge-coloring game of a graph by two persons who are an adversary and an edge-painter. The former strategy is to add or to delete edges in the graph successively. The strategy is called input. Since the adversary makes a family of input. On the other hand, as soon as an edge is added in the graph, the latter has to color the edge. Then it is never changed the color of edges colored. The latter is not given any further information of the adversary's. We call the coloring strategies of the painter online edge-coloring algorithm. A maximum ratio which is a ratio between the number of colors used of an online edge-coloring algorithm A for adversary's family of input and that of offline edge-coloring algorithm is called a competitiveness coefficient of an online edge-coloring algorithm A. It is better algorithm that the online edge-coloring algorithm has smaller value of this coefficient. We analyze a deterministic online edge-coloring algorithm and a randomized online edge-coloring algorithm under the following cases. The painter is allowed arbitrary amount of colors and has to color all of edges added in the graph. For this case the online edge-coloring algorithm was analyzed [20]. In [20], the competitiveness coefficient of arbitrarily deterministic online edge-coloring algorithm was proved that it was greater than $\frac{2k-1}{k}$ where k was maximum degree of the graph. But for randomized online edge-coloring algorithm the outcome was only shown briefly, however the proof was imperfect. In this section we have proved that the competitiveness coefficient of arbitrary randomized online edge-coloring algorithm is greater than $\frac{2k-1}{k}$ where k is maximum degree.

3.2 Preliminaries

Here we wish to refer to some notations about graph in [19], and the graph is undirected and is allowed to have multiple edges except on special occasions.

Theorem 3.1 *The chromatic number of arbitrary graph G is equal to the maximum degree of the graph or is greater than it [19].*

Theorem 3.2 *The chromatic number of bipartite graph G is equal to the maximum degree of the graph [19].*

We consider a two persons game by an edge-painter and an adversary. Let Π be a problem with a finite set I of input instances (fixed size), and a finite set of deterministic algorithms A. For input $i \in I$ and algorithms $a \in A$, let $C(i, a)$ denote the cost of algorithm a on input i. An edge-painter wants to make algorithms whose cost is smaller and an adversary wants to make input instances whose cost is larger. For probability distributions p over I and q over A, let i_p denote a random input chosen according to p and a_p denote a random algorithm chosen according to q.

Theorem 3.3 (Yao's Minimax Principle [23]) *For all distributions p over I and q over A,*

$$\min_{a \in A} E[C(i_p, a)] \leq \max_{i \in I} E[C(i, a_q)]$$

where $E[X]$ is expectation of X.

There are an oblivious adversary and an adaptive adversary for adversary of randomized online edge-coloring algorithm. When the former dose input, he cannot get the information that edge-painter had chosen colors for family of input before. When the latter dose input, he can get the information that edge-painter had chosen colors for family of input before. Since the adaptive adversary is stronger than the other. Let a graph $G_0 = (V, E, k)$ be the beginning of a graph G, $|V| \ll k$, $E = 0$. Adversary adds or deletes edges to G_0 successively, under the following conditions : the degree of all vertices are always less than k on the way of inputting and once at least the degree of a vertex is just k on the way of inputting. The adversary knows the strategies of the edge-painter. When the adversary adds an edge, the edge-painter colors the edge then the color is different from each of adjacent edges'. When the adversary deletes edges, the edge-painter dose not anything. The edge-painter never changes the color of the edge colored, and the adversary knows

the limit of degree of the graph. The edge-painter's aim is to make small using the number of colors according to the families of input of the adversary. The adversary's aim is to give the families of input that the ratio between using colors of online edge-coloring algorithm and that of offline edge-coloring algorithm is made large. Let $G_{e_1,e_2,\cdots,e_i}$ be a graph just after input e_i according to the family of input $e_1, e_2, \cdots, e_i, \cdots$. Let $f_A(e_1, e_2, \cdots, e_M)$ be the number of using colors of deterministic online edge-coloring algorithm A according to the family of input $e_1, e_2, \cdots, e_M$, and $f_O(e_1, e_2, \cdots, e_M)$ be the number of using colors of optimal offline edge-coloring algorithm according to the family of input $e_1, e_2, \cdots, e_M$.

Definition 3.1 *A deterministic online edge-coloring algorithm A is said to be C-competitive such that for any families of input $e_1, e_2, \cdots, e_M$,*

$$f_A(e_1, e_2, \cdots, e_M) - C \times f_O(e_1, e_2, \cdots, e_M) \leq 0.$$

And the competitiveness coefficient of A, denoted C_A^{det} , is the infimum of C such that A is C-competitive.

Oblivious adversary is the same that he makes all of families of input previously, because he has no influence that edge-painter colored edges before. Let $f_R(e_1, e_2, \cdots, e_M)$ be the number of using colors of randomized online edge-coloring algorithm R according to the family of input $e_1, e_2, \cdots, e_M$, then the $f_R(e_1, e_2, \cdots, e_M)$ is the random variable.

Definition 3.2 *For oblivious adversary a randomized online edge-coloring algorithm R is said to be C-competitive such that for any families of input $e_1, e_2, \cdots, e_M$,*

$$E[f_R(e_1, e_2, \cdots, e_M)] - C \times f_O(e_1, e_2, \cdots, e_M) \leq 0.$$

And the competitiveness coefficient of R, denoted C_R^{obl}, is the infimum of C such that R is C-competitive.

Let P be a probability distribution of choosing families of input.
The $f_A(e_1, e_2, \cdots, e_M)$ and the $f_O(e_1, e_2, \cdots, e_M)$ are random variables under P. For a deterministic online edge-coloring algorithm A, let C_A^P be competitiveness coefficient under P and it is infimum of C such that

$$E[f_A(e_1, e_2, \cdots, e_M)] - C \times E[f_O(e_1, e_2, \cdots, e_M)] \leq 0.$$

Yao's Minimax Principle implies that

$$C_R^{obl} \geq \min_A C_A^P.$$

The implication of this is as follows : the competitiveness of the best deterministic online edge-coloring algorithm C_A^P, according to the probability distribution P of choosing the worst-case families of inputs, is the infimum of C_R^{obl} of any randomized online edge-coloring algorithm according to the oblivious adversary.

Definition 3.3 *For an adaptive adversary a randomized online edge-coloring algorithm R is said to be C-competitive such that for the family of input $e_1, e_2, \cdots, e_M$ made by the adaptive adversary.*

$$E[f_R(e_1, e_2, \cdots, e_M)] - C \times E[f_O(e_1, e_2, \cdots, e_M)] \leq 0.$$

And the competitiveness coefficient of R, denoted C_R^{ada}, is the infimum of C such that R is C-competitive.

Definition 3.4 *An online edge-coloring algorithm which dose not use greater then h colors as much as possible is defined as follow : when an adversary adds the edge e_i, the algorithm must color with the one of colors, (they are $color_1, color_2, \cdots, color_{h-1}$) which are not used yet on the adjacent edges to e_i, in $G_{e_1,e_2,\cdots,e_i}$. Color in this section has linear order as follow $color_1 < color_2 < \cdots$. If new color is used, it must be smallest number that is not yet used. A set of colors on edges incident from a vertex v is denoted Sv.*

3.3 Online edge-coloring problem

3.3.1 Deterministic online edge-coloring algorithm

Lemma 3.1 *Let A be a deterministic online edge-coloring algorithm. For this A, it is*

$$f_A(e_1, e_2, \cdots, e_M) \geq 2k - 1$$

according to the family of inputs $e_1, e_2, \cdots, e_M$.

Proof *We prove it by reductio ad absurdum. We assume that it is*

$$f_A(e_1, e_2, \cdots, e_M) < 2k - 1$$

according to any family of input for A. Here we think that the family of input is as follow.
1. Repeat $k-1$ times adding (v_1, w_1).

