The Nepali Math. Sc. Report
Vol. 20 No. 1&2, 2002

Recognition and Isomorphism Algorithms of Shop Graphs

TANKA NATH DHAMALA*

Abstract: Shop graphs are the graphic representations of classical shop scheduling problems. We present an efficient algorithm to decide whether a given directed graph is a shop graph. This extends a result on the recognition algorithms of sequence graph (acyclic orientation of the Hamming graph $K_n \times K_m$). Orientations of the $K_n \times K_m$ play important roles in practical shop problems. Moreover, we give a short review on the concept of sequence (acyclic shop graph) isomorphisms.

Key Words: shop problems, recognition algorithm, isomorphic sequences.

1. Introduction

In an $n \times m$ *shop scheduling problem*, each job i with $i \in I = \{1, \ldots, n\}$ has to be processed on each *machine* j with $j \in J = \{1, \ldots, m\}$ exactly once without preemption for the processing time $p_{ij} > 0$. We assume that, at a time, each machine can process at most one job and each job can be processed on at most one machine. Let $SIJ = I \times J$ be the set of all operations o_{ij} with $i \in I \wedge j \in J$. The matrix of processing times is denoted by $P = [p_{ij}]_{n \times m}$. We denote the completion time of job i on machines by C_i and the matrix of completion times by $C = [c_{ij}]_{n \times m}$ so that $C_i = \max_j (c_{ij})$ holds, where c_{ij} is the completion time of operation o_{ij}. A scheduling problem is denoted by a triple $\alpha \,|\beta|\, \gamma$, where α describes the machine environment, β gives the job characteristics and γ represents the objective function (cf. [11]). The machine order for job i is the order of machines which process job i, whereas the job order on machine j is the order of jobs processed on machine j. We have to find a feasible combination of machine orders and job orders

* The author would like to extend sincere thanks to DAAD, and to Prof. Heidemarie Bĭasel

(a sequence) which minimised the makespan $C_{max} = \min_i \{C_i\}$. A *schedule* give the corresponding time table.

A computational problem is a function $\prod : Z \to Y$, where Z is the set of all problem instances I and Y is the set of solutions both reasonably encoded as strings of symbols in predefined alphabet. A problem $\prod$ is called decision problem if $Y = \{yes, no\}$. Each optimisation problem has its decision counterpart which is associated by defining an additional threshold value y for the corresponding objective function γ. For example, given an additional threshold value y for the objective function γ we ask : does there exist a feasible solution $x \in X$ such that $\gamma(x) \leq y$? The significant meanings of each of the space and time complexities from the computational point of view are systematically analysed in [9]. Informally, a decision problem is said to be in class $\mathcal{P}$ if there is a polynomial time algorithm solving it. A decision problem belongs to the class $N\mathcal{P}$ if a positive answer can be verified in polynomial time. One of the major open problems of modern mathematics is whether $\mathcal{P} = N\mathcal{P}$. A decision problem in $N\mathcal{P}$ is called *$N\mathcal{P}$-complete* if it can be solved polynomially only if $\mathcal{P} = N\mathcal{P}$.

In Section 2, we summarise the block-matrices model and give some basic notions of graphs applicable in the considered shop scheduling problems. In the block-matrices model, all graph theoretical structures in shop problems are basically described by means of special latin rectangles. For a detailed description of these structures we refer to [5] and to **http://fma2.math.uni-magdeburg.de/ ~ lisa.**

In section 3, given a connected digraph we mainly deal the problem of deciding whether it is a shop graph in a shop scheduling problem. We give an efficient recognition algorithm with linear time and space complexities. One of the objectives to consider such a problem is to investigate some interesting properties of Hamming graph in shop scheduling problems.

The problem of efficiently recognising whether a given graph is a Hamming graph is often treated in the literature (cf. [1,13,14]). A first non-algorithmic proof is : up to isomorphism, all finite connected graphs have unique prime factorisation (cf. [21]). Recall that none of the algorithms which belongs to the class of general decomposition algorithms of graphs with respect to the Cartesian product is linear. Several algorithms for recognising Hamming graphs have already been proposed in the literature. The running time complexity of the first algorithm is bounded by $O(|V|^5)$ (cf. [23]). The fastest known approach for recognising whether a given graph is a Hamming graph, by solving prime factorisation algorithms with respect to the graph Cartesian product, is presented in [1]. Given an undirected connected graph $G = (V,E)$ in its adjacency list data structure, a (unique) prime factorisation of G is obtained with respect to the Cartesian product in $O(|E| \log (|V|))$ time (cf. [1]). An algorithm with linear time complexity with respect to the edges of the graph is

presented in [14] and [13]. Note that the space complexity is also linear in [13] with adjacency list representations. In this way, the considered Hamming graph recognition problem is solved in linear time. We refer to a couple of algorithms in the cited articles for detailed descriptions and the references therein for many striking characterisations of Hamming graphs and several other classes of graphs closely related to Hamming graphs.

In section 4, we consider the decision version of a sequence graph isomorphism problem arising from shop scheduling problems. The decision version of the graph isomorphism problem is to decide whether given two graphs G_1 and G_2 are isomorphic. The Graph automorphism problem is : given a graph G, decide whether its automorphism group contains non-trivial automorphism.

