4-1-1988

# Partitioning of Uniform Dependency Algorithms for Parallel Execution on MIMD/ Systolic Systems

Weijia Shang
*Purdue University*

Jose A. B. Fortes
*Purdue University*

# Partitioning of Uniform Dependency Algorithms for Parallel Execution on MIMD/Systolic Systems

Weijia Shang
Jose A. B. Fortes

TR-EE 88-18
April 1988

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

# PARTITIONING OF UNIFORM DEPENDENCY ALGORITHMS FOR PARALLEL EXECUTION ON MIMD/SYSTOLIC SYSTEMS

*Weijia Shang and Jose A. B. Fortes*
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

**Abstract:** An algorithm can be modeled as an *index set* and a set of *dependence vectors*. Each index vector in the index set indexes a computation of the algorithm. If the execution of a computation depends on the execution of another computation, then this dependency is represented as the difference between the index vectors of the computations. The *dependence matrix* corresponds to a matrix where each column is a dependence vector. An *independent partition* of the index set is such that there are no dependencies between computations that belong to different blocks of the partition. This report considers uniform dependence algorithms with any arbitrary kind of index set and proposes two very simple methods to find independent partitions of the index set. Each method has advantages over the other one for certain kind of application, and they both outperform previously proposed approaches in terms of computational complexity and/or optimality. Also, lower bounds and upper bounds of the cardinality of the maximal independent partitions are given. For some algorithms it is shown that the cardinality of the maximal partition is equal to the greatest common divisor of some subdeterminants of the dependence matrix. In an MIMD/multiple systolic array computation environment, if different blocks of an independent partition are assigned to different processors/arrays, the communications between processors/arrays will be minimized to zero. This is significant because the communications usually dominate the overhead in MIMD machines. Some issues of mapping partitioned algorithms into MIMD/systolic systems are addressed. Based on the theory of partitioning, a new method is proposed to test if a system of linear Diophantine equations has integer solutions.

This research was supported in part by the National Science Foundation under Grant DC1-8419745 and in part by the Innovative Science and Technology Office of the Strategic Defense Initiative Organization and was administered through the Office of Naval Research under contract No. 00014-85-k-0588.

## 1. INTRODUCTION

Parallel processing holds the potential for computational speeds that surpass by far those achievable by technological advances in sequential computers. This potential is predicated on two often conflicting assumptions, namely, that many computations can take place concurrently and that the time spent in data exchanges between these computations is small. In order to meet these assumptions, algorithms and/or programs must be partitioned into computational blocks that can execute in parallel and have communication requirements efficiently supported by the target parallel computer. Ideally, it may be desirable to identify, if at all possible, the independent computational blocks of a program, i.e., those that require no data communication between them. This report describes two practical and computationally inexpensive approaches to achieve this goal. It is based on a sound mathematical framework which yields optimal results for a meaningful class of algorithms and they outperform approaches proposed in extant work.

The identification of a possible partition of an algorithm or program can be done by the user, by the analysis phase of an optimizing compiler or by the machine at run time [GaPe85]. The techniques proposed in this report, while usable by a patient and dedicated programmer, are best suited for an optimizing compiler. They address the specific problem of identifying *independent partitions* of an algorithm with goals that are similar to those of the early works of D.A. Padua [Pad79] and J. Peir, D. Gajski and R. Cytron [Pei86], [PeGa86], [PeCy87]. The focus of these efforts is on the optimization of programs consisting mainly of nested loops with regular data dependencies. The techniques proposed in those papers are intended to complement many other tools for the analysis and restructuring of sequential programs for execution in multiprocessing machines [Baetal79], [Paetal80], [Wol82], [Kuetal84], [Poetal86]. A related potential application of partitioning techniques is in the design of algorithmically specialized concurrent VLSI architectures [MoFo86].

In this report, nested loop programs with regular data dependencies are modeled as *uniform dependence algorithms* which resemble the uniform recurrence equations considered in [Kaetal67] and the linear recurrences of [PeCy87]. Data dependencies are represented as dependence vectors (with as many entries as the number of nested loops) that describe the distance between dependent computations in terms of loop indices ( the vectors are called dependence distance vectors in [PeCy87] and are also considered in [Wol82] and [Cyt86] in a complemented form ). Dependence vectors are collected in a matrix, the *dependence matrix,* which is used in this report and in [Pad79], [Pei86] and [PeCy87] to identify independent partitions as briefly described in the following paragraphs.

The *greatest common divisor method* [Pad79], [PeCy87] considers, for each row of the dependence matrix, the greatest common divisor of the entries in that row. The resulting greatest common divisors are used to partition the iteration space of the program (also called the index set) and the cardinality of the resulting partition is the product of the greatest common divisors. In addition, an "alignment" method is provided in [Pad79] which allows in some cases the transformation of dependencies so that the value of the greatest common divisors is increased. For a given set of dependencies, this approach yields a unique independent partition which is not necessarily optimal. In some cases, when all of the greatest common divisors equal unity, the number of the blocks in the partition is one, i.e., the whole program.

In the *minimum distance method* [Pei86], [PeCy87], the dependence matrix is transformed into an upper triangular matrix which is then used to identify an independent partition. For some algorithms the cardinality of the partition is the product of the diagonal elements of the upper triangular matrix. This approach yields partitions which are better than those obtained through the greatest common divisor method. However, the computational complexity of this method is high (though affordable according to [PeCy87]) and the optimality is not guaranteed.

In the first method, called *Smith normal form approach,* proposed in this report, a matrix is used to find independent partitions of uniform dependence algorithms and the block a given index vector belongs to can be identified by the product of the matrix with the index vector. In the second method, called *partitioning vector approach,* proposed in this report, a set of vectors defined later in Section 4 is derived from the dependence matrix. These vectors are used to find independent partitions of uniform dependence algorithms with any arbitrary kind of index set. The block to which a given index point belongs to can be identified by simply computing the dot products of each of the vectors by the index point. Both methods provide lower bound and upper bound on the cardinality of the resulting partitions. The first method yields maximal partitions for any algorithms with uniform dependence structure and the second method gives maximal partitions for a meaningful class of algorithms. Comparisons of this two methods proposed in this report and the minimum distance method are provided in Section 6.

The organization of this report is as follows. Section 2 presents basic definitions and notation. Section 3 describes the Smith normal form approach where the notion of Smith normal form is introduced and the procedure of finding independent partitions by the Smith normal form approach is presented. Sections 4 and 5 present the partitioning vector approach. In Section 4, partitioning and separating vectors are defined and three types of independent algorithm partition by these vectors are derived. In Section 5, a procedure finding an independent algorithm partition by the partitioning vectors is presented and sufficient conditions for the resulting partition to be maximal are discussed. Section 6 compares the methods proposed in this report and the minimum distance method. Section 7 discusses some implementation issues of mapping the partitioned algorithms into MIMD/systolic arrays. Finally, Section 8 concludes this report and points out some future work. Based on the partitioning vector approach, necessary and sufficient conditions are derived for a system of linear Diophantine equations to have an integer solution which is presented in Appendix 1.

## 2. BASIC DEFINITIONS AND NOTATION

Throughout this report, *sets*, *matrices* and *row vectors* are denoted by capital letters, *column vectors* are represented by lower case symbols with an overbar and *scalars* correspond to lower case letters. The *transposes* of a vector $\bar{v}$ and a matrix M are denoted $\bar{v}^T$ and $M^T$, respectively. The symbol $E_i$ denotes the row vector whose entries are all zeros except that the ith entry is equal to unity. The vector $\underline{1}$ (or $\underline{0}$) denotes the row vector or column vector whose entries are all ones (or zeroes). The dimensions of 1 and 0 and whether they denote row or column vectors are implied by the context in which they are used. The vector space spanned by a set of vectors $S=\{\bar{v}_1, \bar{v}_2, ..., \bar{v}_k\}$ is denoted $sp\{\bar{v}_1, \bar{v}_2, ..., \bar{v}_k\}=sp\{S\}$ and its dimension (i.e., the number of linearly independent vectors in S) is denoted $dim\{S\}$. The symbol $I$ denotes the identity matrix. The rank of a matrix A is denoted *rank(A)* and the determinant of matrix A is represented by $det A$. The set of rational numbers, the real space and the set of integers are denoted $Q$, $IR$ and $Z$, respectively. The set of non-negative integers and the set of positive integers are denoted $N$ and $N^+$, respectively. The empty set is denoted $\emptyset$ and the notation $A-B$ denotes the set $\{x: x\in A, x\notin B\}$. The notation $|S|$ means the cardinality of set S and $|\alpha|$ represents the absolute value of scalar $\alpha$. Let a, b, c, $d\in Z$ and $a>0$, the notation $a\,|\,b$ means "a divides b", i.e., $b=ca$, and $b(\text{mod } a)=d$ if and only if $b=d+ca$ where $0\leq d<a$. As defined in [Kaetal67], a function $f(x_1, x_2, ..., x_m)$ is *strictly dependent* on $x_i$ if for any arbitrary fixed values $b_j$ assigned to $x_j$, $j\neq i$, $f(b_1, ..., b_{i-1}, x_i, b_{i+1}, ..., b_m)$ is not a constant function. As a final remark, if the element $a$ belongs to a set S, the notation $a\in S$ is used and this notation is "abused" to indicate also that a column vector $\bar{m}_j$ (or a row vector $M_j$) is a column (row) of a matrix M, i.e., $\bar{m}_j\in M$ ($M_j\in M$) means $\bar{m}_j$ ($M_j$) is a column (row) vector of matrix M.