2. Repeat $k-1$ times adding (v_2, w_2).

$$\vdots$$

By the assumption, $S_{v_1} \cup S_{v_2} \cup \cdots$ is a subset of $\{ color_1, color_2, \cdots, color_{2k-2} \}$. A subset of $\{ color_1, color_2, \cdots, color_{2k-2} \}$ is finite, since there exist adequate sequence $x_1, x_2, \cdots, x_k$ and $S_{v_{x_1}} = S_{v_{x_2}} = \cdots = S_{v_{x_k}}$ if this repeating are continued enough. And then this pattern is disconnected. After that, next following edges are added.
3. Add (v_{x_1}, w_0).
4. Add (v_{x_2}, w_0).

$$\vdots$$

5. Add (v_{x_k}, w_0).
Since it cannot color with the element of S_{x_j} on edges added form 3 to 5, these need k colors without $k-1$ colors element of S_{x_j}. For this family of input, total of colors used is as follow :
(colors between v_{x_j}, and w_{x_j}) + ((v_{x_1}, w_0), (v_{x_2}, w_0),$\cdots$,(v_{x_k}, w_0))= $(k-1)$+k=$2k-1$.
But this is the contradiction of the assumption :

$$f_A(e_1, e_2, \cdots, e_M) < 2k-1.$$

Hence it is

$$f_A(e_1, e_2, \cdots, e_M) \geq 2k-1.$$

□

On the above step2, the continuing enough is the maximum

$$\begin{pmatrix} 2k-1 \\ k-1 \end{pmatrix} \times (k-1) + 1,$$

and the maximum number of vertices, for making the families of inputs of the above, is

$$\left(\begin{pmatrix} 2k-1 \\ k-1 \end{pmatrix} \times (k-1) + 1 \right) \times 2 + 1.$$

Theorem 3.4 *Let A be a deterministic online edge algorithm. Then*

$$C_A^{det} \geq \frac{2k-1}{k}.$$

Proof *We prove it by reductio ad absurdum. We assume that it is $C_A^{det} < \frac{2k-1}{k}$. By the assumption, it must be made up a following equation (1) for any family of inputs $e_1, e_2, \cdots, e_M$,*

$$f_A(e_1, e_2, \cdots, e_M) - C_A^{det} \times f_O(e_1, e_2, \cdots, e_M) \leq 0. \qquad (1)$$

Let $e_1, e_2, \cdots, e_M$ be a family of input which are used by $2k-1$ colors-algorithm of lemma 3.1. Since $G_{e_1,e_2,\cdots,e_i}$ is bipartite graph, the chromatic number is k from Theorem 3.2. Since these only adds edges, it is $f_O(e_1, e_2, \cdots, e_M) = k$ for offline edge-coloring algorithm because of that edge-painter can color with the colors which are colored k-edge-coloring, (such that G has k colors to color edges), to $G_{e_1,e_2,\cdots,e_i}$. These values are substituted for the left hand of equation (1). Then the left hand of equation (1) is contradiction of the assumption :

$$(2k-1) - C_A^{det} \times k > (2k-1) - (2k-1) = 0$$

Hence

$$C_A^{det} \geq \frac{2k-1}{k}.$$

□

3.3.2 Randomized online edge-coloring algorithm

Theorem 3.5 *Let R be an randomized online edge algorithm. Then $C_R^obl \geq \frac{2k-1}{k}$.*

Proof *For proving this Theorem we apply Yao's Minimax Principle to the competitiveness coefficient. we prove it by reductio ad absurdum. Let C_R^{obl} be any randomized online edge algorithm R. We assume that*

$$C_R^{obl} < \frac{2k-1}{k}.$$

From Yao's Minimax Principle, there exist a deterministic online edge-coloring algorithm according to any families of inputs chosen of a probability distribution. Then we think a family of input chosen of a probability distribution as follow,

1. Repeat $k-1$ times adding (v_1, w_1).
2. Repeat $k-1$ times adding (v_2, w_2).

$\vdots$

By the assumption, $S_{v_1} \cup S_{v_2} \cup \cdots$ is a subset of $\{color_1, color_2, \cdot, color_{2k-2}\}$. A subset of $\{color_1, color_2, \cdot, color_{2k-2}$ is finite, since there exist adequate sequence $x_1, x_2, \cdots, x_k$ and $S_{v_{x_1}} = S_{v_{x_2}} = \cdots = S_{v_{x_k}}$ if this repeating are continued enough.
3. After that, choose k vertices in the same probability from $v_1, v_2, \cdots$, and add edges from k vertices chosen to w_0.
4. Delete all of edges added by steps from 1 to 3.
5. Repeat enough times from step1 to step4.

Then these colors on the $k-1$ edges, which come out from k vertices chosen, are all consisted for any deterministic online edge-coloring algorithm. Hence the k vertices chosen are all consisted with $v_{x_1}, v_{x_2}, \cdots, v_{x_k}$. Since these edges are not colored with the elements of Sv_{x_j} , we need k colors to color these edges except for the $k-1$ colors elements of Sv_{x_j} . Then the total of colors used is as follow :
(colors between v_{x_j}, and w_{x_j}) + $((v_{x_1}, w_0), (v_{x_2}, w_0), \cdots, (v_{x_k}, w_0))$= (k-1)+k = 2k-1.
For any families of inputs chosen of a probability distribution, the expectations of colors used of any deterministic online edge-coloring algorithm is $2k-1$. And then we study the expectation of colors used of offline edge-coloring algorithm for those families of inputs chosen of a probability distribution, and we pay attention to the graph that is made by step1 to 3 in first time. The chromatic number is k from Theorem 3.2, since these graphs made are bipartite graphs for any families of inputs chosen. As inputs from step1 to 3 are only adding edges, it is enough to use the colors from $color_1$ to $color_k$ by k-edge-coloring if all of families of inputs are known. As it returns to the beginning at step4, , it is enough to use the colors from $color_1$ to $color_k$ for adding edges from step1 to 3 after second stage. Therefore the expectation of necessary colors is k if it is offline edge-coloring algorithm according to like these families of inputs chosen of a probability distribution. Under like these families of inputs chosen of a probability distribution P, any deterministic online edge-coloring algorithm is satisfied with next equation :

$$E[f_A(e_1, e_2, \cdots, e_M)] - C_A^P \times E[f_O(e_1, e_2, \cdots, e_M)] \leq 0.$$

We substitute the above result for the equation, however, it is as follow:

$$\text{Left\_side} = 2k - 1 - C_A^P \times k > 2k - 1 - \frac{2k-1}{k} \times k = 0.$$

This is the contradiction of the assumption, hence

$$C_R^o bl \geq \frac{2k-1}{k}.$$

□

On the above step2, the continuing enough is the maximum

$$\begin{pmatrix} 2k-1 \\ k-1 \end{pmatrix} \times (k-1) + 1 (= X \text{written})$$

as same as the deterministic online edge-coloring algorithm, and the maximum number of vertices, for making the families of inputs of the above, is $\left(\begin{pmatrix} 2k-1 \\ k-1 \end{pmatrix} \times (k-1) + 1\right) \times 2 + 1$. Furthermore repeating at step5 are enough $\begin{pmatrix} X \\ k \end{pmatrix}$. For a randomized online edge-coloring algorithm R there is the relation of $C_R^{obl} \leq C_R^{ada}$ between C_R^{obl} and C_R^{ada} from the definition. Let C^{obl} be the minimum of competitiveness coefficient for all randomized online edge-coloring algorithm according to the oblivious adversary, let C^{ada} be the minimum of competitiveness coefficient for all randomized online edge-coloring algorithm according to the adaptive adversary, and let C^{det} be the minimum of competitiveness coefficient for all deterministic online edge-coloring algorithm. There are the relation of $C^{obl} \leq C^{ada} \leq C^{det}$. Here is $C^{det} = \frac{2k-1}{k}$ from Theorem3.4 and Theorem3.5.

Lemma 3.2 *Let B be an online edge-coloring algorithm which dose not use greater then h colors as much as possible. The B only uses the colors from $color_1$ to $color_k$ for any family of inputs $e_1, e_2, \cdots, e_M$,.[2]*

Proof *We prove it by reductio ad absurdum. We assume that the algorithm B colors the $color_{2k}$ to some edges. Let $e = (v_i, v_j)$ be the edge that were colored $color_{2k}$ at first time. We consider the e colored just before at this time. We pay attention to the colors of edges going out from v_i to v_j , then they must be colored all colors from $color_1$ to $color_k$ by the property of the algorithm B. This is that the summation of degree of v_i and v_j must be at least $2k-1$. But as the degrees of each vertex are not greater then k, at least the degree of either v_i or v_j is k. When the edge e is added at this condition,*

the degree that is already k becomes $k+1$. This is the contradiction of the role of adversary's input. Therefore the algorithm B dose not use the $color_{2k}$ to some edges.

□

From Lemma3.2 , Thorem3.1, Theorem3.4, and Theorem3.5, the competitiveness coefficient of an online edge-coloring algorithm which dose not use greater then $h(h \leq 2k)$ colors as much as possible is $\frac{2k-1}{k}$.

3.4 Conclusion

W have described that the competitiveness coefficient of any online edge-coloring algorithm is greater then $\frac{2k-1}{k}$, where k is maximum degree, for deterministic or randomized online edge-coloring algorithm. We will present new algorithm for online edge-coloring algorithm in next section.

4 On-line edge-coloring algorithms for degree bounded bipartite graphs

4.1 Introduction

Vertex- or edge- coloring of a graph is to assign colors to vertices or edges in such a way that no two vertices or edges with the same color are adjacent. There seem to exist two types of coloring problems: "assignment maximization type" in which the number of available colors is prespecified and the objective is to maximize the number of vertices or edges that are assigned colors, and "color minimization type" in which the objective is to minimize the number of colors used to color all vertices or edges.

In resent years, online versions of existing problems have been considered, among them are the online versions of these coloring problems.

Competitive analysis of vertex coloring problems of both color minimization type and of assignment maximization type are reported [20], [21].