Much effort has been made to find efficient algorithms for the graph isomorphism problem but no polynomial algorithm for this problem has been developed and it is unknown if such an algorithm can exist. The graph isomorphism problem belongs to the class NP but it is still unknown whether it belongs to the class $\mathcal{P}$ or $\mathcal{NP}$-complete. However, for certain special subclasses of graphs, the isomorphism problem is efficiently solvable. For example, planar graphs solved by J. E. HOPCROFT and J.W. WONG in 1974, undirected graphs generated from latin squares solved by G.L. MILLER in 1978, graphs of bounded valence (cf. [17]), cyclic tournaments (cf. [19]) and graphs of bounded average genus (cf. [6]). Because all undirected edges can be replaced by two anti-parallel arcs, the isomorphism problem for undirected graphs is polynomial time reducible to a corresponding isomorphism problem for the directed graphs. On the other hand, the directed graph isomorphism problem is polynomial time reducible to the problem of undirected graph isomorphism (cf. [18]); the problem of directed graph isomorphism and undirected graph isomorphism are polynomially equivalent.

Some concluding remarks are contained in the final section of the paper

2. Basic Concepts

Given a combination of machine orders and job orders in a shop problem of n jobs and m machines, we define the following pair of acyclic digraphs with vertex set SIJ:

- *Machine order graph* $G_{MO} = (SIJ, E_{MO})$, where the set of arcs contains the precedence constraints of all machine orders.
- *Job order graph* $G_{JO} = (SIJ, E_{JO})$, where the set of arcs contains the precedence constraints of all job orders.

The graphs G_{MO} and G_{JO} consist of n and m acyclic components, respectively. We represent by $MO = [mo_{ij}]$ and $JO = [jo_{ij}]$ the $n \times m$ rank matrices of given graphs G_{MO} and G_{JO}, called machine order matrix and job order matrix, respectively. The rank of a vertex in an acyclic directed graph is the number of vertices on a longest

path from a source to the vertex itself. Moreover, jo_{ij} is the position of job i in the job order on machine j and mo_{ij} is the position of machine j in the machine order for job i. The collections of all job order matrices and machine order matrices are, respectively, denoted by JO and MO.

- For given MO and JO, and the set SIJ of operations we define a digraph $G_{MO,JO} = (SIJ, E_{MO,JO})$, where the arc set $E_{MO,JO} = E_{MO} \cup E_{JO}$ reflects all machine orders and all orders.

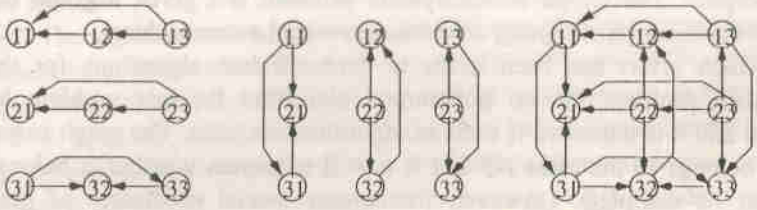

Figure 1: The Transitive Graphs G_{MO}, G_{JO} and $G_{MO,JO}$

Note that , the graph $G_{MO,JO}$ is connected and it may or may not be acyclic. We use [G] for the underlying undirected graph of a digraph G.

Definition 1. *For any pair* (MO, JO), *the graph* $G_{MO,JO}$ *is called a transitive shop graph. If the graph is acyclic* (*cyclic*) *we call it a transitive sequence graph* (*non-sequence graph*).

If we consider only direct precedence constraints in the foregoing graphs, the attribute "transitive" is dropped. The pair (MO, JO) is a sequence (non-sequence) if the shop graph is acyclic (cyclic), respectively. Here, transitive shop graphs and shop graphs are denoted by the same notations for the easiness. The graphs in Figure 1 illustrate the transitive graphs of machine orders, job orders and non-sequence graph with $m = n = 3$.

For each sequence graph $G_{MO,JO}$ we can describe the sequence (MO, JO) by a rank matrix, too. The corresponding matrix $A = [a_{ij}]$ contains the rank of the vertex o_{ij} for each operation o_{ij} in the sequence graph $G_{MO,JO}$. Note that the rank matrix A is a special latin rectangle with *sequence property* : for each integer $lr_{ij} > 1$ there exists the integer $lr_{ij} - 1$ in row i or in column j or in both. Recall that a latin rectangle $LR[n, m, q] = [lr_{ij}]$ is a matrix of size $n \times m$ with its entries $l_{ij} \in \{1, 2, \ldots, q\}$ such that each integer of the symbol set occurs at most once in each row and at most once in each column of LR (cf. [7]). If $n = m = q$ holds, then the matrix is a latin square of order n and is denoted by $LS[n]$.

On the other hand, given any latin rectangle $LR\ [n, m, q] = [l_{ij}]$, we can define a sequence graph by means of its entries l_{ij} as a level of the vertex o_{ij}. Therefore, in particular, any latin rectangle satisfying the sequence property obviously produces an acyclic digraph for a shop scheduling problem with n jobs and m machines. Therefore, there exists a one-to-one correspondence between the set of all latin rectangles with sequence property and the set of all sequence graphs for the open shop scheduling problem (cf.[5]).