The algorithms of interest in this report are the so-called uniform dependence algorithms defined as follows.

**Definition 2.1 (Uniform dependence algorithm)** A *uniform dependence algorithm* is an algorithm that can be described by an equation of the form

$$v(\bar{j}) = f_{\bar{j}}(v(\bar{j}-\bar{d}_1), v(\bar{j}-\bar{d}_2), \cdots, v(\bar{j}-\bar{d}_m)) \tag{2.1}$$

where

(1) $\bar{j}\in J\subset Z^n$ is an index point, J is the *index set* of the algorithm and $n\in N^+$ is the number of components of $\bar{j}$;

(2) $f_{\bar{j}}$ is the computation indexed by $\bar{j}$, i.e., a single-valued function computed " at point $\bar{j}$ " in a single unit of time and strictly dependent on each of its arguments;

(3) $v(\bar{j})$ is the value computed "at $\bar{j}$", i.e. the result of computing the right hand side of (2.1) and

(4) $\bar{d}_i\in Z^n$, $i=1, ..., m$, $m\in N$ are *dependence vectors*, also called *dependencies*, which are constant (i.e. independent of $\bar{j}\in J$); the matrix $D=[\bar{d}_1, ..., \bar{d}_m]$ is called the *dependence matrix* and $\text{rank}(D)\leq \min\{n,m\}$ is denoted by $m'$.

The class of uniform dependence algorithms is a simple extension of the class of computations described by uniform recurrence equations [Kaetal67]. The main difference is that uniform dependence algorithms allow for different functions to be computed (in a unit of time) at different points of the index set. From a practical viewpoint, uniform dependence algorithms can be easily related to programs where (1) a single statement appears in the body of a multiply nested loop and (2) the indices of the variable in the left hand side of the statement differ by a constant from the corresponding indices in all references to the same variable in the right hand side. Alternative computations can occur in each iteration as a result of a single conditional statement as long as data dependencies do not change. Nested loop programs with multiple statements can also use the techniques of this report together with the alignment method discussed in [Pad79] and [PeCy87]. For the purpose of this report, only structural information of the algorithm, i.e., the index set J and the dependence matrix D, is needed. Other information such as what computations occur at different points and where and when input/output of variables takes place can be ignored. Therefore, a uniform dependence algorithm with index set J and dependence matrix D is hereon characterized simply by the pair (J, D). Also, as in Definition 2.1, the letters n, m and m′ always denote the dimension of points in J, the number of dependence vectors and the rank of the dependence matrix D, respectively.

**Definition 2.2 (Algorithm dependence graph and connectivity) :** The *dependence graph* of an algorithm (J, D) is the nondirected graph (J, E) where J is the set of nodes of the graph and $E = \{(\bar{j}', \bar{j}) : \bar{j} - \bar{j}' = \bar{d}_i$ or $\bar{j}' - \bar{j} = \bar{d}_i, \bar{d}_i \in D, \bar{j}', \bar{j} \in J\}$ is the set of edges. Two index points $\bar{j}, \bar{j}'$ are *connected* if there exist index points $\bar{j}_1, \ldots, \bar{j}_l \in J$ such that $(\bar{j}, \bar{j}_1), (\bar{j}_1, \bar{j}_2), \ldots, (\bar{j}_{l-1}, \bar{j}_l), (\bar{j}_l, \bar{j}') \in E$.

**Definition 2.3 (Independent partition, maximal independent partition and partitionability):** Given an algorithm (J, D) and the corresponding dependence graph (J, E), let $\mathcal{P} = \{J_1, \ldots, J_q\}$, $q \in N^+$, be a partition of J. If for any arbitrary points $\bar{j}_1 \in J_i$ and $\bar{j}_2 \in J_l$, $i \neq l$ and $0 < i, l \leq q$, $(\bar{j}_1, \bar{j}_2) \notin E$, then $\mathcal{P}$ is an *independent partition* of the algorithm (J, D). The sets $J_i$, i=1, ..., q, are called *blocks* of partition $\mathcal{P}$. For an independent partition $\mathcal{P}$, if any two arbitrary points $\bar{j}, \bar{j}' \in J_i$, i=1, ..., q, are connected in the dependence graph, then $\mathcal{P}$ is the *maximal independent partition* of (J, D) and is denoted $\mathcal{P}_{max}$. The cardinality of the maximal independent partition $|\mathcal{P}_{max}|$ is referred to as the *partitionability* of the algorithm (J, D).

Informally, an independent partition of the index set J is such that there are no dependencies between computations which belong to different blocks of the partition. In graph theoretical terms, each block of an independent partition of (J, D) corresponds to a component of its dependence graph (J, E)

Generally speaking, the shape and the size of the index set influence the partitionability of the algorithm because of boundary conditions. Consider two algorithms (J, D) and (J′, D′) such that D′=D and $J' = J \cup \{\bar{j}\}$, i.e., they differ only in the size of the index sets. The corresponding dependence graphs (J, E) and (J′, E′) can be such that $\bar{j}_1, \bar{j}_2 \in J$ are not connected in (J, E) but are connected in (J′, E′) because it is possible that $E' = E \cup \{(\bar{j}, \bar{j}_1), (\bar{j}, \bar{j}_2)\}$. In other words, $\bar{j}_1$ and $\bar{j}_2$ can belong to different blocks of the maximal independent partition of (J, D) but belong to the same block of the maximal independent partition of (J′, D′). The following example illustrates this concept.

**Example 2.1:** Consider algorithms (J, D) and (J′, D), where

$$D = \begin{bmatrix} 3 & 0 \\ -3 & 2 \end{bmatrix}, \quad J = \{[j_1, j_2]^T : 0 \leq j_1, j_2 \leq s, s \in N^+\}$$

and

$$J' = \{\bar{j} : \begin{bmatrix} -1 & 1 \\ 0 & -1 \\ -1 & 0 \end{bmatrix} \bar{j} \leq \begin{bmatrix} 0 \\ 0 \\ s \end{bmatrix}, s \in N^+\}$$

Figure 2.1 shows the index sets J and J′ where s=8. These two algorithms have the same dependence matrix but different index sets. In J', point $[1,1]^T$ is not connected to any other points in J' because $[1,1]^T \pm \bar{d}_i$, i=1,2, do not belong to J'. However, in J it is connected to $[4,1]^T \in J'$. End of example.

The dependence of the partitionability of an algorithm (J, D) on the shape and size of its index set J is a complicated issue and has practical implications. For example, in many programs, the loop bounds are not known at compile time and partitions must be identified which are independent of the size and shape of the index set and based solely on data dependencies. To concentrate on the relationship between the structure of the dependence vectors and the partitionability of the algorithm, the following concepts are introduced.

**Definition 2.4 (Pseudo-connectivity):** Given an algorithm (J, D), two points $\bar{j}, \bar{j}' \in J$ are *pseudo-connected* if there exists a vector $\bar{\lambda} \in Z^m$ such that $\bar{j}=\bar{j}'+D\bar{\lambda}$.

As an example of pseudo-connectivity, in algorithm (J', D) of Example 2.1, point $[1,1]^T$ is pseudo-connected to $[4,0]^T$ through point $[1,3]^T \in (J - J')$.

**Definition 2.5 (Pseudo-independent partition, maximal pseudo-independent partition and pseudo-partitionability):** Given an algorithm (J, D), let $P=\{J_1, \ldots, J_q\}$ be a partition of J. If any two arbitrary points $\bar{j}_1 \in J_i \in P$ and $\bar{j}_2 \in J_l \in P$, $i \neq l$, are not pseudo-connected, then P is a *pseudo-independent partition* of the algorithm (J, D). If P is a pseudo-independent partition and any two arbitrary points $\bar{j}, \bar{j}' \in J_i$, i=1, ..., q, are pseudo-connected, then P is the *maximal pseudo-independent partition* of (J, D) and is denoted $P_{max}$. The cardinality of the maximal pseudo-independent partition $|P_{max}|$ is referred to as the *pseudo-partitionability* of the the algorithm (J, D).

In many practical cases, e.g., when "while" loops are present in a program, it is also convenient to consider algorithms whose index sets are arbitrarily large along one or more dimensions. The general case, i.e., when this applies to all dimensions, is captured in the following definition and is also considered in this report.

**Definition 2.6 (Semi-infinite index set):** An index set J is *semi-infinite* if it takes the following form:

$$J = \{\bar{j} = [\, j_1, \ldots, j_n \,]^T \colon 0 \leq j_i < \infty, i = 1, \ldots, n\} \tag{2.2}$$

**Example 2.2** Consider the algorithm (J, D), where $D=\begin{bmatrix} 2 & -3 \\ -1 & 2 \end{bmatrix}$ and $J=N^2$ is semi-infinite, i.e., $J = \{\, \bar{j} = [\, j_1, j_2 \,]^T \colon 0 \leq j_1, j_2 < \infty\}$. The index set J is partially shown in Figure 2.2. The maximal partition $\mathcal{P}_{max}=\{J_1, J_2, J_3, J_4\}$ where $J_1=\{[0,0]^T\}$, $J_2=\{[1,0]^T\}$, $J_3=\{[0,1]^T, [2,0]^T\}$ and $J_4=\{\bar{j} \colon \bar{j} \in (J - \bigcup_{i=1}^{3} J_i)\}$. Points $\bar{j}_1= [0,0]^T$ and $\bar{j}_2=[0,1]^T$ are not actually connected in the dependence graph of the algorithm. However, they are pseudo-connected by Definition 2.4 since $\bar{j}_2=\bar{j}_1 + D\bar{\lambda}$, $\bar{\lambda}= [3,2]^T$. Intuitively, $\bar{j}_1$ and $\bar{j}_2$ are connected through points $[2,-1]^T$, $[4,-2]^T$, $[6,-3]^T$ and $[3,-1]^T$ which are not in J. $\mathcal{P}_{max}$ is not a pseudo-independent partition. Since $\det D=1$, equation $D\bar{\lambda} = \bar{j} - \bar{j}'$ always has an integer solution for $\bar{\lambda}$. So any two arbitrary points in J are pseudo-connected to each other. This implies that there is only one pseudo-independent partition P={J} which is also the maximal pseudo-independent partition. End of example.