Competitive analysis on edge coloring problems of assignment maximization type seems not to have been reported. In this section we concentrate on online edge coloring problems of this type in which the graph is subjected to operations of both edge addition and edge deletion such that at any time the graph has some special properties: 1) the graph is bipartite with prespecified bipartition, and 2) maximum degree of vertices is not more than a prespecified integer k. Such a property is related to assignment problems on some switching network. We consider only the case in which the number of available colors is equal to k.

Assignment of colors to edges is done under the restriction that an edge can be (irrevocably) colored only at the time it is added. The added edges can be left uncolored. Colors once assigned cannot be changed until the edge is deleted. An edge which is decided to remain uncolored at the time of its addition can not given colors afterward (until the edges is deleted).

The performance of the overall output is defined to be the total number of coloring operations actually done.

As deterministic algorithm have poor competitiveness coefficients, we concentrate on randomized algorithms. After showing that earnest (or diligent) randomized algorithms have poor competitiveness coefficients, we propose two lazy randomized algorithms: one with competitiveness coefficient $\frac{1}{4}$, and one with larger competitiveness coefficient (which is between $\frac{1}{4}$ and $\frac{1}{2}$) but needs more random bits than the former(one random bit per one edge addition on the fly)irrevocably.

4.2 Preliminaries

We are given two sets of vertices U, V, $U \cap V = \emptyset$, empty initial edge set, a finite integer k, and a finite sequence $L = (t_1, t_2, \cdots)$, of operations (edge addition or deletion) such that, at any time (that is , after operations $t_1, t_2, \cdots, t_j$ are executed in this order, for any j), the graph is a bipartite graph with bipartition (U, V), with degree-bound satisfied: degree of each vertex is less than or equal to k . Multiple edges are allowed.

We are to assign colors to edges in the manner described below. When an edge is added, a color can be irrevocably assigned, and at any time, no two edges with the same color can be adjacent to each other, where "irrevocably" means that, until the edge deleted, the color cannot change (including from colorless to colored, and from colored to colorless). The added edge can remain colorless. (colorless edges can be adjacent to each other). The measure of performance (or simply performance) we are to maximize is the number of the times edge colorings are performed (i.e., the number of coloring operations performed).

For notational simplicity, edges between vertices u and v are denoted simply by (u, v). For example, we say "add 3 (u, v) edges" to describe adding 3 different edges having the same end vertices u and v. Such a description will cause no confusion.

The following Theorem is fundamental for our problem.

Theorem 4.1 *[22] For a bipartite graph, chromatic index is equal to the maximum degree, where chromatic index is the minimum number of colors needed to color all the edges of the graph.* □

For a random variable X , the expectation of X is denoted by $\mathcal{E}(X)$.

We call a coloring algorithm to be online if the algorithm has to execute coloring not knowing the operations (edge addition or deletion) to follow, while an algorithm is called offline if the algorithm can know the whole input at the beginning of the coloring of the first edge.

For arbitrary algorithm H, let $f_H(L)$ denote the performance of algorithm H for input sequence L, and let $f_O(L)$ denote the performance of offline optimal algorithm for input sequence L.

A deterministic algorithm D is said to be C-competitive if there exists a constant b such that for any (finite length) input sequence L, $f_D(L) - Cf_O(L) \geq b$, where b is a constant independent of $|L|$ (the length or cardinality of L). b may be dependent on $|U|, |V|$ and k. The competitiveness coefficient of D is the supremum of C such that D is C-competitive.

Also for a randomized algorithm R, let $f_R(L)$ denote the performance of algorithm R for input sequence L. Now $f_R(L)$ is a random variable. A

randomized algorithm R is said to be C-competitive against an oblivious adversary if for any input sequence L, $\mathcal{E}(f_R(L)) - Cf_O(L) \geq b$, where b is a constant independent of $|L|$ (may be dependent on $|U|, |V|, k$). The oblivious competitiveness coefficient is defined to be the supremum of C such that R is C-competitive. In the following, as we are concerned only with oblivious competitiveness coefficients, the term "oblivious" will be omitted.

Let P be a probability distribution on the input to a problem. For a deterministic algorithm D for the problem, define competitiveness of D under P, C_D^P, to be the supremum of C such that $\mathcal{E}(f_D(L)) - C\mathcal{E}(f_O(L)) \geq b$, for a constant b independent of $|L|$.

The next Theorem is used to demonstrate an upper bound for competitiveness coefficients for our problem.

Theorem 4.2 *[23]*
$sup_R C_R = inf_P sup_A C_D^P$ □

As the case for $k \leq 1$ is trivial, we assume that $k \geq 2$.

4.3 Upper bound

In this section, we give a rather weak upper-bound of competitiveness coefficient for the edge coloring problem we study.

Theorem 4.3 *For any randomized online edge coloring algorithm R, $C_R \leq \frac{X-1}{X}$, where $X \stackrel{def}{=} \frac{(k+1)k((k^2+1)}{2}$.*

Proof *Suppose that there exists a randomized algorithm R such that $C_R = \frac{X-1}{X} + \delta$ for $\delta > 0$.*

Theorem 4.2 tells us that for any probability distribution P imposed on the input, there exists a deterministic algorithm D such that
$C_D^P \geq \frac{X-1}{X} + \delta$.

I set P as follows. The input (i.e., sequence of operations) we am describing consists of rounds. Each round is described as follows.

For $i = 1, 2, \cdots, k+1$ do the following: add edge (u_i, v_i) consecutively $k-1$ times. Then, select u^* among $u_1, u_2, \cdots, u_{k+1}$ uniformly at random. Add edge (u^*, v_{k+2}). Further, select u^{**} among $\{u_1, u_2, \cdots, u_{k+1}\} - u^*$ uniformly at random, and add edge (u^{**}, v_{k+2}). Finally delete all edges thus far added, one by one. This completes one round.

It is easy to show that after $\frac{k(k+1)}{2}$ rounds, for any deterministic algorithm D', the expected number of miss (i.e. the number of times edges are left uncolored) of D' is greater than or equal to 1, while offline algorithm can

color all edges due to Theorem 4.1 and to the manner of edge additions and deletions. Thus, iterating the round sufficiently many times, we have that $C^P_{D'} \leq \frac{X-1}{X}$, *a contradiction.*

□

We call an algorithm earnest (or diligent) if it colors added edges whenever possible (of course at the time of edge addition).

We show that earnest algorithms have poor competitiveness coefficients.

Theorem 4.4 *For any earnest algorithm* R,
$C_R = 0$.

Proof *We show that earnest algorithms (with all randomization) are surely guided to undesirable situations which offline algorithms can avoid.*

Consider the following input: (only 6 *vertices are involved)*
Add (u_1, v_1) $k-2$ *times, add* (u_1, v_2) *two times,*
add (u_2, v_2) $k-2$ *times, add* (u_2, v_3) *two times,*
add (u_3, v_3) $k-2$ *times, delete all the edges* (u_1, v_2) *and* (u_2, v_3),
add one (u_1, v_1) *(call this edge* e^* *for later reference), add one* (u_1, v_2),
add one (u_2, v_1), *add one* (u_2, v_3), *delete* $e*$.

Now any earnest algorithm has colored edges in such a way that colors assigned to edges incident to u_1 *and colors assigned to edges incident to* v_3 *together existing all the colors available.*

Input continues as follows: add (u_1, v_3) *(call this edge* e^{**}*), delete* e^{**}, *add* e^{**}, *delete* e^{**},...

Any earnest algorithm cannot color e^{**}, *thus missing arbitrarily large number of times. On the other hand an offline algorithm can (non-earnestly) ignore preceding edges and concentrate on* e^{**}, *resulting in only a constant number of misses.*

□

4.4 Lazy Algorithm I

The previous section concluded that earnest algorithms are poor in competitiveness coefficients. In this section and in the section to follow, we give lazy algorithms whose competitiveness coefficients are not 0.

This section deals with the one which requires fewer random bits than the algorithm given later, with the cost of less competitiveness coefficients.

The algorithm will color edges with laziness of $\frac{3}{4}$, that is, will try to color each added edge with probability $\frac{1}{4}$, assuring that the intended coloring is always possible. To facilitate such a scheme, each vertex $w \in U \cup V$ is attached

a random vector A_w of size k which is defined as follows. Let h_1 be the vector $(1,1,...,1,0,0,...,0)$ (the number of 0's is $\lfloor \frac{k}{2} \rfloor$ and the number of 1's are $\lceil \frac{k}{2} \rceil$) and let $\overline{h_1}$ be the complement of h_1, that is, $\overline{h_1} = (0,0,...,0,1,1,...,1)$.

Next, we present an algorithm with competitiveness coefficient $\frac{1}{4}$ in which the number of random bits used is equal to the number of vertices (independent of the number of edge additions).

For each $w \in U \cup V$, A_w is chosen to be either h_1 or $\overline{h_1}$ equally probably and independently of the choice for other vertices.

Also each vertex w is attached a (not random) vector B_w of dimension k, each entry of the vector being either "marked" or "nonmarked".

The algorithm is as follows.