The sets of all sequences and sequence graphs of format $n \times m$ in machine environment α are denoted by $\mathcal{LR}(\alpha\ ;\ n, m)$ and by $\mathcal{G}_{\mathcal{LR}}(\alpha\ ;\ n, m)$, respectively. If the context is clear from the considerations, then we may denote these sets simply by $\mathcal{LR}$ and $\mathcal{G}_{\mathcal{LR}}$, respectively. If $m = n = q$, then the set of all latin squares $\mathcal{LS}[n]$ concentrated in machine environment α is denoted by $\mathcal{LS}(\alpha\ ;\ n)$. Clearly, all latin squares satisfy the sequence property. Each element of $\mathcal{LR}(\alpha\ ;\ n, m)$ is also called sequence. A sequence contains all information about machine orders and job orders of the corresponding sequence graph. The terms like source, sink, operation, path, etc., are interchangeable in the sequence graph and the sequence accordingly. By the one-to-one correspondence, we see that the determination of the cardinality of $\mathcal{G}_{\mathcal{LR}}(\alpha\ ;\ n, m)$, which in general is an unsolved counting problem, can also be described as the problem of determination of cardinality $\mathcal{LR}\ (\alpha\ ;\ n, m)$ in the open shop scheduling problem, but the latter problem is also quite hard. One possibility to handle the former problem is the chromatic polynomial of the Hamming graph $K_n \times K_m$, however, the calculation is hard (see [12]).

Note that an infinite set of schedules can be assigned to each sequence. On the other hand, each schedule contains all information about its unique sequence. More formally, we can define an equivalence relation $\mathcal{R}$ on the set of all schedules as:

$S_1 \mathcal{R} S_2 \Leftrightarrow$ both schedules S_1 and S_2 base on the same sequence LR.

It is clear that each equivalence class contains an infinite number of schedules but the number of classes is finite; a trivial upper bound is $(n!)^m (m!)^n$; we refer to [8, 12] for some improvements. In order to find a set of distinct representatives, we may use the semiactive schedules under unit processing times, i.e., all sequences. A schedule is called semiactive if each operation $o_{ij} \in SIJ$ is started as early as possible with respect to the given machine orders and job orders. However, to every sequence of a shop scheduling problem, we can associate a unique semiactive schedule $C = (A, P)$ in linear time $O(nm)$ (cf. [5]). The same time complexity holds for the calculation of the weights of a longest path through the operation o_{ij} for each operation $o_{ij} \in$ SIJ (cf. [5]). Given a sequence of certain

format, determining the associated semiactive schedule is a polynomially solvable problem. Therefore, the main difficulty on the complexity of shop scheduling problems lies also to the construction of appropriate sequences.

The transitive sequence graph is an acyclic oriented digraph of the disjunctive graph (see [22]), where all vertices $o_{ij} \in SIJ$ have unit processing time weights. Disjunctive graphs are widely used to represent certain schedules for general shop scheduling problems. On the other hand, if we assign in a graph $\mathcal{G}_{\mathcal{LR}}(SIJ, E_{MO\cdot JO})$ a vertex cost p_{ij} to each $o_{ij} \in SIJ$, we can consider different objective functions on the set $\mathcal{G}_{\mathcal{LR}}(\alpha\,; n, m)$ of sequences graphs and hence on the set $\mathcal{LR}(\alpha\,; n, m)$ of sequences, too. In particular, if $p_{ij} = 1$ for all i and j, then we have the equation $C = \mathcal{LR}[n, m, r]$. Moreover, $C_{\max} = \max\{c_{ij} | o_{ij} \in \text{SIJ}\}$ is given by the weight of a critical path (the length of a longest path from a source to a vertex itself) in G_{LR} in the case of operations set $SIJ = I \times J$. Then the problem in this case is to determine a sequence with minimal cardinality of the insertion set, for instance.

The graph $[G_{LR}]$ corresponds to the linegraph of bipartite graph $G = (I \cup J, E)$ with edge $(i, j) \in$ E if and only if job i with $i \in \{1, 2, \ldots, n\}$ is processed machine j with $j \in \{1, 2, \ldots, m\}$. For example, the underlying graph $[G_{LR}]$ for $O2\,|n = 2|\,\gamma$ is isomorphic to the 4–cycle Z_4. The linegraph $L(G) = (V_L, E_L)$ of a graph $G = (V, E)$ is a graph with $V_L = E$ and $\{ab, xy) \in E_L$ if and only if $\{a,b\}$ and $\{x, y\}$ belonging to the edge set E are adjacent in the graph G. A graph $G = (V, E)$ is a bipartite if there exist disjoint subsets U and W of V with $V = U \cup W$ such that $\{u, w\} \in E$ implies either $(u \in U \wedge w \in W)$ or $(w \in U \wedge u \in W)$. The graph $[G_{LR}^{tr}]$ of the transitive closure G_{LR}^{tr} of a sequence graph G_{LR} is known as a comparability graph. The transitive closure of a digraph $G = (V,E)$ is the digraph denoted by $G^{tr} = (V, E^{tr})$ such that for each are $(x, y) \in E^{tr}$ there is a path $w_G = (v_0, v_1, \ldots, v_k)$ in acyclic digraph $G = (V, E)$ with $x = v_0$ and $y = v_k$. An undirected graph $G = (V,E)$ is a comparability graph (cf. [10]) if there exists an acyclic orientation E^* of edge set E such that the corresponding digraph $G^* = (V, E^*)$ is transitive closure.