At this point, some comments are in order. First, by Definitions 2.3 and 2.5, a pseudo-independent partition is also an independent partition regardless of the shape and size of the index set. However, an independent partition is not necessarily a pseudo-independent partition. This is due to the fact that $\bar{j}_1, \bar{j}_2 \in J$ are pseudo-connected if they are connected and the reverse is not necessarily true. Secondly, for practical purposes, it is sufficient and more efficient to identify pseudo-independent partitions instead of independent partitions for the reasons explained next. Blocks of independent partitions that are not blocks of a pseudo-independent partition and contain only a few index points (hereon called *boundary blocks*) always occur at or near the boundaries of an index set. This can be shown for the general case when J is semi-infinite. In fact, according to Lemma 3 in [Kaetal67], there exists always a point $\bar{p}=[p_1, p_2, ..., p_n]^T \in J$ such that for any arbitrary points $\bar{j}=[j_1, j_2, ..., j_n]^T \in J$ and $\bar{j}'=[j'_1, j'_2, ..., j'_n]^T \in J$ beyond $\bar{p} \in J$ (i.e., $j_i \geq p_i$ and $j'_i \geq p_i$, $1=1, ..., n$), $\bar{j}$ and $\bar{j}'$ are connected in the dependence graph if and only if they are pseudo-connected. Boundary blocks are typically such that their individual cardinalities are very small in relation to the sizes of the algorithm and pseudo-independent blocks. As a consequence, little additional speed-up can result from executing boundary blocks concurrently with other blocks. Moreover, assigning small boundary blocks and other large pseudo-independent blocks to different processors of a multiprocessor can cause a non-balanced load distribution and inefficient system operation. In addition, as pointed out before, when index sets are known only at run time, it is not possible to determine the boundary blocks. Finally, many algorithms are such that they have the same partitionability and pseudo- partitionability. For all of the above reasons, this report considers hereon only the problem of identifying pseudo-independent partitions of an algorithm.

## 3. SMITH NORMAL FORM APPROACH

This section discusses an approach, called Smith normal form approach, of finding maximal pseudo-independent partitions where the Smith normal form (abbreviated SNF) of the dependence matrix D is used. First, a theorem about the SNF is restated and followed by the definitions of the partitioning matrix and the displacement vector. These concepts are used then to define a partition of index set J which is also the maximal pseudo-independent partition of the algorithm. Then a procedure is presented which constructs the maximal pseudo-independent partition of a given algorithm. Complexity of the procedure is also discussed.

**Theorem 3.1 (Smith normal form)** [Sch86, pp. 50]: Given a matrix $D \in Z^{n \cdot m}$, there exist two *unimodular* (A non-singular matrix is unimodular if its elements are integral and its determinant is $\pm 1$) matrices $U \in Z^{n \cdot n}$ and $V \in Z^{m \cdot m}$ such that

$$UDV=S=\begin{bmatrix} s_1 & 0 & ... & 0 & 0 & ... & 0 \\ 0 & s_2 & ... & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & ... & s_{m'} & 0 & ... & 0 \\ 0 & 0 & ... & 0 & 0 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... \\ 0 & 0 & ... & 0 & 0 & ... & 0 \end{bmatrix}.$$

S is called the Smith normal form (abbreviated SNF) of matrix D, S is unique, $s_1, ..., s_{m'}$ are positive integers, $s_1 | s_2 | ... | s_{m'}$, $\prod_{i=1}^{k} s_i$, $k=1, ..., m'$, is the greatest common divisor of subdeterminants of order k of the dependence matrix D and m' is the rank of the dependence matrix D. □

More details and explanations about SNF can be found in [Sch86, pp. 50] and [VeFr21].

**Example 3.1:** Consider the algorithm (J, D) studied in [PeCy87]. $D=\begin{bmatrix} 1 & 2 & 0 \\ -2 & 4 & 4 \\ 4 & -1 & 2 \end{bmatrix}$ and $J=\{[j_1, j_2, j_3]^T: 1 \leq j_i \leq 16, i=1,2,3\}$. The matrices U, S=SNF and V is as follows.

$$U=\begin{bmatrix} 1 & 0 & 0 \\ 2 & -1 & -1 \\ -14 & 9 & 8 \end{bmatrix}, \quad S=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 52 \end{bmatrix}, \quad V=\begin{bmatrix} 1 & -2 & -12 \\ 0 & 1 & 6 \\ 0 & 0 & 1 \end{bmatrix}$$

Clearly UDV=S and U and V are unimodular. End of example.

**Definition 3.1 (Partitioning matrix and displacement vector):** Given an algorithm (J, D), the matrix U such that UDV=S is the SNF of D is called *partitioning matrix* of (J, D). Let $s_1, ..., s_{m'}$ be the non-zero diagonal elements of S, then vector $\bar{s}=[s_1, ..., s_{m'}, \infty, ..., \infty]^T \in (N^+)^n$ is called *displacement vector* of (J, D).

**Definition 3.2 (U-partition):** Let U be a partitioning matrix of algorithm (J, D), the partition of index set J $P_U=\{J_{\bar{y}_1}, ..., J_{\bar{y}_t}\}$, where $\bar{y}_i \in Z^n$, i=1, ..., t, is called the *U-partition* of algorithm (J, D) if $J_{\bar{y}_i}=\{\bar{j}: U\bar{j}(\text{mod } \bar{s})=\bar{y}_i, \bar{j}\in J\}$†.

**Example 3.2:** Consider the algorithm of Example 3.1. U is the partitioning matrix; $\bar{s}=[1,1,52]^T$ is the displacement vector; $P_U=\{J_{[0,0,0]^T}, ..., J_{[0,0,51]^T}\}$ is the U-partition where $J_{[0,0,i]^T}=\{\bar{j}: U\bar{j}(\text{mod } \bar{s})=[0,0,i]^T\}$, i=0, ..., 51. End of example.

It is clear that $P_U$ is a partition of the index set J because for each $\bar{j}\in J$, $U\bar{j}(\text{mod } \bar{s})$ is unique. Actually, $P_U$ is an independent partition. To show the independence of U-partition, the following lemma is introduced first and followed by a theorem.

**Lemma 3.1:** Given algorithm (J, D), let $\bar{j}_1, \bar{j}_2 \in J$ and $\bar{s}$ be the displacement vector, then $\bar{j}_1$ and $\bar{j}_2$ are pseudo-connected if and only if $U\bar{j}_1(\text{mod } \bar{s})=U\bar{j}_2(\text{mod } \bar{s})$.

Proof (=>): Let's assume that $\bar{j}_1$ and $\bar{j}_2$ are pseudo-connected, then, by Definition 2.4, there exists an integer vector $\bar{\lambda}\in Z^m$ such that $D\bar{\lambda}=\bar{j}_1-\bar{j}_2$. Let $V\in Z^{m\cdot m}$ be such that UDV=S is the SNF of matrix D. Then $SV^{-1}\bar{\lambda}=U(\bar{j}_1-\bar{j}_2)$. V is unimodular implies that $V^{-1}$ is also an integer matrix and therefore, $V^{-1}\bar{\lambda}$ is an integer vector. So, $U(\bar{j}_1-\bar{j}_2)(\text{mod } \bar{s})=SV^{-1}\bar{\lambda}(\text{mod } \bar{s})=\bar{0}$, i.e., $U\bar{j}_1(\text{mod } \bar{s})=U\bar{j}_2(\text{mod } \bar{s})$.

(=>): Let's assume that $U\bar{j}_1(\text{mod } \bar{s})=U\bar{j}_2(\text{mod } \bar{s})$, i.e., $U(\bar{j}_1-\bar{j}_2)=[y_1s_1, y_2s_2, ..., y_{m'}s_{m'}, 0, ..., 0]^T$ where $y_i$, i=1, ..., m', are integers. Let $\bar{y}=[y_1, y_2, ..., y_{m'}, 0, ..., 0]^T\in Z^m$, then $U(\bar{j}_1-\bar{j}_2)=S\bar{y}$. This implies that $\bar{j}_1-\bar{j}_2=U^{-1}UDV\bar{y}$, or $\bar{j}_1-\bar{j}_2=DV\bar{y}$. V is integral implies that $V\bar{y}$ is integral. So, there exists an integral vector $\bar{\lambda}=V\bar{y}$ such that $\bar{j}_1-\bar{j}_2=D\bar{\lambda}$ which means $\bar{j}_1$ and $\bar{j}_2$ are pseudo-connected. □

**Theorem 3.2:** Given an algorithm (J, D), let $U_i$ be the ith row of the partitioning matrix U, i=1, ..., n, $\delta_{iu}=\max\{U_i\bar{j}: \bar{j}\in J\}$, $\delta_{il}=\min\{U_i\bar{j}: \bar{j}\in J\}$ and $x_i=\delta_{iu}-\delta_{il}+1$, i=m'+1, ..., n. The following statements are true:

(1) The U-partition is the maximal pseudo-independent partition, i.e., $P_U=P_{\max}$.