Algorithm A_1

Addition: If the operation is addition of edge (u,v), then select a "nonmarked" entry, say the a-th entry, in B_u, and turn it "marked". Also select an "nonmarked" entry, say the b-th entry, in B_v, and turn it "marked". If the a-th entry of A_u and the b-th entry of A_v are both 1, then "try to color" the currently added edge (u,v). Else decide to leave the edge uncolored.Note that degree-boundedness implies that there exist nonmarked entries in B_u and B_v.

Deletion: If the operation is deletion of edge (u,v), then change the pertinent entries of B_u and B_v from "marked" to "nonmarked", where pertinent entries of B_u and B_v are the ones marked by the currently deleted edge (u,v) (at the time of edge addition). □

Clearly, the number of random bits needed is $|U \cup V|$ independent of the number (times) of edge additions.

Note that selection of nonmarked entry in B_w is arbitrary as long as it does not depend on the choice of h_1 or $\overline{h_1}$. Selecting the entry with the least index among nonmarked entries will do.

Lemma 4.1 *For edge addition, the probability that the case "try to color" happens is* $\frac{1}{4}$.

Proof *The proof is by counting the number of cases as follows. Let* b_j *be the j-th operation and be addition of edge* (u,v). *Let* a^**-th entry and* b^**-th entry of* B_u *and* B_v*, respectively, are the selected entries for operation* b_j *(addition of* (u,v)*).* a^* *and* b^* *are fixed for all choises of* h_1 *and* $\overline{h_1}$ *for* A_w*'s. Thus the number of cases of* $A_u(a^*) = A_v(b^*) = 1$ *is one fourth of the number of all possible cases.*

□

Theorem 4.5 $C_{A_1} = \frac{1}{4}$.

Proof *As "try to color " case happens with probability $\frac{1}{4}$, competitiveness coefficient cannot exceed $\frac{1}{4}$. To show that equality holds, it suffices to prove that when the algorithm "trys to color " it always succeeds in coloring. For each vertex w in $U \cup V$, the number of colored edges among incident edges of vertex w is less than or equal to the number of 1's in A_w. Thus when "try to color " case occur for edge (u, v), less than $\lceil \frac{k}{2} \rceil$ edges among the incident edges of vertex u are colored and also less than $\lceil \frac{k}{2} \rceil$ edges among the incident edges of vertex v are colored. Only $2(\lceil \frac{k}{2} \rceil - 1)$ colors are used around vertex u and v in total. So newly added edge (u, v) can be colored.* □

4.5 Lazy Algorithm II

Another lazy algorithm is as follow.

Algorithm A_2: With probability $\frac{1}{2}$, decide whether or not to try to color the added (new) edge (in other words, with probability $\frac{1}{2}$, decide to leave the edge uncolored). If the decision is "to try to color" and if coloring is possible, then color the edge with the least color index possible.

□

Note that the one random bit is needed at the time of each edge addition.

Theorem 4.6 $\frac{1}{4} < C_{A_2} \leq \frac{1}{2}$.

Proof $C_{A_2} \leq \frac{1}{2}$, *as probability of attempting to coloring does not exceed $\frac{1}{2}$.*

To prove that $\frac{1}{4} < C_{A_2}$, we will show that for each edge addition the probability that the added edge is colored is greater than $\frac{1}{4}$. Let b_j be any edge addition operation in the input and let the edge added be (u, v). If the added edge (u, v) can be colored if I intend to, we call the situation "positive". One the other hand, if the edge cannot be colored even if we intend to, we call the situation "negative".

We caluculate the probability of occurence of "negative" cases.

Let E_{uv} denote the set of edges incident to vertex u or v at that time.

Then, the probability of "negative" is not greater than the probability of k or more edges in E_{uv} being "tried to color", that is

$\sum_{i=k}^{t} \left(\begin{array}{c} t \\ i \end{array} \right) (\frac{1}{2})^i \cdot (\frac{1}{2})^{t-i}$,

where $t \stackrel{\triangle}{=} |E_{uv}|$.

For any $t \leq 2(k-1)$, the value is less than $\frac{1}{2}$. Note that for $t \leq k$, the probability of "negative" is 0. Thus situation "positive" occurs with probability greater than $\frac{1}{2}$. So the probability of "try to color" and at the same time "positive" is greater than $\frac{1}{4}$.

We have shown that each added edge can be colored with probability more than $\frac{1}{4}$. This proves that $C_{A_2} > \frac{1}{4}$. □

4.6 Conclusion

We have presented two lazy randomized edge coloring algorithms both with non-zero competitiveness coefficients. Unfortunately, these algorithms are poor against adaptive adversaries (their adaptive competitiveness coefficients are 0).

For practical assignment problems, different criteria for optimality from the one described in this manuscript may be possible. Devising algorithms for different criteria on edge coloring problems and for vertex coloring problems deserve future study.

5 Application of Transitive Graph for Education Support System

5.1 Introduction

There are actually a require subject and an optional subject, etc., and it is the very severe one to study all the subjects though the students have to study a lot of subjects from entrance to graduation in an academic education. And for the educated side, it is true to want the students to master a lot of subjects if possible. Especially there are many subjects related to among each subject. If the students do not master them to some degree, obtaining deep expertise is difficult and they cannot graduate from the special subject in the professional education. Thus, the system that is the master of the credit of each subject and going on to school is a nonreversible system if student dose not get neither repeating a year nor mastering units. Here, we propose a directed graph by showing each subject about the credit earning system from entrance to graduation by vertices, and showing the relations among each subject by the arrow. In addition, we show as the directed st-graph where entrance is vertex s and graduation is vertex t, and we propose the method for the decision of the required subject for the credits toward graduation acquisition.

On the other hand, the generation problem concerning the graph is to find all subgraphs with a special character with the given graph, and the some kinds are [1] $\sim$ [7] that has been considered about various classes in the graph. Taki, Masuda and Kashiwahara [24] thought about equivalence by a structural set with graph G and another structural set in other graph H same as the family of the vertex set.

This time, it was theoretically shown here to be able to apply transitive character in the graph to an educational system. Explanation of Sect. 5.2 defines several terms and notations. Sect. 5.3 describes the relation between transitive graph and the directed st-graph. In Sect. 5.4, it is described that the method for the decision of the required subject for the credits toward graduation acquisition is minimal st-separator in the directed st-graph, and describes the conclusion in Sect. 5.5.

5.2 Preliminaries

Here, to show the relation for the subjects from entrance to graduation by vertices and edges, the term and the definition concerning the graph are described. As described in the foregoing paragraph, entrance and graduation are shown a vertex s and a vertex$t$, respectively. And when the arbitrary

vertices show the subjects from entrance to graduation, the relations among the subjects are shown by edges. Then the acquisition subject relations from entrance to graduation can be graphed.

Here, a directed graph is called a digraph that is distinct from undirected graph, and just a graph is as both types. An undirected edge between vertices u and v is denoted by (u,v). A directed edge from u to v is denoted by $(u \rightarrow v)$. For a graph G, $V(G)$ and $E(G)$ denote the set of vertices and edges, respectively, of G. Therefore, the graph is denoted by $G=(V(G), E(G))$.

In a directed graph, a directed walk is a sequence of vertices and edges $(v_1, e_1, v_2, e_2, \ldots, e_k, v_{k+1})$ such that e_i is an edge from v_i to v_{i+1} for i=1,2, ,k. A directed walk is called a directed path if no vertex appears more than once. A directed walk is called a directed cycle if no vertex appears more than once except that the first and last vertices are the same.

In a directed graph G, if the number of edges coming into vertex v is 0, then v is called a source of G. On the other hand, if the number of edges going out of v is 0, v is called a sink of G.

Definition 5.1 *A directed st-graph is a digraph with distinguished vertice s and t.*

Definition 5.2 *An acyclic graph is a digraph that has no directed cycle.*

Definition 5.3 *An st-path is a simple directed path starting from s and ending at t in a directed st-graph.*

Definition 5.4 *A digraph is said to be transitive if the existence of two edges $(u \rightarrow v)$ and $(v \rightarrow w)$ implies the existence of edge $(u \rightarrow w)$.*

Definition 5.5 *An undirected graph is comparability graph if the edges can be oriented so that the resultant digraph is transitive. (Figure 7)*

Definition 5.6 *A set of vertices in a graph is called an independent vertex set if no two vertices in it are mutually adjacent.*

Definition 5.7 *An independent vertex set is called maximal if it is contained in no other independent vertex set. For undirected graph $G=(V(G),E(G))$, a maximal independent vertex set is abbreviated to MIS(G).*

$MIS(G) \equiv \{S \mid S$ is a maximal independent vertex set of $G\}$

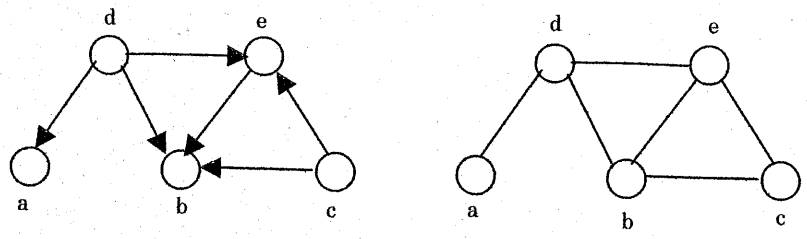


Figure 7: Transitive graph and comparability graph.