Two graphs $G_i = (V_i, E_i)$ $(i = 1,2)$ are said to be isomorphic denoted by $G_1 \cong^{\psi} G_2$, if there exists a bijection $\psi\colon V_1 \to V_2$ such that for all $v,w \in V_1$ we have $\{v, w\} \in E_1$ if and only if $\{\psi(v), \psi(w)\} \in E_2$; the bijection ψ is called graph isomorphism. A permutation on the set of vertices of a graph is called an automorphism of a graph $G = (V, E)$ if it is adjacency preserving.

The Cartesian product $G := G_1 \times G_2$ of two simple finite graphs G_1 and G_2 is the graph with vertex set $V := V_1 \times V_2$ and edge $\{ux, vy\} \in E_1 \times E_2 =: E$ whenever $\{u,v\} \in E_1$ and $x = y$, or $\{x, y\} \in E_2$ and $u = v$. It is easy to see that the graph Cartesian product is commutative and associative and has the one-vertex simple graph K_1 as a unit such that $K_1 \times G = G = G \times K_1$ for any graph G. Because of the

associativity property, one may write $G = G_1 \times G_2 \times \ldots \times G_r$ for a product G of graphs and the vertex set of such a product the set of all r-tuples $u_1 u_2 \ldots u_r$ where u_k is the vertex in the k-th component. Moreover, the Cartesian product of two graphs is connected if and only if both factors are connected. A graph G is called prime if $G = G_1 \times G_2$ implies $G_1 = K_1$ or $G_2 = K_1$. A set $\{G_1, G_2, \ldots, G_r\}$ of r-graphs is called a prime factorisation of G if $G = G_1 \times G_2 \times \ldots \times G_r$ and $G_k \neq K_1$ for all $k = 1,2, \ldots r$, where G_i is a prime graph.

A Hamming graph $H(G)$ is the Cartesian product of complete graphs, and the Hamming distance, which is the shortest path distance in graph $H(G)$, between two r-tuples $u = u_1 u_2 \ldots u_r$ and $v = v_1 v_2 \ldots v_r$ is the number of positions in which the entries in u and v differ. More formally, Hamming distance is defined as the following discrete metric function:

$$\text{for all } u, v \in H(G),\ d_H(u,v) := |\{k : u_k \neq v_k\}|,$$

so that the Hamming graph $H(G)$ is a discrete metric space. In terms of Hamming distance one may characterise two adjacent vertices $u = u_1 u_2 \ldots u_r$ and $v = v_1 v_2 \ldots v_r$ in $H(G)$ if and only if $d_H(u,v) = 1$. For example, the Cartesian product of r copies of the complete graph K_2 is a special Hamming graph, called hypercube Q_r(r-cube for short) or binary Hamming graph. This class of graphs is a well known object in the field of graph theory as well as computer science.

The definition of Hamming labelling can be described as follows. For all $k = 1, 2, \ldots, r$, let $t_k \geq 2$ be given integers. Furthermore, we consider the vertices of a r–dimensional Hamming graph as all $\prod_{k=1}^{r} t_k$ such r–tuples $u := u_1 u_2 \ldots u_r$ with $1 \leq u_k \leq t_k$ for all $k = 1, 2, \ldots, r$, where two vertices are adjacent if and only if they differ in exactly one place. Then, this type of labelling of the vertices of Hamming graph is called a Hamming labelling. Note that the product graph $H(G) = K_{t_1} \times K_{t_2} \times \ldots \times K_{t_r}$ is an $(\sum_{k=1}^{r} t_k - r)$ –regular graph on $\prod_{k=1}^{r} t_k$ vertices, and hence it has $\frac{1}{2} \prod_{k=1}^{r} t_k \ (\sum_{k=1}^{r} t_k - r)$ edges. Moreover, the neighbourhood of a vertex v in this graph $H(G)$ induces a disjoint union of complete graphs.

Let $G = (V, E)$ be a connected digraph. A topological sorting of the vertices of V is a mapping $\sigma: V \to \{1, 2, \ldots, |V|\}$ such that $\sigma(v) < \sigma(w)$ for all $(v, w) \in E$. For example, $\sigma(v) = 1$ for all sources $v \in$ V. It is well known that a digraph G is acyclic if and only if there exists a topological sorting of G. The Topological Sorting Algorithm is implemented in $O(|E| + |V|)$ time (cf. [15, 16]) using an appropriate data structure.

3. The Recognition Algorithm

In this section we present an efficient algorithm for the following decision problem:

Given : A connected digraph $G = (V, E)$.

Question : Is G a transitive shop graph ?

Moreover, if the answer of this question is "yes", the algorithm yields the matrices *MO* and *JO* for the (transitive) shop graph $G_{MO,JO}$.

We use two well-known algorithms, namely, the Hamming Graph Algorithm (cf. [13 14]) and the Topological Sorting Algorithm (cf. [15,16]), for solving this problem. Here for our purpose, the prescribed labelling algorithm is applicable to the Cartesian product $K_n \times K_m$ of two complete graphs starting from the label o_{11} for the first chosen vertex (cf. [2,8]).