(2) The pseudo-partitionability is bounded above by $\prod_{k=1}^{m'} s_k \prod_{i=m'+1}^{n} x_i$, i.e., $|P_{\max}|=\prod_{k=1}^{m'} s_k \prod_{i=m'+1}^{n} x_i$.

Proof: Let $\bar{s}$ be the displacement vector.

(1) First, $P_U$ is a pseudo-independent partition. This can be proven as follows. For any two arbitrary points $\bar{j}_1\in J_{\bar{y}_i}\in P_U$ and $\bar{j}_2\in J_{\bar{y}_l}\in P_U$, $\bar{y}_i\neq\bar{y}_l$, by Definition 3.2, $U\bar{j}_1(\text{mod } \bar{s})=\bar{y}_i$ and $U\bar{j}_2(\text{mod } \bar{s})=\bar{y}_l$. Because $\bar{y}_i\neq\bar{y}_l$, $U\bar{j}_1(\text{mod } \bar{s})\neq U\bar{j}_2(\text{mod } \bar{s})$. By Lemma 3.1, $\bar{j}_1$ and $\bar{j}_2$ are not pseudo-connected which implies that $P_U$ is pseudo-independent.

Secondly, $P_U$ is the maximal pseudo-independent partition. This can be verified as follows. For any two arbitrary points $\bar{j}_1, \bar{j}_2\in J_{\bar{y}}\in P_U$, by definition of the U-partition (Definition 3.2), $U\bar{j}_1(\text{mod } \bar{s})=U\bar{j}_2(\text{mod } \bar{s})=\bar{y}$. By Lemma 3.1, $\bar{j}_1$ and $\bar{j}_2$ are pseudo-connected. By Definition 2.5, $P_U$ is the maximal pseudo-independent partition.

(2) Let $P_U=\{J_{\bar{y}_1}, ..., J_{\bar{y}_t}\}$. Consider a block $J_{\bar{y}}\in P_U$ where $\bar{y}=[y_1, ..., y_n]^T$. Clearly, $0\le y_i<s_i$, i=1, ..., m' and $\delta_{il}\le y_k\le\delta_{iu}$, k=m'+1, ..., n. So there are at most $s_1\times s_2\times...\times s_{m'}\times x_{m'+1}\times...\times x_n$ distinct block indices, i.e., $|P_{\max}|=|P_U|\le\prod_{k=1}^{m'} s_k \prod_{i=m'+1}^{n} x_i$. □

---

†: Let $\bar{x}=[x_1, ..., x_n]^T\in Z^n$ and $\bar{s}=[s_1, ..., s_{m'}, \infty, ..., \infty]^T$ with $s_i>0$, i=1, ..., m'. The notation $\bar{x}(\text{mod } \bar{s})$ denotes the vector $[x_1(\text{mod } s_1), ..., x_{m'}(\text{mod } s_{m'}), x_{m'+1}, ..., x_n]^T$.

For every $\bar{y}=[y_1, ..., y_n]^T$, $0 \le y_i < s_i$, $i=1, ..., m'$, $\delta_{il} \le y_k \le \delta_{iu}$, $k=m'+1, ..., n$, if there exists at least one index point $\bar{j} \in J$ such that $\bar{j} \in J_{\bar{y}}$, then $J_{\bar{y}} \neq \emptyset$ and $|P_U| = \prod_{k=1}^{m'} s_k \prod_{i=m'+1}^{n} x_i$. Notice that $\det U = \pm 1$. So, for each such vector $\bar{y}$, there always exists an integer vector $\bar{j}$ such that $U\bar{j} (\bmod\ \bar{s}) = \bar{y}$. Therefore, it is reasonable to make the following assumption on algorithms of interest of this report. That is, for each vector $\bar{y}=[y_1, ..., y_n]^T$, $0 \le y_i < s_i$ $i=1, ..., m'$, $\delta_{il} \le y_k \le \delta_{iu}$, $k=m'+1, ..., n$, there is at least one index point $\bar{j} \in J$ such that $\bar{j} \in J_{\bar{y}}$. This assumption makes sense, especially, when J is *dense* (informally, an index set J is dense if any arbitrary point $\bar{j} \in Z^n$ that is inside the boundaries of J belongs to J), and large enough and rank(D)=n. Under this assumption, the following corollary is true.

**Corollary 3.1:** Given algorithm (J, D), let rank(D)=n and $\bar{s}=[s_1, ..., s_n]^T$ be the displacement vector. Then $|P_{\max}| = \prod_{i=1}^{n} s_i$, i.e., the pseudo-partitionability of algorithm (J, D) is equal to the greatest common divisor of subdeterminants of order n of the dependence matrix D.

**Procedure 3.1 (Finding the maximal pseudo-independent partition by SNF approach):**
Input: Algorithm (J, D).
Output: U-partition $P_U$ of algorithm (J, D).
Step 1: Find a partitioning matrix U and the displacement vector $\bar{s}$.
Step 2: For every index point $\bar{j} \in J$, compute $U\bar{j} (\bmod\ \bar{s}) = \bar{y}$ and assign $\bar{j}$ to $J_{\bar{y}}$, the block indexed by $\bar{y}$.
Step 3: $P_U = \{J_{\bar{y}_1}, ..., J_{\bar{y}_t}\}$. Stop.□

The complexity of the procedure is a linear function of the cardinality of the index set J, i.e., the number of the index points of the algorithm, and a polynomial function of n and m, the number of components of the index vectors and the number of the dependence vectors. To find the displacement vector $\bar{s}$, it needs to find the SNF of matrix D. In [KaBa79], a polynomial algorithm is proposed for finding the SNF of any arbitrary matrix $A \in Z^{n \cdot n}$ and the corresponding left and right multipliers U and V such that UAV=SNF. The complexity of the product of the memory space and the total execution time of this algorithm is $O((\max\{n, m\})^{10})$ [KaBa79]. So, the operations of Step 1 is bounded by $O((\max\{n, m\})^{10})$. For Step 2, it needs at most $O(|J| n^2)$ operations to compute $U\bar{j}$ for every index point $\bar{j}$. Therefore, the total complexity is $O((\max\{n, m\})^{10}) + O(|J| n^2)$.

## 4. PARTITIONING VECTOR APPROACH--BASIC RESULTS

Sections 4 and 5 present the partitioning vector approach. In this approach, independent algorithm partitions are determined by two types of vector, called partitioning vectors and separating vectors, which must satisfy certain conditions. Together with some auxiliary terminology they are introduced in Definitions 4.1 and 4.3. These definitions are followed by a theorem and an example which make clear the relation between these vectors and independent algorithm partitions.

**Definition 4.1 (Partitioning vector, determining vector, equal partitioning vector and algorithm coefficient):** Given an algorithm (J, D), $\Pi=[\pi_1, \pi_2, ..., \pi_n] \in Z^{1 \cdot n}$ is a *partitioning vector* of (J, D), if and only if it satisfies the following conditions.
(1) gcd $(\pi_1, \pi_2, ..., \pi_n)=1$†.
(2) There exists a set of $m'$=rank(D) linearly independent dependence vectors $\bar{d}_{t_1}, \bar{d}_{t_2}, ..., \bar{d}_{t_{m'}}$ such that

$$\Pi \bar{d}_{t_1} = \cdots = \Pi \bar{d}_{t_{m'}} = disp\Pi > 0. \tag{4.1}$$

The dependence vectors $\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}$ are called the *determining vectors* of $\Pi$ and $[\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}]$ is called *determining matrix* of $\Pi$. If $\Pi \bar{d}_i (\bmod\ disp\Pi)=0$, $i=1, ..., m$, then $\Pi$ is called an *equal partitioning vector* of (J, D). The constant $\alpha = \gcd(disp\Pi, \alpha_1, ..., \alpha_m)$ where $\alpha_i = \Pi \bar{d}_i (\bmod\ disp\Pi)$, $i=1, ..., m$, is called the

† : gcd $(a_1, ..., a_n)$=the greatest common divisor of $a_1, ..., a_n$.

*algorithm coefficient.*

For a given partitioning vector the set of determining vectors is not necessarily unique and, therefore, displI might not be unique, either. However, given a partitioning vector and a set of determining vectors, displI is unique. Therefore, whenever displI is mentioned, it is associated with a particular set of determining vectors.

By Definition 4.1, if $m'=n$, then for each set of determining vectors $\bar{d}_{t_1}, \ldots, \bar{d}_{t_{n'}}$, the corresponding partitioning vector $\Pi$ is the unique solution that satisfies conditions 1 and 2 in Definition 4.1 and the following system of linear equations:

$$\begin{cases} \Pi(\bar{d}_{t_1} - \bar{d}_{t_2}) = 0 \\ \Pi(\bar{d}_{t_1} - \bar{d}_{t_3}) = 0 \\ \ldots \\ \Pi(\bar{d}_{t_1} - \bar{d}_{t_{n'}}) = 0 \end{cases} \tag{4.2}$$

When $m' < n$, the partitioning vector determined by $m'$ linearly independent dependence vectors $\bar{d}_{t_1}$, ..., $\bar{d}_{t_{n'}}$ is not unique, and of course, it belongs to the solution space of Equation 4.2. In the next section, a closed form expression is provided for a partitioning vector as a solution of Equation 4.2.