Definition 5.8 *A set of vertices in a directed st-graph is called a path independent vertex set if no two vertices in it are on the same st-path, and the relation of such vertices is called the path independence.*

Definition 5.9 *A path independent vertex set which is not contained in any other path independent vertex set is called a maximal path independent vertex set (MPI, for short). For a directed st-graph $H=(V(H),E(H))$, a maximal path independent vertex set is abbreviated to MPI(H).*

$MPI(H) \equiv \{S \mid S$ *is a maximal path independent vertex set of* $H\}$

Definition 5.10 *For directed edge $(u \rightarrow w)$, it is called short cut edge if there exist two directed edges $(u \rightarrow v)$ and $(v \rightarrow w)$.*

Definition 5.11 *An st-path which dose not do confluence and divergence among paths at all is called an independent st-path (Ist-path, for short). That is, Ist-paths do not share any vertices and edges.*

Definition 5.12 *An st-separator of a directed st-graph whose removal disconnects all directed paths from s to t in the st-graph is a set of vertices.*

Definition 5.13 *An st-separator is called a minimal st-separator if it has no proper set of st-separator in the st-separator.*

Definition 5.14 *An st-separator is called a minimum st-separator if it has minimum number of elements among of them.*

5.3 Relation between a directed st-graph and a comparability graph

Here, we discuss what kind of graph can be used as st-directed graph to explain the relations of the acquisition subjects about all processes from entrance to graduation, though it is clear to be able to graph the acquisition

subjects relation from entrance to graduation by defining the foregoing paragraph. For students, the school year progresses by the acquisition of each subject from entrance to graduation as previously described if they do not get neither repeating a year nor mastering units. And they become a special subject end. This is a nonreversible system. Meanwhile, subject A is required for subject B master among them. When there is a relation that subject B is required for subject C, it is difficult to master C if no subject A do master. That is, if $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$. This is transitive relation. Here, we pay attention to this transitive relation, and the structure of graph related to the subject is discussed.

If a graph G is a comparability graph, the edges can be oriented so that the resultant digraph is transitive. Furthermore, such an orientation is called a transitive orientation. Let G' be the graph obtained from G by a transitive orientation. Eliminate all short cut edges of G'. Let V_{source} and V_{sink} be the set of sources and sinks, respectively, of the resultant digraph. Create a new source s and add edge $(s \rightarrow x)$ for each x_{source}. Similarly, create a new sink t and add edge $(y \rightarrow t)$ for each y_{sink}. Let G'' be a st-graph obtained from G' by above procedures. (Figure 8)

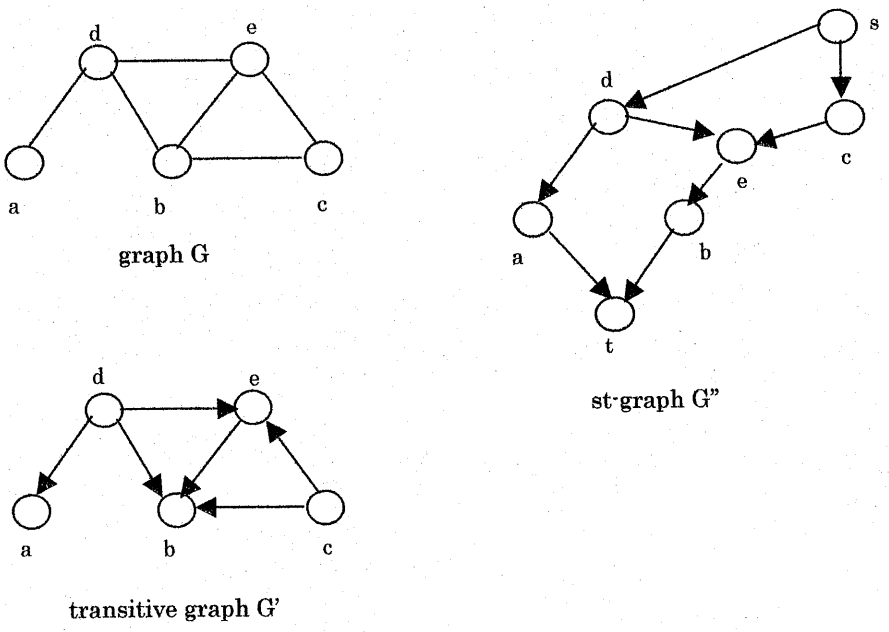


Figure 8: Graph G, transitive graph G', and st-graph G" from G'.

Corollary 5.1 $MPI(G'') \supseteq MIS(G)$.

Proof *For independent vertex a and vertex b in a graph G, it is not reachable from vertex a to vertex b and from vertex b to vertex a in the graph G'. Therefore, it is not reachable from vertex a to vertex b and from vertex b to vertex a in the graph G''. This is mean that, any two independent vertices in the graph G do not exist on same path from vertex s to vertex t in the graph G''. Thus an arbitrary element of* $MIS(G)$ *is an element of a path independent vertex set of* G''*. In addition, let* I_1 *be an arbitrary element of* $MIS(G)$*. Let c be an arbitrary vertex which does not belong to* I_1 *in the graph G. Then, there exist an not independent at least one vertex in* I_1 *of vertex c. Let d be such a vertex. In the graph G', there exist a directed edge from vertex c to vertex d, or from vertex d to vertex c. Therefore, it is reachable from vertex c to vertex d or from vertex d to vertex c in the graph G''. Here, the graph G'' is made from the graph G' by deleting short cut edges of the graph G' and adding vertex s and vertex t to the graph G'. This is mean that, a reachable vertex from an arbitrary vertex in the graph G' is the same as a reachable vertex except vertex s and vertex t in the graph G''.*

Therefore, there exist an st-path passing both vertex c and vertex d in the graph G''. Because this consists for arbitrary I_1 *and arbitrary vertex c,* $MPI(G'') \supseteq MIS(G)$. □

Corollary 5.2 $MPI(G'') \subseteq MIS(G)$.

Proof *For any two vertices a and vertex b not existing on the same st-path in a graph G'', it is not reachable from vertex a to vertex b and from vertex b to vertex a in the graph G''. Therefore, it is not reachable from vertex a to vertex b and from vertex b to vertex a in the graph G'. This is mean that, these two vertices a and b are independent in the graph G. Thus an arbitrary element of* $MPI(G'')$ *is an element of a independent vertex set of the graph G.*

In addition, let MPI1 be an arbitrary element of $MPI(G'')$*. Let ebe an arbitrary vertex which does not belong to MPI1 in the graph G''. Then, there exist at least one vertex on the same st-path in MPI1. Let f be such an arbitrary vertex. Since it is reachable from vertex e to vertex f or from vertex f to vertex e in the graph G'', there exist a directed edge from vertex e to vertex f, or from vertex f to vertex e in the graph G'.*

Therefore, there exist a directed edge between vertex e and vertex f. These two vertex e and vertex f in the graph G are not independent.
Because this consists for an arbitrary vertex e and an arbitrary vertex f, $MPI(G'') \subseteq MIS(G)$. □

From Corollary 5.1 and Corollary 5.2, we obtain the following lemma 5.1.

Lemma 5.1 : *If a graph G is a comparability graph, there exist a directed acyclic st-graph with maximal path independent vertex sets that is equal to maximal independent vertex sets of G.*

Because a directed st-graph is an acyclic finite graph, it has finite number of st-path with finite length. For all of st-paths, we give numbers $(s \rightarrow 1 \rightarrow 2 \rightarrow \cdots \rightarrow n \rightarrow t)$ to vertices in the order of the sequence from vertex s to vertex t on each st-paths. Because all of the vertices in graph H except vertex s and vertex t correspond to graph G, the vertices in graph G are numbered $1, 2, \ldots, n$ in the same way. Each elements of $MPI(H)$ dose not contain any two arbitrary vertices of the vertices, which are numbered $1, 2, \ldots, n$ in the graph H, at the same time.

Corollary 5.3 *If $MPI(H)$=$MIS(G)$, a graph G can be oriented.*

Proof *From $MPI(H)$=$MIS(G)$, there exist undirected edges between any two arbitrary vertices which are numbered $1, 2, \ldots, n$ in the graph G. When these edges are oriented toward from small number to large number which put on both ends point of each edges, it can be oriented to be transitive (Relation $<$ fills the transition for all positive integers). For two vertices a and b which two different st-path or more in graph H passes, it is assumed that two different st-path p_1 and p_2 which satisfies the following conditions exist.*

For numbering vertices on p_1, (the number of vertex a) $<$ (the number of vertex b), and

for numbering vertices on p_2, (the number of vertex a) $>$ (the number of vertex b).