A clique in a graph is its subgraph which is isomorphic to a complete graph and a tournament is an oriented complete graph. We call E^{tr} a transitive tournament if for all vertices $(x,y) \in E^{tr}$ and $(y, z) \in E^{tr}$ implies $(x,z) \in E^{tr}$. Clearly, a tournament is transitive if and only if it is acyclic. Each 2-dimensional Hamming graph $K_n \times K_m$, which is of order *nm*, contains m disjoint subgraphs isomorphic to K_m (so-called n-cliques), and n disjoint subgraphs isomorphic to K_m (so-called m-cliques). On the other hand, a transitive shop graph contains m disjoint subgraphs isomorphic to transitive tournaments of order *n* (so-called column tournaments) and n disjoint subgraphs isomorphic to transitive tournaments of order *m* (so-called row tournaments).

Now, we are able to formulate the following algorithm and verify its validity.

Algorithm 1. Transitive Shop Graph Recognition

Input : Connected digraph $G = (V, E)$.

Output : *MO* and *JO*, if *G* is a transitive shop graph.

Step 1. *Call the Hamming Graph Algorithm for the graph* $[G]$ *of* $= (V, E)$.
If $[G]$ *is not a Hamming Graph,* ***then goto Step 5;***
else *output of the Hamming graph Algorithm:*
n,m, for all $v \in V$ *: label* $(v) = o_{ij}$ *with* $1 \le i \le n$, $1 \le j \le m$;

Step 2. *Identify each vertex* v *with the associated label* $(v) = o_{ij}$;
the cliques K_i^r *and* K_j^c *are the complete subgraphs of* $G = (V, E)$ *induced by the vertex sets* $\bigcup_{j=1}^{m}\{o_{ij}\}$, $i = 1, 2, \ldots, n$, *and* $\bigcup_{i=1}^{n}\{o_{ij}\}$, $j = 1, 2, \ldots, m$, *respectively* ;

Step 3. *For all* $i \in \{1, 2, \ldots, n\}$ ***do***
begin
Call the Topological Sorting Algorithm for the clique K_i^r *;*
if K_i^r *contains a cycle then goto Step 5;*
else output of this algorithm : $rk(o_{ij})$ *for all* $o_{ij} \in V(K_i^r)$ *;*
insert the values $rk(o_{ij})$ *into row i of matrix MO;*
end;

Step 4. ***For all*** $j \in \{1, 2, \ldots, m\}$ ***do***
begin
Call the Topological Sorting Algorithm for the clique K_j^c *;*
if K_j^c *contains a cycle* ***then goto*** *Step 5;*
else output of this algorithm : $rk(o_{ij})$ for all $o_{ij} \in V(K_j^c)$ *;*
insert the values $rk(o_{ij})$ *into column j of matrix JO;*
end;

Output : *G is a transitive shop graph ; MO, JO ;* ***stop.***

Step 5. *G is not a transitive shop graph ;* ***stop***

Because a tournament is transitive if and only if it is acyclic, it provides a linear time complexity for recognising transitive (equivalently, acyclic) tournaments (cf. [10]). The matter is first to calculate the in-degree (or, out-degree) of each vertex, and then to verify that there are no repetitions among the in-degrees (or, out-degrees). It is well known (cf. [20]) that each tournament contains an oriented Hamiltonian path. Moreover, such an acyclic (equivalently, transitive) tournament contains exactly one Hamiltonian path. Afterwards, each tournament which is a subgraph of a transitive shop graph is spanned by a unique path.

Theorem 1. Let $G = (V, E)$ *be a connected digraph. Then the problem of deciding whether G is a transitive shop graph is solvable in* $O(\max\{mn^2, m^2n\})$ *time.*

Proof: Given a connected undirected graph $G = (V, E)$, the problem of deciding whether $[G]$ is a Hamming graph is solvable by the Labelling Algorithm and the Hamming Graph Algorithm in $O(|E|)$ time complexity (cf. [13, 14]). Here, given the connected digraph $G = (V, E)$ we consider the underlying undirected graph $[G]$. Afterwards, we call the Hamming Graph Algorithm [13, 14] and decide if it is a Hamming graph of this kind.

Total time complexity for the Steps 3 and 4 is approximately the same as that of Topological Sorting Algorithm, namely, $O(|V|+|E|)$ and this bound can be reduced to $O(|E|)$ in this case, because $|V| < |E|$ in our shop model. Because each acyclic orientation of a complete graph contains exactly one Hamiltonian path (cf.

[20]), by calling the Topological Sorting Algorithm [16] for all cliques in the Hamming graph we determine the ranks of vertices which reflect the precedence constraints. Thus, since the Steps 2 to 4 do not need more time than Step 1 in the algorithm, the overall time complexity of the algorithm is not worse than the complexity of the Hamming Graph Algorithm.

Moreover, since the total number of edges in a Hamming graph associated to any classical shop scheduling problems of our interest in $n\binom{m}{2} + m\binom{n}{2}$, the time complexity reaches $O(\max\{mn^2, m^2n\})$. ■

Clearly, the presented recognition algorithm is linear in time with respect to the arcs of the input digraph.