A partitioning vector $\Pi$ defines a set of hyperplanes $\Pi\bar{j}\,(\mathrm{mod}\alpha)=c$, $c\in Z$, in the index space. Since an index point lies on only one of the hyperplanes, the index set J can be partitioned according to them, i.e., all points $\bar{j}$ lying on hyperplanes such that for a fixed c, $\Pi\bar{j}\,(\mathrm{mod}\alpha)=c$ belong to the same block of the partition. The following definition states this concept formally.

**Definition 4.2 ($\alpha$-partition):** Let $\Pi$ be a partitioning vector and $\alpha$ be the algorithm coefficient for (J, D). The partition $P_\alpha=\{J_0, \ldots, J_{\alpha-1}\}$ where $J_i=\{\bar{j}:\bar{j}\in J, \Pi\bar{j}\,(\mathrm{mod}\alpha)=i\}$, $i=0,\ldots,\alpha-1$, is called the *$\alpha$-partition* of (J, D).

Clearly, $P_\alpha$ is a partition and it is shown in Theorem 4.1 that $P_\alpha$ is also a pseudo-independent partition.

For the case where $m'<n$, i.e., $\mathrm{rank}(D)\le n$, a necessary condition for two index points $\bar{j}_1, \bar{j}_2 \in J$ to be pseudo-connected is that equation $D\bar{x}=(\bar{j}_1-\bar{j}_2)$ has at least a real solution $\bar{x}\in \mathbb{R}^m$. This motivates the introduction of the following concepts. Let row vector $\Psi_i$ be such that $\Psi_i D=0$. Clearly, there are $n-m'$ linearly independent such vectors, denoted $\Psi_1, \ldots, \Psi_{n-m'}$, and they define a set of hyperplanes

$$\begin{bmatrix} \Psi_1 \\ \ldots \\ \Psi_{n-m'} \end{bmatrix} \bar{j} = \bar{y}, \quad \bar{y} \in Z^{n-m'}, \tag{4.3}$$

in the index space. The index set J can be partitioned such that points lying on the same hyperplane belong to the same block of the partition. It will be clear later that if two index points $\bar{j}_1, \bar{j}_2 \in J$ lie on the same hyperplane defined by (4.3), the equation $D\bar{x}=(\bar{j}_1-\bar{j}_2)$ has a solution. These concepts are formally defined as follows.

**Definition 4.3 (Separating vector and separating matrix):** Given an algorithm (J, D), $\Psi_i=[\psi_{i1}, \ldots, \psi_{in}]\in Z^{1\times n}$ is a *separating vector* of (J, D) if and only if it satisfies the following conditions.

(1) $\gcd(\psi_{i1}, \ldots, \psi_{in})=1$

(2) $\Psi_i D=0$.

Let $\Psi_1, \ldots, \Psi_{n-m'}$ be all the linearly independent separating vectors; the matrix $\Psi=\begin{bmatrix} \Psi_1 \\ \ldots \\ \Psi_{n-m'} \end{bmatrix}$ is called *separating matrix.*

A set of $n-m'$ linearly independent separating vectors $\Psi_1, \ldots, \Psi_{n-m'}$ for algorithm (J, D) can be found by solving the equation in condition 2 of Definition 4.3. The following definition indicates how to use these separating vectors to construct a corresponding algorithm partition.

**Definition 4.4 (Ψ-partition):** Let $\Psi$ be a separating matrix of algorithm (J, D). The partition $P_\Psi=\{J_{\bar{y}_1}, \ldots, J_{\bar{y}_q}\}$ of J is called the *Ψ-partition* of algorithm (J, D) if $J_{\bar{y}_i}=\{\bar{j}: \bar{j}\in J, \Psi\bar{j}=\bar{y}_i\}$, where $\bar{y}_i=[y_{1i}, \ldots, y_{(n-m')i}]^T\in Z^{(n-m')}$ is called the *index* of block $J_{\bar{y}_i}$, $i=1, \ldots, q$.

Clearly, $P_\Psi$ is a partition of J. If $m'=n$, then $P_\Psi=\{J\}$ is a trivial partition since the only separating vector is 0 in this case. As for $P_\alpha$, $P_\Psi$ is actually pseudo-independent as shown later in Theorem 4.1.

Let $J_{\bar{y}} \in P_\Psi$ and consider the subalgorithm $(J_{\bar{y}}, D)$. Clearly, if $\alpha > 1$, subalgorithm $(J_{\bar{y}}, D)$ can be further partitioned by the partitioning vector $\Pi$. In other words, the index set J can be partitioned by a set of hyperplanes

$$\begin{bmatrix} \Pi\bar{j}(\text{mod}\,\alpha) \\ \Psi\bar{j} \end{bmatrix} = \begin{bmatrix} y_0 \\ \bar{y} \end{bmatrix}, \quad y_0\in\{0,1,\ldots,\alpha-1\} \text{ and } \bar{y}\in Z^{n-m'}, \tag{4.4}$$

and all points lying on the same hyperplane belong to the same block of the partition. This partition is formally stated next.

**Definition 4.5 (αΨ-partition):** Let $\Pi$ be a partitioning vector and $\Psi$ be a separating matrix of algorithm (J, D). The partition $P_{\alpha\Psi}=\{J_{\bar{y}_1}, \ldots, J_{\bar{y}_k}\}$ of index set J is called the *αΨ-partition* if $J_{\bar{y}_i}=\{\bar{j}: \bar{j}\in J, \begin{bmatrix} \Pi\bar{j}(\text{mod}\,\alpha) \\ \Psi\bar{j} \end{bmatrix}=\bar{y}_i\}$, where $\bar{y}_i=[y_{0i}, y_{1i}, \ldots, y_{(n-m')i}]^T\in Z^{n-m'+1}$ is called the *index* of block $J_{\bar{y}_i}$, $i=1, \ldots, k$.

Partitioning vectors and separating vectors play a very important rôle in algorithm partition. The next theorem gives some of the motivation for the introduction of these concepts. More specifically, it provides sufficient conditions for two computations to belong to different blocks of an independent partition, in terms of those vectors and the index points associated with the computations. Moreover, it shows that α−partitions, Ψ−partitions and αΨ-partitions are all pseudo-independent.

**Theorem 4.1** : Let $\Pi$ be a partitioning vector, $\alpha$ be the algorithm coefficient and $\Psi$ be a separating matrix of algorithm (J, D), respectively. The following statements are true:

(1) For any two arbitrary points $\bar{j}_1, \bar{j}_2 \in J$, if $\Pi\bar{j}_1(\text{mod}\,\alpha)\neq\Pi\bar{j}_2(\text{mod}\,\alpha)$ then they are not pseudo-connected. Therefore, $P_\alpha$ is a pseudo-independent partition of (J, D).

(2) For any two arbitrary points $\bar{j}_1, \bar{j}_2\in J$, if $\Psi\bar{j}_1\neq\Psi\bar{j}_2$, then they are not pseudo-connected. Therefore, $P_\Psi$ is a pseudo-independent partition of (J, D).

(3) $P_{\alpha\Psi}$ is a pseudo-independent partition.

Proof: Provided in Appendix 2.

**Corollary 4.1:** If algorithm (J, D) has an equal partitioning vector $\Pi$, then $\bar{j}_1, \bar{j}_2\in J$ are not pseudo-connected if $\Pi\bar{j}_1(\text{mod disp}\Pi)\neq\Pi\bar{j}_2(\text{mod disp}\Pi)$ or $\Psi\bar{j}_1\neq\Psi\bar{j}_2$.

As a particular case of Theorem 4.1, Corollary 4.1 is obviously true. If algorithm (J, D) has an equal partitioning vector $\Pi$, then the algorithm coefficient $\alpha=\text{disp}\Pi$. By Theorem 4.1, Corollary 4.1 holds.

**Example 4.1:** Consider algorithm (J, D) where $J=\{[j_1, j_2]^T: 0\le j_1, j_2\le s, s\in N^+\}$ and $D=[\bar{d}]$ where $\bar{d}=[2, 2]^T$. Figure 4.1 shows the index set J for s=4. There is only one possible set of determining vectors $\{\bar{d}\}$. One of the partitioning vectors determined by $\bar{d}$ is $\Pi=[-1, 2]$. It follows that $\text{disp}\Pi=\Pi\bar{d}=2$ and the algorithm coefficient $\alpha=2$. Consider index points $\bar{j}_1=[0, 0]^T$ and $\bar{j}_2=[1, 0]^T$; since $\Pi\bar{j}_1(\text{mod}\,\alpha)=0$ and $\Pi\bar{j}_2(\text{mod}\,\alpha)=1$, by Theorem 4.1, they are not pseudo-connected. There is only one linearly independent separating vector $\Psi_1=[1, -1]$ and a separating matrix is $\Psi=[1, -1]$. Again, consider index points $\bar{j}_1$, $\bar{j}_3=[0, 1]^T$ for which $\Psi\bar{j}_1=0$ and $\Psi\bar{j}_3=-1$. By Theorem 4.1, $\bar{j}_1$ and $\bar{j}_3$ are not pseudo-connected. In Figure