At this time, the graph H has a path from vertex a to vertex b and the graph H also has a path from vertex b to vertex a. This is a contradiction to the graph H being acyclic, because the graph H has a directed cycle. Therefore, all of edges in the graph G can be oriented to be transitive. □

From Corollary 5.3, we obtain the following lemma.

Lemma 5.2 *If there exist a directed acyclic st-graph with maximal path independent vertex sets which is equal to maximal independent vertex sets of a graph G, the graph G is a comparability graph.*

From Lemma 5.1 and Lemma 5.2, we obtain the following theorem.

Theorem 5.1 : *There exist a directed acyclic st-graph with maximal path independent vertex sets which is equal to maximal independent vertex sets of a graph G iff the graph G is a comparability graph.*

5.3.1 St-separator in st-graph

The structure of the graph was described in the forgoing paragraph. There are some subjects of necessity at least to complete the course of study in special subject though there are various subjects in the curriculum. These are offered as requiring optional subjects according to the subject or the department. Here, various subjects (vertex $v_i(i$=1,2,...,$k)$) from entrance (vertex s) to graduation (vertex t) are caught as directed st-graph. At this time, among the vertex sets that separate s and t, the minimal set is requiring optional subjects. That is, the method of finding the requiring optional subject is offered by catching minimal st-separator as requiring optional subjects in the directed st-graph. Moreover, student only has to find minimum st-separator from among the minimal st-separator for the master of the required subject to graduate in addition.

Here, we define the st-path degree $\rho(v_i)$ which is the number of st-path that passes vertex v_i (i=1,2,...,k) in directed st-graph. It is the same as cutting $\rho(v_i)$ st-pathes from the directed st-graph to remove vertex v_i in the directed st-graph. Therefore, the separation of s and t becomes possible. It is the same as the exclusion of st-path from vertex s to vertex t to remove the vertex v. So, if this repeats a similar operation about st-path of the remainder, all st-path from s to t will be cut because it is limited for the number of st-path. The set of these excluded vertices becomes st-separator. It is trivial that the set of adjacent vertices of vertex s (or vertex t) is also st-separator.

St-separator generation process:

Now, there are p st-pathes in the directed st-graph, and at this time, the maximum value of the st-path degree is $Max\rho(v_{i1}) = \alpha_1$. (However, the vertex with α_1 is not necessarily one. Make any one of these vertices with some α_1 vertex v_{i1}, and to same) If this vertex v_{i1} is excluded, α_1 st-pathes are excluded. When the maximum value of the st-path degree is $Max\rho(v_{i2})= \alpha_2$ for (p-α_1) st-pathes of the remainder, α_2 st-pathes are excluded if this vertex v_{i2} is excluded. Hereafter, when the maximum value of the st-path degree is $Max\rho(v_{ij})=\alpha_j$ ($j = 1, 2, \ldots, n$) for st-pathes of the remainder and vertex v_{ij} ($j = 1, 2, \ldots, n$) is excluded, all st-path from s to t will be excluded if the similar operation is repeated. This set of vertices, which are excluded, becomes st-separator. □

By the way, for α_j ($j = 1, 2, \ldots, n$), it is the maximum value in each operation of the above-mentioned generation process at that time, so the

maximum number st-pathes are cut by removing v_{ij}. Therefore, removing one vertex of the maximum st-path degree in each operation was to remove maximum number st-pathes at that time, continuing this operation is following that the maximum value α_j st-pathes were excluded by removing the minimum number of vertices. Obtained st-separator is unique though removed st-path is different because of how to choose the vertex with maximum st-path degree in each operation. That is, these vertex sets are minimal st-separator. The following Corollary is obtained.

Corollary 5.4 *The vertex with the maximum value of the st-path degree is an element of minimal st-separator.*

By the way, there might be two or more vertices with the maximum value of the st-path degree of the same value in the directed st-graph. These are assumed to be v_i and v_j now. And they are v_i $\in$st-pathP_l, v_i $\in$st-pathP_m, v_i $\in$st-pathP_n, v_j $\in$st-pathP_k, v_j $\in$st-pathP_l and v_j $\in$st-pathP_m. That is, when two vertices v_i and v_j are in st-pathP_l and st-pathP_m, and v_j is in st-pathP_k, the v_j is the vertex of st-pathP_k, st-pathP_l and st-pathP_m. Moreover, when vertex v_j is removed, st-pathP_l, st-pathP_k, and st-pathP_m are separated to s and t by vertex v_j though st-pathP_k is not separated to s and t by vertex v_i even if this vertex v_i is removed though v_i is the top of st-pathP_l and st-pathP_m(Figure 9, 10, and 11). Therefore, the following Corollary is obtained.

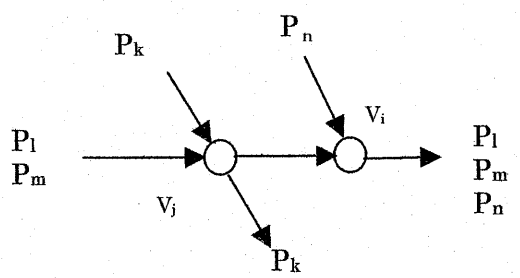


Figure 9: Position of vertex v_i and vertex $v_j$in st-path.

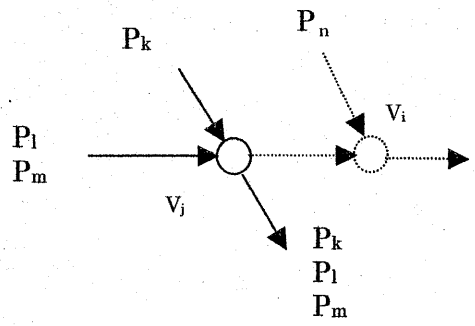


Figure 10: Position of st-path after removing vertex v_i.

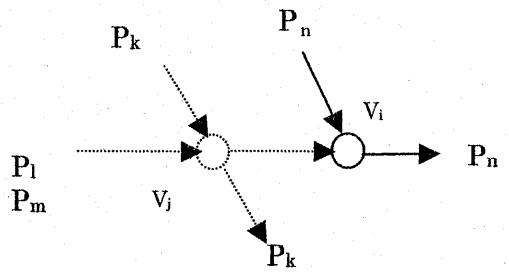


Figure 11: Position of st-path after removing vertex v_j.

Corollary 5.5 *When the vertex with the maximum value of the same st-path degree exists, the vertex on st-path with such only one the vertex is an element of minimal st-separator.*

From each element vertex v_j with the maximum value of the st-path degree in the above-mentioned process, $\rho(v_i)$ st-pathes are cut by excluding v_j for element v_j of minimal st-separator. That is, all st-path is separated to s and t by a minimum number of elements.

Lemma 5.3 *The element of minimum st-separator is the vertex with the maximum value of the st-path degree in each operation of the st-separator generation process.*

Proof *Now, if st-separator obtained by the above-mentioned process is not minimal st-separator, there exist st-separator as a proper subset in this st-separator. That is, there will be the vertex as st-separator besides this proper subset. By the way, because the vertex, which had been obtained for each operation by the generation process, was the vertex that the number of st-path in the directed st-graph was limited with the maximum value of the st-path degree, all st-path from s to t was cut without remaining. This is contradiction that there is the vertex as st-separator besides this proper subset.* □

The following theorem is obtained from Corollary 5.4, Corollary 5.5 and lemma 5.3.

Theorem 5.2 *(Maximum st-path degree - Minimal st-separator.) The vertex with the maximum value of the st-path degree is an element of minimal st-separator.*

An actual algorithm is described based on above.

Here, st-path matrix **S**= $[s_{ij}]$ is defined. Then s_{ij}=1 if there is the vertex v_j in the st-pathP_i, s_{ij}=0 if there is not the vertex v_j in the st-pathP_i. The sum of the each row of the st-path matrix shows the number of st-path that passes the vertex of the row.

Algorithm

‘ st-path matrix **S**.

‘ v_m is maximum sum of row of **S**.

‘ matrix **Sm** is except st-path including the maximum sum of row of the vertex.

Start
1: **S**

$v_m \leftarrow \underset{j}{Max}(\sum_{i=1}^{n} v_{ij})$

Store st-spt() $\leftarrow v_m$

S $\leftarrow$ **Sm**

If **S=0** then end

Go 1

End

5.4 Conclusion

Therefore, the following theorem was presented for the graph when the structure of the graph related to the subject was discussed with the start, and various subject subjects from entrance to graduation were caught as st-directed graph, and the validity was proven.

(1) we have shown that there exist a directed acyclic st-graph with maximal path independent vertex sets which is equal to maximal independent vertex sets of a graph G iff the graph G is a comparability graph.

(2) we have also presented Maximum st-path degree - Minimal st-separator theorem, and proposed an actual algorithm based on it.

Each subject is made the vertex to use this for the educational support system, and the relations between subjects are connected by edges. The relations among these subjects show a transitive relation of applying the short cut edges. Then, if minimal st-separator is obtained by using the directed st-graph, the elements becomes a required subjects in educational study. A flexible to some degree a subject education is natural from the viewpoint of education though there are minimum st-separator, and minimum st-separator it, too. It is a reason for this not to obtain minimum st-separator but to obtain minimal st-separator.