A transitive sequence graph is an acyclic orientation of the Hamming graph $K_n \times K_m$ (cf. [3, 12]). Therefore, if we consider the transitive sequence graph, then the number of arcs of such a digraph G_{MO_JO} is $n\binom{m}{2} + m\binom{n}{2}$, and it is reduced to $n(m-1) + m(n-1)$ in the case of a sequence graph without transitive arcs. Therefore,

Theorem 2 *For a given digraph $G = (V, E)$, the problem of deciding if it is a transitive sequence graph is solvable in $O(\max\{mn^2, m^2n\})$ time. Moreover, the computational complexity of the sequence according to a given $n \times m$ transitive sequence graph is $O(\max\{mn^2, m^2n\})$.*

Proof: The prescribed algorithms in the cited references [3, 12] are the Hamming Labelling Algorithm and the Hamming Graph Algorithm [14], and the Topological Sorting Algorithm [16]. The focus point of the proof of stated statement is to consider the problem into two different algorithmic parts. In the first part, one considers the former two algorithms in order to check and label the vertices if it is a Hamming graph. The final part is to calculate the rank if it is acyclic which can be performed by topological sorting. Precisely, the rank of the vertex labeling o_{ij} in the sequence graph is associated with the value of the element o_{ij} in the sequence. In this sorting algorithm, we have to mark all sources instead of a single source at a time and delete the marked sources and the adjacency arcs. The time complexity is clear because of the number of arcs in this specific sequence graph. Moreover, the space complexities remain $O(\max\{mn^2, m^2n\})$ by using the advantages of the algorithms in [13], where the adjacency lists data structures are implemented. ■

However, note that the complexity $O(mn)$ is correct in the case of given $n \times m$ sequence graphs without any transitive arcs. Furthermore, it is also clear that if the number of jobs is completely dominated by the number of machines ($n \ll m$), then the computational time complexities $O(\max\{mn^2, m^2n\})$ and $O(mn)$ reduce to $O(m^2)$ and $O(m)$, respectively.

4. Isomorphism of Sequences

In this section we review an efficient algorithm (cf. [3, 12, 8]) for the decision problem :

Given : Two sequences of the same format.

Question : Does there exists a sequence isomorphism ?

Moreover, if the answer of this question is "*yes*" the algorithm yields an isomorphism. We denote by π_r, π_c, Φ, and ψ, respectively, a row permutation $\pi_r \in S_n$, a column permutation $\pi_c \in S_m$, a transposition $\Phi \in Z_2$, and $\psi \in Z_2$, of a matrix, where S_t is the usual symmetric group on $\{1,2, \ldots, t\}$ and Z_2 is the cyclic group of order two. The transposition of a matrix means reflecting in the main left-to-right diagonal. Note that Φ maps sequence A to the transposed sequence A^T. For matrix reversion of sequence A, we have to replace each arc $(o_{ij}, o_{kl}) \in E_{MO,JO}$ by an oppositely oriented arc (o_{kl}, o_{ij}) in the sequence graph $G_A = (SIJ, E_{MO,JO})$. In order to classify the main structural differences and to get deeper study of equivalent sequence properties between shop scheduling problems, the following definition is more useful (see [4]).

Definition 2 *Two sequences A and B are called structure isomorphic, graph isomorphic or permutation isomorphic, denoted by $A \cong_s B$, $A \cong_g B$, or $A \cong_p B$, if there exists a mapping such that $(\pi_r, \pi_c, \Phi, \Psi)A = B$, $(\pi_r, \pi_c, \Phi)A = B$ or $(\pi_r, \pi_c)A = B$, respectively.*

For the sake of simplicity, we simply say that two sequences A and B are isomorphic if they are isomorphic under any one of the above isomorphism relations. The notion $A \cong B$ is used to represent such an arbitrary isomorphism. Note that the collection of all isomorphisms of the same type under the same formats forms a group. Therefore, there are three groups, namely, $S_n \times S_m$, $S_n \times S_m \times Z_2$ and $S_n \times S_m \times Z_2 \times Z_2$ according to each isomorphism type mentioned above. The order of the group of permutation isomorphisms is always $n!m!$, whereas, for $m \neq n$, this number is $n!m!$, and $2n!m!$ if we consider the group of graph isomorphisms, and group of structure isomorphisms, respectively. Furthermore, if $m = n$, then the latter two numbers are exactly doubled, because transposition of a sequence is also applicable in this case.

Clearly, each of the relations $A \cong_s B$, *or* $A \cong_g B$ and $A \cong_p B$ defined above yields an equivalence relation on the set $\mathcal{LR}$ decomposing the set of all sequences into disjoint isomorphism classes. We denote by $\mathcal{PI}$, $\mathcal{GI}$ and $\mathcal{SI}$, respectively, the collections of all permutation isomorphism classes, graph isomorphism classes and structure isomorphism classes. Obviously, $|\mathcal{PI}| \geq |\mathcal{GI}| \geq |\mathcal{SI}|$, since the relations $A \cong_p B \Rightarrow A \cong_g B \Rightarrow A \cong_s B$ hold. The

sequences of an isomorphism class in $\mathcal{PI}$ are equivalent with respect to an arbitrary reindexing of the machines and of the jobs. Also, the sequences of an isomorphism class in $\mathcal{GI}$ are equivalent with respect to an arbitrary renumbering of the completely interchangeable machine set and job set. Finally, any two sequences belonging to the same isomorphism class in $\mathcal{SI}$ are equivalent in the sense that the orientations of each operations are also allowed in addition to an arbitrary reindexing of the completely interchangeable machine set and job set.