4.1 (a) and (b), hyperplanes $\Pi\bar{j}(\text{mod}\,\alpha)=c_1$ and $\Psi\bar{j}=c_2$, $c_1$, $c_2 \in Z$, are drawn, respectively. All the points lying on the same hyperplane $\Pi\bar{j}(\text{mod}\,\alpha)=c_1$ belong to the same block of the $\alpha$-partition and all the points lying on the same hyperplane $\Psi\bar{j}=c_2$ belong to the same block of the $\Psi$-partition. Figure 4.1 shows the $\alpha$-partition, $\Psi$-partition and $\alpha\Psi$-partition pictorially. Let s=3, then $P_\alpha=\{J_0, J_1\}$ where $J_0=\{[0,0]^T, [0,1]^T, [0,2]^T, [0,3]^T, [2,0]^T, [2,1]^T, [2,2]^T, [2,3]^T\}$ and $J_1=\{[1,0]^T, [1,1]^T, [1,2]^T, [1,3]^T, [3,0]^T, [3,1]^T, [3,2]^T, [3,3]^T\}$. Also $P_\Psi=\{J_{[-3]}, ..., J_{[0]}, ..., J_{[3]}\}$ where $J_{[3]}=\{[3,0]^T\}$, $J_{[2]}=\{[2,0]^T, [3,1]^T\}$, $J_{[1]}=\{[1,0]^T, [2,1]^T, [3,2]^T\}$, $J_{[0]}=\{[0,0]^T, [1,1]^T, [2,2]^T, [3,3]^T\}$, $J_{[-1]}=\{[0,1]^T, [1,2]^T, [2,3]^T\}$, $J_{[-2]}=\{[0,2]^T, [1,3]^T\}$ and $J_{[-3]}=\{[0,3]^T\}$. $P_\alpha\Psi$ can be obtained by intersecting $J_i \cap J_{[j]}$, i=0, 1 and j=-3, ..., 3. Table 4.1 lists all blocks of $\alpha\Psi$-partition and their index points. $|P_{\alpha\Psi}| = 12 \leq \alpha\,|P_\Psi|=14$. Clearly, $P_\Psi$, $P_\alpha$ and $P_{\alpha\Psi}$ are pseudo-independent partitions. In Section 5, it is shown that the $\alpha\Psi$-partition is also the maximal pseudo-independent partition. End of example.

By Theorem 4.1, if for any arbitrary value of $a \in Z$, $0 \leq a < \alpha$ there is at least one point $\bar{j} \in J$ such that $\Pi\bar{j}(\text{mod}\,\alpha)=a$, then there are at least $\alpha$ points in J that are not pseudo-connected to each other and $J_i \in P_\alpha$ is such that $J_i \neq \emptyset$, i=0, ..., $\alpha-1$. Therefore, $|P_{\max}| \geq \alpha$. Intuitively, if J is large enough and dense, then for any arbitrary value of a, $0 \leq a < \alpha$ and $a \in Z$, there usually exists at least one index point $\bar{j}$ such that $\Pi\bar{j}(\text{mod}\,\alpha)=a$. Therefore, it is reasonable to make the following assumption:

**Assumption 4.1 (Index set):** For an algorithm (J, D) under consideration in this report, let $\Pi$ be a partitioning vector and $\alpha$ be the algorithm coefficient. It is assumed that for any arbitrary value of $a \in Z$, $0 \leq a < \alpha$, there is at least one point $\bar{j} \in J$ such that $\Pi\bar{j}(\text{mod}\,\alpha)=a$.

**Corollary 4.2:** Let $\alpha$ and $P_\alpha$ be the algorithm coefficient and the $\alpha$-partition, respectively. Then $|P_\alpha|=\alpha$ under Assumption 4.1.

The next theorem shows that this is true if the index set J is defined by (2.2), i.e., $J=N^n$. Therefore, $|P_{\max}| \geq \alpha$ if J is semi-infinite.

**Theorem 4.2** : Let $\Pi$ be a partitioning vector of (J, D) where J is defined by (2.2) and $\alpha$ be the algorithm coefficient. Then for any arbitrary value of $a \in Z$, $0 \leq a < \alpha$, there exists at least one index point $\bar{j} \in J$ such that $\Pi\bar{j}(\text{mod}\,\alpha)=a$ and the pseudo-partitionability of (J, D) is greater than or equal to $\alpha$, i.e., $|P_{\max}| \geq \alpha$.

Proof: Provided in Appendix 2.

## 5. PARTITIONING VECTOR APPROACH--METHOD

In this section, Theorem 4.1 and other results and concepts introduced in Section 4 are used to prescribe a partitioning procedure. Afterwards, Section 5.1 discusses how to find the partitioning vectors required by the procedure. Then Section 5.2 characterizes algorithms for which the method yields the optimal partition and derives lower and upper bounds on the pseudo-partitionability of arbitrary uniform dependence algorithms. The independent partitioning procedure is as follows:

**Procedure 5.1 (Finding $\alpha\Psi$-partition for algorithm (J, D) by partitioning vector approach):**
Input: Algorithm (J, D).
Output: $\alpha\Psi$-partition $P_{\alpha\Psi}$ for algorithm (J, D).
Step 1: Select m′ linearly independent dependence vectors $\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}$, set $D_c=[\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}]$, find $T \in Z^{m' \cdot n}$ such that $\text{rank}(TD_c)=m'$ and compute the corresponding partitioning vector $\Pi$ according to Theorem 5.2 provided in Section 5.1. If $\text{disp}\Pi \neq |\det(TD_c)|$, then select another set of m′ linearly independent dependence vectors and compute the corresponding partitioning vector until all distinct sets of m′ linearly independent dependence vectors are considered. If a partitioning vector $\Pi$ such that $\text{disp}\Pi = |\det(TD_c)|$ is not found, then select the partitioning vector $\Pi$ such that $\dfrac{|\det(TD_c)|}{\text{disp}\Pi}$ is minimum. Then compute the algorithm coefficient $\alpha$ according to Definition 4.1.

Step 2: Obtain $n-m'$ linearly independent separating vectors $\Psi_1, ..., \Psi_{n-m'}$ by solving equation $\Psi_i D = \bar{0}$. Set $\Psi = \begin{bmatrix} \Psi_1 \\ ... \\ \Psi_{n-m'} \end{bmatrix}$

Step 3: For every index point $\bar{j} \in J$, if $\begin{bmatrix} \Pi \bar{j} (\text{mod} \alpha) \\ \Psi \bar{j} \end{bmatrix} = \bar{y}_i = \begin{bmatrix} y_{0i} \\ y_{1i} \\ ... \\ y_{(n-m')i} \end{bmatrix}$, then assign $\bar{j}$ to $J_{\bar{y}_i}$, the block indexed by $\bar{y}_i$, i.e., $\bar{j} \in J_{\bar{y}_i}$.

Step 4: $P_{\alpha\Psi} = \{J_{\bar{y}_1}, ..., J_{\bar{y}_k}\}$.

## 5.1. Finding a partitioning vector

This subsection provides in Theorem 5.2 a closed form expression for the computation of partitioning vector $\Pi$, as required in Step 1 of the partitioning Procedure 5.1. In addition, because of the simple and regular mappings that result from equal partitioning vectors, necessary and sufficient conditions are provided in Theorem 5.1 and Corollary 5.1 for the existence of this type of vectors for a given algorithm.

**Theorem 5.1** : An algorithm $(J, \bar{D})$ has an equal partitioning vector if and only if there exists a set of $m'$ linearly independent vectors $\bar{d}_{t_1}, \bar{d}_{t_2}, ..., \bar{d}_{t_{m'}}$ such that

$$D = [\bar{d}_{t_1}, \bar{d}_{t_2}, ..., \bar{d}_{t_{m'}}] \begin{bmatrix} a_{11} & a_{12} & ... & a_{1m} \\ a_{21} & a_{22} & ... & a_{2m} \\ ... & ... & ... & ... \\ a_{m'1} & a_{m'2} & ... & a_{m'm} \end{bmatrix}, \quad a_{ij} \in \mathbb{R} \tag{5.1}$$

where $\sum_{i=1}^{m'} a_{ij}$ is an integer, $j=1, ..., m$.

Proof: Provided in Appendix 2.

It is not easy to test whether a given algorithm has an equal partitioning vector using the condition in Theorem 5.1. The following corollary provides sufficient conditions which are easier to test.

**Corollary 5.1:** An algorithm (J, D) has an equal partitioning vector if it satisfies one of the following conditions:

(1) $\text{rank}([\bar{d}_1 - \bar{d}_2, \bar{d}_1 - \bar{d}_3, ..., \bar{d}_1 - \bar{d}_m]) < \text{rank}(D)$.

(2) There exists a set of $m'$ linearly independent dependence vectors $\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}$, such that all dependence vectors can be expressed as an integer linear combination of $\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}$, i.e., $\bar{d}_j = \sum_{i=1}^{m'} a_{ij} \bar{d}_{t_i}$, $j=1, ..., m$, where $a_{ij}$, $i=1, ..., m'$ and $j=1, ..., m$, are integer constants.

Proof: Provided in Appendix 2.

If algorithm $(J, \bar{D})$ satisfies condition 1 in Corollary 5.1, then it has an equal partitioning vector $\Pi$ such that $\Pi \bar{d}_1 = ... = \Pi \bar{d}_m = \text{disp}\Pi$. To see if a given algorithm satisfies condition 2 in Corollary 5.1, one has to see if Equation 5.1 has an integer solution. This can be achieved by applying the necessary and sufficient conditions for a linear system of equations to have an integer solution provided in Appendix 1.