6 Conclusion

In chapter 1, we provided a necessary and sufficient condition for a digraph to be an MIS-diagram for an undirected graph, and then showed that we also presented a simple method for obtaining MWIS-diagrams from MIS-diagrams for positively weighted cases. Based on these results, we can generate efficiently all MIS's or MWIS's of cocomparability graphs. As mentioned before, every interval graph is a cocomparability graph. Thus, our results are extensions of the pertinent parts of Refs. [4] and [6].

In chapter 2, we have shown that the DP algorithm in [14] can be extended to be an output-sensitive generation algorithm for MWIS's of a circle graph. We have extended the result further to obtain output-sensitive generation algorithm for MWIS's of a spider graph and of a circular-arc overlap graph. Chordal graphs admit similar output-sensitive generation algorithm for MWIS's, which will be described in separate paper.
No output-sensitive generation algorithm have been obtained so far for maximal independent sets of a circle graph and of a chordal graph. These problems deserve further study.

In chapter 3, we have described that the competitiveness coefficient of any online edge-coloring algorithm is greater then $\frac{2k-1}{k}$, where k is maximum degree, for deterministic or randomized online edge-coloring algorithm. we have presented new algorithm for online edge-coloring algorithm in next chapter.

In chapter 4, we have presented two lazy randomized edge coloring algorithms both with non-zero competitiveness coefficients. Unfortunately, these algorithms are poor against adaptive adversaries (their adaptive competitiveness coefficients are 0).

For practical assignment problems, different criteria for optimality from the one described in this manuscript may be possible. Devising algorithms for different criteria on edge coloring problems and for vertex coloring problems deserve future study.

In chapter 5, we provided two theorems, and after that we have presented an algorithm for education support system that students have to take required credits from entrance to graduation in a college. It is discussed an st-digraph that entrance and graduation are vertex s and vertex t respectively and subjects from the entrance to the graduation are vertices. Then, if there are minimal st-separators of the st-digraph, those are the required credits from entrance to graduation. We propose a method to decide the required credits for graduation.

References

[1] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa, "A new algorithm for generating all the maximal independent sets," *SIAM J. Comput.*, vol.6, pp.505-517, 1977.

[2] S. Tsukiyama, H. Ariyoshi, I. Shirakawa, and H. Ozaki, "An algorithm to enumerate all cutsets of a graph in linear time per cutset," *J. Assoc. Comput. Mach.*, vol.27, pp.619-632, 1980.

[3] D. Rotem and J. Urrutia, "Finding maximum cliques in circle graphs," *Networks*, vol.11, pp.269-278, 1981.

[4] J. Y. -T. Leung, "Fast algorithm for generating all maximal independent sets of interval, circular-arc and chordal graphs," *J. Algorithms* , vol.5, pp.22-35, 1984.

[5] S. Masuda, K. Nakajima, T. Kashiwabara and T. Fujisawa, "Efficient algorithms for finding maximum cliques of an overlap graph," *Networks*, vol.20, pp.157-171, 1990.

[6] Y. D. Liang, *S.* K. Dhall and *S.* Lakshmivarahan, "On the problem of finding all maximum weight independent sets in interval and circular-arc graphs," *1991 Symposium on Applied Computing*, pp.465-470, IEEE Comput. Soc. Press, 1991.

[7] T. Kashiwabara, S. Masuda, K. Nakajima and T. Fujisawa, "Generation of maximum independent sets of a bipartite graph and maximum cliques of a circular-arc graph," *J. Algorithms* , vol.13, pp.161-174, 1992.

[8] C. Berge, *Graphs and Hypergraphs*, North-Holland, Amsterdam, 1973.

[9] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, San Diego, CA, 1980.

[10] J. Spinrad, "On comparability and permutation graphs," *SIAM J. Comput.*, vol.14, pp.658-670, 1985.

[11] A. V. Aho, M. R. Garey and J. D. Ullman, "The transitive reduction of a directed graph," *SIAM J. Comput.*, vol.1, pp.131-137, 1972.

[12] A. Apostolico, M. J. Atallah and S. E. Hambrusch, "New clique and independent set algorithms for circle graph", *Discrete Applied Math.*, 36, 1-24, 1992.

[13] S. Even and A. Itai, "Queues, stacks and graphs", in *Theory of Machines and Computations*, A. Kohavi and A. Paz (eds.), Academic Press, NewYork, 71-86, 1971.

[14] F. Gavril, "Algorithms for a Maximum Clique and a Maximum Independent Set of a Circle graph", *Networks*, 3, 261-273, 1973.

[15] J. Y. Hsiao and C. Y. Tang, "An efficient algorithm for finding a maximum weight 2-independent set on interval graphs", *Inform. Process. Lett.*, 43, 229-235, 1992.

[16] D. S. Johnson, "The NP-completeness column: an ongoing guide",*J. Algorithms*, 6, 434-451, 1985.

[17] M. Koebe, "Colouring of spider graphs", in R. Bodendiek and R. Henn (eds.) *Topics in Combinatorics and Graph Theory,* Heidelberg, 435-441, 1990.

[18] J. M. Keil, "The complexity of domination problems in circle graphs", *Discrete Applied Math.*, 42, 51-63, 1993.

[19] Robin J. Wilson and John J. Watkins, " Graphs : An Introductory Approach ," John Wiley&Sons Inc, 1990.

[20] A. Bar-Noy, R. Motwani and J. Naor, "The greedy algorithm is optimal for online edge coloring," *Inf. Process. Lett.*, vol.4, pp.251-253, 1992.

[21] U. Faigle and W.M. Nawijn, "Note on scheduling intervals on-line," *Discrete Appl. Math.*, vol.58, pp.13-17, 1995.

[22] S. Fiorini and R.J. Wilson, "Edge-colourings of graphs," Pitman, London 1977.

[23] R. Motwani and P. Raghavan, "RandomizedAlgorithms," Cambridge University Press 1995.

[24] M. Taki, S. Masuda, T. Kashiwahara, "A Representation Diagram for Maximal Independent Sets of a Graph," IEICE Transactions, Vol. E81-A , No.5, pp.784-788, MAY 1998.

[25] V. Bouchitte, and I. Todince, "Listing all Potential Maximal Cliques of a Graph," Lecture Note in Computer Science, pp503_515, Vol.1770, 2000.

ALGORITHMS FOR INDEPENDENT SET IN SOME KINDS OF GRAPHS AND ITS APPLICATIONS

（ある種のグラフにおける独立集合を求める為のアルゴリズムとその応用）

多　喜　正　城

内容梗概

この学位論文において、ある種のグラフにおける、独立頂点集合に関するアルゴリズムと、それを用いた、教育システムへの応用について述べている。

この学位論文は、5章からなり、第1章では、極大独立頂点集合に関する生成問題を取り上げ、第2章では、最大重みを持つ独立頂点集合に関するアルゴリズムについて、いくつかのインターセクショングラフに関して述べ、第3章及び、第4章では、バイパータイトグラフの辺をオンラインで彩色していくアルゴリズムについて述べている。そして、第5章では、独立頂点集合に関するの応用として、教育システムに関する1解答を提供している。各章の関連について、それぞれ、独立集合という共通の課題から議論している。各章の概略をここで述べる。

第1章、頂点sから頂点tに至る単純有向道st-path上の頂点集合（s、tを除く）と、無向グラフGの極大独立集合（MIS）の族が一致するとき、GのMIS-diagramという。ここでは、無向グラフがMIS-diagramであるための有向グラフに対する必要十分条件と、無向グラフがMIS-diagramを持つための必要十分条件は、GがCo-comparability　グラフであることを示す。さらに、正の重みを持つ場合に、MIS-diagramは、重み付MIS-diagramを得る方法も示している。

第2章、頂点数がn、最大重みを持つ独立頂点集合の頂点数の総和をβとしたとき、circle-graphの最大重みを持つ独立頂点集合を重複なしに、列挙するアルゴリズムが、$O(n^3+\beta)$の時間オーダであることと、それが、他の交グラフにも応用することができることを示している。

第3章、辺の色付け（入力）が予め判っている時の方法を、off-line辺彩色アルゴリズム、それに対し、予め判っていなくて、辺が追加されるたびに彩色していく方法を、online辺彩色アルゴリズムという。online辺彩色アル

ゴリズムの使う色と、off-line 辺彩色アルゴリズムの使う色との比の最大値を、online 辺彩色アルゴリズムの competitiveness 係数という。使われる色の最大数 k としたとき、competitiveness 係数が $\frac{2k-1}{k}$ となることを証明。

第 4 章、 上記の辺彩色をバイパータイトグラフで行い、k を制限したとき、competitiveness 係数が、$\frac{1}{4}$≤competitiveness 係数 ≤ $\frac{1}{2}$ になるアルゴリズムを与えている。