If an isomorphism $\cong$. applied to the sequence A results to the same sequence A, it is called a sequence automorphism. Naturally, there corresponds to three types of sequence automorphisms, namely, permutation automorphism, graph automorphism and structure automorphism. The collection of all automorphisms in each class forms a subgroup of the group of isomorphisms with respect to their corresponding classes. Because, ((2, 1, 3), (2, 1, 3), Φ) $A = A$ holds if we have the sequence $A = \begin{pmatrix} 2 & 3 & 1 \\ 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix}$, the mapping ((2, 1, 3), (2, 1, 3), Φ) is a graph automorphism.

Definition 3 A sequence A = $[a_{ij}]_{n\times m}$ is called in normal form if $a_{11} = 1$, $a_{1j} < a_{1l}$ $(2 \le j < l \le m)$ and $a_{i1} < a_{k1}$ $(2 \le i < k \le n)$.

Two sequences A and B are graph isomorphic if and only if their associated (transitive) sequence graphs G_A and G_B are isomorphic in terms of graph theory. In [3] a polynomial time Sequence Isomorphism Algorithm (*SIA*) is presented in order to decide whether two sequences are graph isomorphic. The main idea of this algorithm is to put one of the given sequences in normal form and to make appropriate permutations of the other so as to put it as the first. If there is no success, one concludes that the given sequences are not graph isomorphic. In this way, for given $n \times m$ sequences A and B, the permutation isomorphism of A and B is decidable in $O(\max\{mn^2, m^2n\})$ time. With the same time complexity, the graph isomorphism of two sequences A and B as well as the sequence graph isomorphism between the transitive sequence graphs G_A and G_B are decidable.

The validity of the following theorem follows from the summarised *SIA* (cf. [8]). The summarised *SIA* is a compact form of the *SIA* in [3] and the concept of structure isomorphism of sequences in [4]. Moreover, the time and space complexities of compact *SIA* remain similar to that of *SIA*. Therefore,

Theorem 3 *Let A and B be $n \times m$ sequences. Then the isomorphism of A and B is decidable in $O(\min\{mn^2, m^2n\})$ time.* ■

Algorithm 2 The Sequence Isomorphism (*SIA*)

Input : Two arbitrary $n \times m$ sequences $A = [a_{ij}]$ and $B = [b_{ij}]$.

Step 1 : *Find* $[\pi_{r_A}, \pi_{c_A}]$ *such that* $[\pi_{r_A}, \pi_{c_A}]\, A = A^n$, *where sequence* A^n *is normal*;

Step 2 : *For all entries* o_{ij} with $b_{ij} = 1$ *in* **B** *do*:

1. Find $[\pi_{r_B}, \pi_{c_B}]$ *such that* $[\pi_{r_B}, \pi_{c_B}]\, B = B^n$, *where* $b^n_{11} = b_{ij}$ *and* B^n *is normal*;
2. If $B^n = A^n$, return the isomorphism $(\pi_{r_A}\, \pi^{-1}_{r_B}, \pi_{c_A}\, \pi^{-1}_{c_B})$, ***stop***;
3. *For* $m = n$, *if* $B^{n^T} = A^n$, ***return*** *the isomorphism* $(\pi_{r_A}\, \pi^{-1}_{r_B}, \pi_{c_A}\, \pi^{-1}_{c_B}, \Phi)$, ***stop*** ;
4. *If* $B^n = A^n_{-1}$, ***return*** *the isomorphism* $(\pi_{r_A}\, \pi^{-1}_{r_B}, \pi_{c_A}\, \pi^{-1}_{c_B}, \Psi)$, ***stop*** ;
5. *For* $m=n$, *if* $B^{n^T} = A^n_{-1}$, ***return*** *the isomorphism* $(\pi_{r_A}\, \pi^{-1}_{r_B}, \pi_{c_A}\, \pi^{-1}_{c_B}, \Phi, \Psi)$, ***stop*** ;

Output : *An isomorphism* $(\pi_r, \pi_c, \Phi, \Psi)$ *if it exists.*

$\begin{pmatrix}1&2\\2&1\end{pmatrix}$	$\begin{pmatrix}2&1\\1&2\end{pmatrix}$		
$\begin{pmatrix}1&2\\2&3\end{pmatrix}$	$\begin{pmatrix}2&1\\3&2\end{pmatrix}$	$\begin{pmatrix}2&3\\1&2\end{pmatrix}$	$\begin{pmatrix}3&2\\2&1\end{pmatrix}$
$\begin{pmatrix}1&2\\4&3\end{pmatrix}$	$\begin{pmatrix}2&1\\3&4\end{pmatrix}$	$\begin{pmatrix}3&4\\2&1\end{pmatrix}$	$\begin{pmatrix}4&3\\1&2\end{pmatrix}$
$\begin{pmatrix}1&4\\2&3\end{pmatrix}$	$\begin{pmatrix}2&3\\1&4\end{pmatrix}$	$\begin{pmatrix}3&2\\4&1\end{pmatrix}$	$\begin{pmatrix}4&1\\3&2\end{pmatrix}$

Table 1: Class Distribution of 2×2 Sequences

Table 1 Illustrates rowwise distribution of all sequences for $O2 \mid n = 2 \mid \gamma$ into different sequence isomorphism classes. Namely, four different rows represent the four permutation isomorphic classes whereas the last two rows together give one class of graph isomorphism as well as structure isomorphism; here, four permutation classes and three graph/structure isomorphism classes. Clearly, the class representatives form a system of distinct representatives for the isomorphism classes of sequences. The number of nonisomorphic sequences can be calculated by applying Algorithm 2 on the set of all sequences, but this procedure is not sufficient from computational point of view. We refer to [4, 12] for its implementation with small formats. However, a set of sequences generated only on the class representatives (for example, the lexicographically minimal) of their equivalence classes plays an important role for further investigation . In [4, 12] , properties of sequence isomorphisms are applied to investigate a set of sequences which contains at least one optimal solution independent of processing times.