Given $m'$ linearly independent vectors $\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}$, the corresponding partition vector $\Pi$ belongs to the solution space of Equation 4.2. In [For83], a closed form expression for a partitioning vector which is determined by $\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}$ is given. This result is restated as Theorem 5.2 as follows.

**Theorem 5.2** [For83] : Let $\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}$ be linearly independent, consider matrix $D_c = \left[\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}\right]$ and let $T \in Z^{m' \cdot n}$ be such that $\text{rank}(TD_c) = m'$. Then $\Pi = \beta \bar{1} (TD_c)^{-1} T$ is a partitioning vector determined by

$\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}$ and disp$\Pi=\beta$, where $\beta \in N^+$ is such that $\Pi \in Z^{1\times n}$ and the greatest common divisor of the n components of $\Pi$ is equal to one.

Notice that matrix $T \in Z^{m'\times n}$ such that rank$(TD_c)=m'$ always exists. Because rank$(D_c) = m'$, there are m' linearly independent rows in $D_c$. Suppose rows $r_1, ..., r_{m'}$ are linearly independent. If $T = \begin{bmatrix} E_{r_1} \\ ... \\ E_{r_{m'}} \end{bmatrix}$ where $E_{r_1}, ..., E_{r_{m'}}$ are as defined in the beginning of Section 2, then rank$(TD_c) = m'$. In other words, the result of multiplying $D_c$ by T is a square submatrix of D that contains exactly m' linearly independent rows of the m' linearly independent columns of D. If m'=n, then T=I, the identity matrix, and $\Pi = \beta \bar{1} D_c^{-1}$. The essence of the proof is as follows [For83]. Because $\beta \bar{1}(TD_c)^{-1} TD_c=\beta \bar{1}$, vector $\beta \bar{1}(TD_c)^{-1}T$ satisfies Equation 4.1 and meets conditions 1 and 2 in Definition 4.1 by the meaning of the constant $\beta$; so $\Pi=\beta \bar{1}(TD_c)^{-1}T$ is a partitioning vector determined by $\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}$ and disp$\Pi=\beta>0$.

## 5.2. Sufficient conditions for optimality

Theorem 4.1 provides a necessary condition for two index points in J to be pseudo-connected. Next it is shown in Theorem 5.3 and 5.3a that when the dependence matrix D satisfies certain constraints, this condition becomes sufficient. The implication of this result is that the partition $P_{\alpha\Psi}$ obtained by Procedure 5.1 is maximal. In order to motivate and facilitate the understanding of the main results of this section, first, a special case is discussed in Theorem 5.3 where m'=n, i.e, rank(D)=n. In this case, the $\Psi$–partition is trivial, i.e., $P_\Psi=\{J\}$. Therefore, by Theorem 4.1, the necessary condition for two index points $\bar{j}_1, \bar{j}_2 \in J$ to be pseudo-connected is $\Pi\bar{j}_1(\text{mod}\,\alpha)=\Pi\bar{j}_2(\text{mod}\,\alpha)$.

**Theorem 5.3** : Let m'=n, $\Pi$ be a partitioning vector of algorithm (J, D) determined by $\bar{d}_{t_1}, ..., \bar{d}_{t_n}$, $D_c=[\bar{d}_{t_1}, ..., \bar{d}_{t_n}]$ and $\alpha$ be the algorithm coefficient. If $|\det D_c|=$disp$\Pi$, then
(1) two index points $\bar{j}_1, \bar{j}_2 \in J$ are pseudo-connected if and only if $\Pi\bar{j}_1(\text{mod}\,\alpha)=\Pi\bar{j}_2(\text{mod}\,\alpha)$;
(2) the $\alpha$-partition is the maximal pseudo-independent partition of (J, D), i.e., $P_{\max} = P_\alpha$, and $|P_{\max}|=\alpha$.

Proof: Provided in Appendix 2.

In this case, Procedure 5.1 becomes very simple. Since rank(D)=n, there is only one trivial separating vector $\bar{0}$ and therefore, $\Psi$–partition=\{J\}. So Step 3 in Procedure 5.1 can be skipped. When $\Pi$ is an equal partitioning vector, then $\Pi\bar{d}_i(\text{mod disp}\Pi)=0$, i= 1, ..., m. So $\alpha=$disp$\Pi= |\det D_c|$. This fact is summarized as Corollary 5.2 as follows.

**Corollary 5.2** : Let m'=n, $\Pi$ be an equal partitioning vector of algorithm (J, D) determined by $\bar{d}_{t_1}, ..., \bar{d}_{t_n}$ and $D_c=[\bar{d}_{t_1}, ..., \bar{d}_{t_n}]$. If $|\det D_c|=$disp$\Pi$, then the pseudo-partitionability of (J, D) is equal to the absolute value of the determinant of matrix $D_c$, i.e., $|P_{\max}|=|\det D_c|$.

The meaning of Corollary 5.2 is as follows. For a class of algorithms, the number of blocks in the maximal pseudo-independent partition is equal to $|\det D_c|$, the absolute value of the determinant of a submatrix of the dependence matrix D. If the algorithm is to be executed by clusters of processors with limited inter-cluster communication capabilities then the number of clusters to be used should be directly related and perhaps equal to the cardinality of the pseudo-independent partition. In such MIMD systems, $|\det D_c|$ is a direct indication of how many clusters can be used to execute the algorithm.

To find the necessary and sufficient conditions for two points $\bar{j}_1, \bar{j}_2 \in J$ to be pseudo-connected in general case, the approach used here is as follows. First, a subalgorithm $(J_{\bar{y}}, D)$ where $J_{\bar{y}} \in P_\Psi$ is considered and the necessary and sufficient conditions for two points $\bar{j}_1, \bar{j}_2 \in J_{\bar{y}}$ to be pseudo-connected are derived. To achieve this, the algorithm $(J_{\bar{y}}, D)$ is transformed, by a linear mapping **T**, into another algorithm $(\mathbf{T}(J_{\bar{y}}), \mathbf{T}(D))$ where the dimension of the index points is m' and there are m' linearly independent dependence vectors. Then Theorem 5.3 is applied to find these necessary and sufficient conditions for algorithm $(\mathbf{T}(J_{\bar{y}}), \mathbf{T}(D))$. Then it is shown that the mapping **T** is bijective and algorithms $(J_{\bar{y}}, D)$ and

$(\mathbf{T}(J_{\bar{y}}), \mathbf{T}(D))$ are equivalent in the sense that $\bar{j}_1, \bar{j}_2 \in J_{\bar{y}}$ are pseudo-connected in algorithm $(J_{\bar{y}}, D)$ if and only if their images $\mathbf{T}(\bar{j}_1)$, $\mathbf{T}(\bar{j}_2)$ are pseudo-connected in algorithm $(\mathbf{T}(J_{\bar{y}}), \mathbf{T}(D))$. So these necessary and sufficient conditions for algorithm $(\mathbf{T}(J_{\bar{y}}), \mathbf{T}(D))$ are actually valid for algorithm $(J_{\bar{y}}, D)$.

**Theorem 5.3a** : Consider algorithm (J, D), let $\bar{d}_{i_1}, ..., \bar{d}_{i_{m'}}$ be linearly independent, $D_c=[\bar{d}_{i_1}, ..., \bar{d}_{i_{m'}}]$, $T\in Z^{m' \times n}$ be such that $\mathrm{rank}(TD_c)=m'$, $\Pi=\mathrm{disp}\Pi\bar{1}(TD_c)^{-1}T$ be the partitioning vector determined by $\bar{d}_{i_1}, ..., \bar{d}_{i_{m'}}$, $\alpha$ be the algorithm coefficient and $\Psi$ be a separating matrix. If $|\det(TD_c)|=\mathrm{disp}\Pi$, then

(1) two points $\bar{j}_1, \bar{j}_2 \in J$ are pseudo-connected if and only if $\Pi(\bar{j}_1-\bar{j}_2)=0(\mathrm{mod}\,\alpha)=0$ and $\Psi\bar{j}_1=\Psi\bar{j}_2$;

(2) the $\alpha\Psi$-partition is the maximal pseudo-independent partition of (J, D), i.e., $P_{max}=P_{\alpha\Psi}$.

(3) $|P_\Psi| \leq \prod_{i=1}^{n-m'} (x_i+1)$, where $x_i=\max\{\Psi_i(\bar{j}_1-\bar{j}_2): \bar{j}_1, \bar{j}_2 \in J\}$, $i=1, ..., n-m'$, and $\alpha \leq |P_{max}| \leq \alpha\,|P_\Psi|$.

Proof: Provided in Appendix 2.

If the cardinalities of the $\alpha$-partitions of algorithms $(J_{\bar{y}_i}, D)$, where $J_{\bar{y}_i} \in P_\Psi$, $i=1, ..., q$, are all equal to $\alpha$, then $|P_{max}|=\alpha\,|P_\Psi|$. However, for some block $J_{\bar{y}} \in P_\Psi$, the cardinality of its $\alpha$-partition might be less than $\alpha$ because for some value of $a \in Z$, $0 \leq a < \alpha$, there might not exist an index point $\bar{j} \in J_{\bar{y}}$ such that $\Pi\bar{j}(\mathrm{mod}\alpha)=a$. This phenomenon is illustrated in the following example.