第 5 章、 上記で考察されたことから、MIS が、st-graph であることを用い、頂点 s を入学、頂点 t を卒業として捉え、入学から卒業までに提供されるカリキュラムの中から、極小な選択科目を選ぶことが、st-graph の極小 st-separator を求めることと一致することを述べた。

keywords:
independent set, comparability graph, intersection graph, competitiveness coefficient, education support system

研究業績

［学術論文］

多喜正城，千原國宏，推移的グラフの教育支援システムへの応用，教育システム情報学会誌，2004.7(in Printing)

M.TAKI, M. SUGIURA,T.KASHIWAHARA, On-Line Edge-Coloring Algorithms for Degree-Bounded Bipartite Graphs, IEICE TRANS. FUNDAMENTALS, Vol.E85-A, No.5,2002. 5

M.TAKI, H.HATAKENAKA, T.KASHIWAHARA, Algorithms for Generating Maximum Weight Independent Sets in Circle Graphs, Circular-Arc Overlap Graphs and Spider Graphs,IEICE TRANSACTIONS Vol.E82-A, No.8,1999.08

M.TAKI, S.MASUDA, T.KASHIWAHARA, A Representation Diagram for Maximal Independent Sets of a Graph,IEICE TRANSACTIONS, Vol.E81-A, No.5,1998.05

［著書］

多喜正城,知的情報処理のための情報数学,晃洋書房,2000.05.15

多喜正城，武村泰宏,コンピュータと通信,晃洋書房,1991.04

情報学教育研究会(第５章担当),情報社会と情報基礎,第一法規,1990.04

［国際会議］

Masakuni Taki, Investigation About Basic Mathematics in EEPIS, Conference of Educa－tion of Polytechnic, Bandon, Indonesia,1992.02

Masakuni Taki, Report of Information Engineering in EEPIS, Commitee of Higher Education, ITS, Indo－nesia,1992.03

［国内発表］

多喜正城，天目隆平，柏原敏伸,st-graph の道独立集合について,奈良高専研究紀要第 38 号,2003. 3

M.TAKI, M. SUGIURA,T.KASHIWAHARA,A New Algorithm of ON-Line Edge-Coloring Algorithm,奈良高専研究紀要第 37 号,2002,03

M.TAKI, M. SUGIURA, T.KASHIWAHARA,A Competitiveness Coefficient of On-Line Edge-Coloring Algorithm,奈良高専研究紀要第 36 号,2001.03

多喜正城，増田澄男，柏原敏伸,あるグラフの極大頂点集合を表現するダイアグラム,奈良高専研究紀要第 34 号,1999.03

M.TAKI, T.KASHIWAHARA,An Algorithm for Generating Maximum Weight Independent Sets in a Circle Graph,奈良高専研究紀要第 33 号,1998.03

多喜，法林，柏原，荒木,ﾊﾟﾗﾒﾄﾘｯｸｸﾞﾗﾌにおける最大ｸﾘｰｸ重みの折れ点数,奈良高専研究紀要第 30 号,1995.03
工藤英男，多喜正城，吉川博史,ｿﾌﾄｳｪｱ設計教育の支援ｼｽﾃﾑにおける教材の構成,情報処理学会（平成 6 年度後期）全国大会論文集,1994.09
工藤英男，多喜正城，富島磨由美，的場祐司,ｿﾌﾄｳｪｱ設計教育のための CAI ｼｽﾃﾑ,情報処理教育研究集会論文集,1993.12
工藤英男，多喜正城，吉川博史,ｿﾌﾄｳｪｱ設計教育の支援ｼｽﾃﾑの基本設計, CAI 学会研究報告論文集 Vol.93,No.4,1993.1
工藤英男，多喜正城,ｿﾌﾄｳｪｱ設計教育の支援ｼｽﾃﾑの構想,情報処理学会（平成 5 年度後期）全国大会論文集,1993.1
多喜正城,国際協力における高専教官の派遣方法について,高専教育第 16 号,1993.02
高橋，多喜，井上，鈴木，角田，関川，真館,ｲﾝﾄﾞﾈｼｱ共和国電子工学 ﾎﾟﾘﾃｸﾆｯｸ・ﾌﾟﾛｼﾞｪｸﾄ,高専教育第 16 号,1993.02
多喜，法林，柏原，荒木,ある種のｸﾞﾗﾌにおける最大ｸﾘｰｸ重みの折点数について,奈良高専研究紀要第 28 号,1992.09
多喜正城,ｲﾝﾄﾞﾈｼｱ共和国電子工学ﾎﾟﾘﾃｸﾆｯｸ情報工学関連分野について,JICA Report,1992.03
松本，山口，柏原，増田，多喜,一層印刷基盤のﾚｲｱｳﾄ設計に置ける極大平面化ｱﾙｺﾞﾘｽﾞﾑ,電子情報通信学会，技報 COMP91-18,1991.05
多喜正城,ﾌｨﾙﾀによる待ち時間予測,奈良高専研究紀要第 24 号,1988.03
多喜正城,BIOSIS ﾃﾞｰﾀﾍﾞｰｽ，大阪大学大型計算機ｾﾝﾀｰﾆｭｰｽ, Vol.16,No.2,1986.05
多喜正城,ACOS-1000 と SX-1 とのﾌｧｲﾙ転送について，大阪大学大型計算機ｾﾝﾀｰﾆｭｰｽ, Vol.16,No.1,1986.03
多喜正城,BIOSIS データベースについて,全国共同利用大型計算機ｾﾝﾀｰ研究開発論文集 No.7,1985.11
多喜，磯本,ﾃﾞｰﾀﾍﾞｰｽｱｸｾｽと連動したｺﾝﾋﾟｭｰﾀﾈｯﾄﾜｰｸでのﾃﾞｰﾀ処理の分散とﾃﾞｰﾀ転送,総合学術情報ｼｽﾃﾑ形成の為の（異機種間）広域分散型ﾃﾞｰﾀﾍﾞｰｽの研究，科学研究費報告書,1985.03
多喜正城,ｱﾀﾞﾌﾟﾃｨﾌﾞﾃﾞｨｼﾞﾀﾙﾌｨﾙﾀによる待ち時間予測,全国共同利用大型計算機ｾﾝﾀｰ研究開発論文集 No.6,1984.11
多喜，井上,確率ｼｽﾃﾑ論による待ち時間予測,情報処理学会，第 28 回全国大会講演論文集,1984.03
多喜正城,大阪大学大形計算機センターの需要予測,七大学大形計算機センター丹羽委員会資料,1983.12
多喜，井上,確率ｼｽﾃﾑ論による待ち時間予測ﾓﾃﾞﾙ,全国共同利用大型計算機ｾﾝﾀｰ研究開発論文集 No.5,1983.12

多喜，北本,稼働状況表からの一考察 PART2,全国共同利用大型計算機ｾﾝﾀｰ研究開発論文集 No.4,1982.11

多喜正城,ﾊﾞｯﾁｼﾞｮﾌﾞ処理の回帰分析より,全国共同利用大型計算機ｾﾝﾀｰ研究開発論文集 No.2,1980.11

多喜正城,大阪大学大形計算機ｾﾝﾀｰ稼働状況表からの一考察,全国共同利用大型計算機ｾﾝﾀｰ研究開発論文集 No.1,1979.11

多喜正城,大型計算機ｾﾝﾀｰにおける過去の統計ﾃﾞｰﾀによる一考察,情報処理学会，第 20 回全国大会講演論文集,1979.07

多喜，藤井，伊澤,動的計画法のよるﾊﾞｯﾌｧ設計の一方法,昭和 49 年度電子通信学会全国大会論文集,1974.07

藤森、多喜,3-regular graph における頂点染色数と頂点被覆数の数え上げとそのｱﾙｺﾞﾘｽﾞﾑ,第 5 回電気学会関西支部高専卒業研究発表会,1998.03

福田、長嶋、多喜,ｻｯｶｰｺｰﾁﾝｸﾞｼｽﾃﾑの開発,第 10 回学生研究発表会(教育システム情報学会),1996.03

多喜、工藤、金堀,ﾏﾙﾁﾒﾃﾞｨｱを利用したｽﾎﾟｰﾂｺｰﾁﾝｸﾞｼｽﾃﾑの研究,教育システム情報学会第 20 回全国大会,1995.08

多喜、内田,ﾍﾞｰｽﾎﾞｰﾙｺｰﾁﾝｸﾞｼｽﾃﾑに関する研究,CAI 学会関西支部学生によるｺﾝﾋﾟｭｰﾀ利用研究発表会,1995.03

下村，多喜,情報工学専攻学生におけるｺﾝﾋﾟｭｰﾀ不安とその要員,教育工学関連協会連合第 4 回全国大会,1994.1

多喜正城,大型計算機ｾﾝﾀｰにおける過去の統計ﾃﾞｰﾀによる一考察,情報処理学会，第 20 回全国大会講演論文集,1979.07

多喜，藤井，伊澤,動的計画法のよるﾊﾞｯﾌｧ設計の一方法,昭和 49 年度電子通信学会全国大会論文集,1974.07