5. Concluding Remarks

In this paper, we have considered recognition algorithms for special class of graphs, so-called transitive shop graphs arising from shop scheduling problems. Investigations of such efficient algorithms allow us to study interesting properties of the 2-dimensional Hamming graph $K_n \times K_m$ in shop problems. Namely, both cyclic as well as acyclic orientations of $K_n \times K_m$ provide useful hints for further structural analysis of shop problems. A study of isomorphic properties of sequences not only provides a decomposition of sequences but also gives some useful hints for further structural analysis of sequences from algebraic point of view. On the other hand, a study of isomorphic properties of cyclic orientations of $K_n \times K_m$ could be one of the interesting subjects from theoretical point of views. Results included in this field are again of both theoretical and practical interests.

REFERENCES

[1] Aurenhammer, F., Hagauer, J., Imrich, W. (1992), *Cartesian Graph factorisation at Logarithmic Cost Per Edge. Comput Complexity*, 2,331–349.

[2] Bräsel, H., Dhamala T.N. (2001), *On Algebraic Properties in Shop Scheduling Problems*. Preprint, *Otto–von–Guericke University*, Magdeburg, Germany.

[3] Bräsel, H., Harborth, M., Willenius, P. (2001), *Isomorphism for Digraphs and Sequences of Shop Scheduling Problems. The Journal of Combinatorial Mathematics and Combinatorial Computing*, 37, 115–128.

[4] Bräsel, H., Harborth, M., Tautenhahn, T., Willenius, P. (1999), *On the Set of Solutions of the Open Shop Problem. Annals of Operations Research*, 92, 241–263.

[5] Bräsel, H., (1990), *Latin Rectangle in Scheduling Theory*, Otto–von–Guericke *University*, Magdeburg, Germany Habilitationsschrift (in German).

[6] Chen, J. (1994), *A Linear-time Algorithm for Isomorphism of Graphs of Bounded Average Genues*. SIAM J. Discrete Mathematics, Vol. 7, No. 4, 614–631.

[7] Dénes, J. and Keedwell, A.D. (1974), *Latin Squares and their Applications. Adadémiai Kiadó*, Budapest.

[8] Dhamala, T. N. (2002), *Shop Scheduling Solution–Spaces with Algebric Characterisations*. Otto–von–Guericke University, Magdeburg, Germany. Ph.D. Thesis.

[9] Garey, M. R. and Johnson, D.S. (1979). *Computers and Intractability*, A Guide to the Theory of NP–Completeness. W.H. Freeman & Co., New York.

[10] Golumbic, M.C. (1980), *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New Work.

[11] Graham, R. E., Lawler, E.L., Lenstra, J.K., and Rinnooy Kan, A.H.G. (1979), *Optimisation and Approximation in Deterministic Sequencing and Scheduling. A survey*, Annals of Discrete Mathematics, 5, 287–326.

[12] Harborth, M., (1999), *Structural Investigation of Shop Scheduling Problems, Number Problems, Potential Optimality and New Enumeration Algorithms.* Otto–von–Guericke University, Magdeburg, Germany. Ph.D. Thesis (in German).

[13] Imrich, W., Klavžar, S. (1997), *Recognising Hamming Graphs in Linear Time and Space*. Information Processing Letters, 63, 91–95.

[14] Imrich, W., Klavžar, S. (1996), *On the Complexity of Recognising Hamming Graphs and Related Classes of Graphs*. European Journal of Combinatorics, 17, 209–221.

[15] Kahan, A.B., (1962), *Topological Sorting of Large Networks, Scientific and Business Applications. Communications of the* ACM, 5, 558–562.

[16] Knuth, D.E., (1968), *Fundamental Algorithms, The Art of Computer Programming*, 1, Addison–Wesley.

[17] Luks, E.M. (1982), *Isomorphism of Graphs of Bounded Valence Can Be Tested in Polynomial Time. Journal of Computer and System Science*, 25, 42–65.

[18] Miller, G.L. (1979), *Graph Isomorphisms, General Remarks, Journal of Computer and System Sciences*, 18, 128–142.

[19] Ponomarenko, I.N. (1992), *Polynomial Time Algorithms for Recognising and Isomorphism Testing of Cyclic Tournaments*. Acta, Appl. Math., 29, 139–160.

[20] Rédei, L., (1934), *Ein Kombinatorischer Satz*, Acta, Litt. Sci. Szeged, 7, 39-43.

[21] Sabidussi, G. (1960), *Graph Multiplication. Math.* Z. 72, 446–457.

[22] Sussmann, B., (1972), *Scheduling Problems with Interval Disjunctions. Z. Operations Research*, 16, 165–178.

[23] Winkler, P. (1984), *Isometric Embeddings in Products of Complete Graphs*. Discrete Applied Mathematics, 7, 221–225.

TANKA NATH DHAMALA
Tribhuvan University
Institute of Science and Technology
Central Department of Mathematics
Kathmandu, Nepal.