**Example 5.1:** Consider the algorithm of Example 4.1 with s=3. There is only one set of determining vectors $\{\bar{d}\}$ and $D_c=D$. If $T=[-1, 2]$, then $TD_c=[2]$. According to Theorem 5.2, $\Pi=2\,\bar{1}\,[2]^{-1}[-1, 2]=[-1, 2]$ and $\mathrm{disp}\Pi=2=\det(TD_c)$. As in Example 4.1, the separating matrix $\Psi=[1, -1]$. To illustrate Theorem 5.3a (1), consider points $\bar{j}_1=[0, 0]^T$ and $\bar{j}_2=[2, 2]^T$. Because $\Pi\bar{j}_1(\mathrm{mod}\,\alpha)=\Pi\bar{j}_2(\mathrm{mod}\,\alpha)$ and $\Psi\bar{j}_1=\Psi\bar{j}_2$, by Theorem 5.3a, they are pseudo-connected. Due to the fact that $\mathrm{disp}\Pi=\det(TD_c)$, by Theorem 5.3a, $P_{\alpha\Psi}$ is the maximal pseudo-independent partition. Consider $J_{[3]}\in P_\Psi$, i.e., the block whose points $\bar{j}$ are such that $\Psi\bar{j}=3$. $J_{[3]}=\{[3, 0]^T\}$ as found in Example 3.1. There does not exist an index point $\bar{j} \in J_3$ such that $\Pi\bar{j}(\mathrm{mod}\,\alpha)=0$. This illustrates the explanation before this example. By Theorem 5.3a (3), $|P_\Psi| = x+1=7$, where $x=3-(-3)=6$ and $|P_{max}|= 12 \leq \alpha\,|P_\Psi|=14$. End of example.

The complexity of Procedure 5.1 is a linear function of the cardinality of the index set. Step 1 computes the partitioning vector. The complexity of the product of the number of memory locations and the number of the operations is bounded by $O(\binom{m}{m'}n^5)$. For Step 2 the complexity of the product of the memory locations and the number of operations is bounded by $O((n-m')n^5)$. Step 3 needs at most $O(\min\{n, n-m'+1\}n\,|J|)$ operations . The total complexity is bounded above by $O(\binom{m}{m'}n^5)+O((n-m')n^5)$ $+O(\min\{n, n-m'+1\}n\,|J|)$.

## 6. COMPARISONS OF MINIMUM DISTANCE APPROACH (MDA), PARTITIONING VECTOR APPROACH (PVA) AND SMITH NORMAL FORM APPROACH (SNFA)

In the minimum distance approach (abbreviated MDA) [PeCy87], [Pei86], an elegant idea is used which consists of using a linear mapping to transform the dependence matrix D into an upper triangular matrix denoted $D^t$ in [PeCy87]. These two dependence matrices are equivalent in the sense that each dependence vector in $D^t$, the upper triangular matrix, is a linear integer combination of the dependence vectors in D and vice versa. A set of initial points, each of which corresponds to a block in the resulting partition, is identified by $D^t$ and the cardinality of the partition is the product of the diagonal elements of $D^t$. The original program is transformed into a parallel program containing parallel statements such as "parallel do" by $D^t$. An independent partition is implicitly expressed by $D^t$ and a set of initial points.

In relation to the terminology used in this report, a clarification needs to be made regarding the ability of the MDA to find the maximal independent partition. In fact, as the next example illustrates, that method finds the maximal pseudo-independent partition instead of the maximal independent partition which is claimed by the authors of [PeCy87], [Pei86].

**Example 6.1:** Consider the algorithm of Example 2.2. In Example 2.2, the maximal independent partition of this algorithm is obtained and it has four blocks, i.e., $|P_{max}|=4$. By the MDA, the upper triangular matrix is $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. So, there is only one block in the partition obtained by the MDA, which, clearly, is not maximal. However, it is the maximal pseudo-independent partition. End of example.

Unfortunately, the MDA finds the maximal pseudo-independent partition only for a restricted class of algorithms as illustrated in the next two examples. Two possible interpretations are considered for the following definition of $D^c_{m \cdot n}$ in line 15, page 218 of [PeCy87]: " $D^c_{m \cdot n}$ contains only those linear-independent dependence cycles." In one interpretation, it is assumed that only $m' \leq n$ linearly independent vectors are taken into account and included in $D^c_{m \cdot n}$ and the remaining vectors are ignored. In the other interpretation it is assumed that all dependence vectors are included in $D^c_{m \cdot n}$. The next two examples illustrate the fact that both interpretations result in inconsistent results.

**Example 6.2** Consider algorithm (J, D) where J is semi-infinite and $D= \begin{bmatrix} 3 & 0 & 0 \\ -3 & 2 & 3 \end{bmatrix} = [\bar{d}_1 \bar{d}_2 \bar{d}_3]$. By the partitioning vector approach (abbreviated PVA), if $\bar{d}_1$, $\bar{d}_2$ are chosen as determining vectors, $D_c= \begin{bmatrix} 3 & 0 \\ -3 & 2 \end{bmatrix}$, the corresponding partitioning vector $\Pi=[5, 3]$ and the algorithm coefficient $\alpha=\gcd(\Pi\bar{d}_1(\text{mod disp}\Pi), \Pi\bar{d}_2(\text{mod disp}\Pi), \Pi\bar{d}_3(\text{mod disp}\Pi))=\gcd(0, 0, 3)=3$. Because $\text{disp}\Pi = \det D_c$, by Theorem 5.3, the $\alpha$-partition (which is equal to the $\alpha\Psi$-partition) for this algorithm is pseudo-maximal and there are three blocks in the maximal pseudo-independent partition. There are two sets of two linearly independent dependence vectors $\{\bar{d}_1, \bar{d}_2\}$ and $\{\bar{d}_1, \bar{d}_3\}$. By the MDA, if $\bar{d}_1$, $\bar{d}_2$ are included in $D^c$, i.e., $D^c= \begin{bmatrix} 3 & -3 \\ 0 & 2 \end{bmatrix}$ (note that $D^c$ in [PeCy87] is $D_c^T$ in this report), then the corresponding upper triangular matrix is $\begin{bmatrix} 3 & -3 \\ 0 & 2 \end{bmatrix}$ and the number of blocks in the maximal pseudo-independent partition is 6. If $\bar{d}_1$, $\bar{d}_3$ are chosen to be in $D^c$, the corresponding upper triangular matrix is $\begin{bmatrix} 3 & -3 \\ 0 & 3 \end{bmatrix}$ and the number of blocks in the maximal pseudo-independent partition is 9. Recall that the number of blocks in the maximal pseudo-independent partition is three. Therefore, both cases yield partitions that are not independent. So all the dependence vectors have to be taken into account to find the maximal pseudo-independent partition instead of only $m'$ linearly independent dependence vectors. End of example.

**Example 6.3** Consider an algorithm (J, D) with n dependence vectors and n−1 linearly independent dependence vectors, i.e., $D \in Z^{n \cdot n}$ and $\text{rank}(D)=n-1$. By the MDA, if all dependence vectors are included in the dependence matrix, then $D^c=D^T \in Z^{n \times n}$. The upper triangular matrix $D^t$ is square and $D^t= K \times D^c$ and all diagonal elements are positive. This implies that $\text{rank}(D^t) = n$. However, since $\text{rank}(D^c) = n-1$, $\text{rank}(D^t) \leq n-1$. This is a contradiction. End of example.

In summary, if the dependence matrix contains only linearly independent dependence vectors, then the MDA is valid only for the case where all dependence vectors are linearly independent. For the case where $m' = m = n$, the MDA generates the maximal pseudo-independent partition and for the case where $m'=m < n$, it generates an independent partition that may not be maximal. In [Pei86], an algorithm to generate initial points is presented for this case. However, its complexity and optimality are not clear. Moreover, only index sets of the form $J=\{[j_1, ..., j_n]^T\colon a_i \leq j_i \leq b_i, i=1, ..., n\}$ (not necessarily dense) are considered, otherwise, the initial points are not easy to identify.

Compared with the PVA proposed in this report, the MDA has the following disadvantages. First, in the MDA, partitions are expressed implicitly in terms of the upper triangular matrix and a set of initial points. According to [PeCy87], to find the upper triangular matrix, it is necessary to solve n integer programming problems with m variables which are NP-complete, where n, m are the number of dimensions of the index points and the number of dependence vectors, respectively. This is expensive although it is affordable when n, m are small. In the PVA, partitions are expressed explicitly in terms of the partitioning vectors and separating vectors. To obtain these vectors, the dominating computations required are to find partitioning vectors, i.e., consider at most all possible combinations of $m'$ vectors from the m dependence vectors and compute $\mathrm{disp}\Pi\,\bar{1}\,(TD_c)^{-1}T$. The complexity of the execution time of Procedure 5.1 is bounded above by $\binom{m}{m'}O(n^3)$.

Secondly, as mentioned above, in the MDA, blocks of the resulting partition are implicitly expressed in terms of the upper triangular matrix and a set of initial points. Although the serial loops in the original program can be transformed into parallel loops by the upper triangular matrix, it is costly to obtain the explicit expression of blocks of the partition and to know which block a given index point belongs to. According to the notations in [PeCy87], given an index point $X \in Z^{1\times n}$, one way to see which block it belongs to is to see if equation $X = X_{i0} + AD^t$ has an integer solution $A \in Z^{1\times n}$, where $X_{i0}$ is an initial point belonging to block i. If it has, then X belongs to block i. If it does not, then another initial point $X_{j0}$ belonging to block j, $j \neq i$, is tried until an initial point $X_{k0}$ is found such that equation $X = X_{k0} + AD^t$ has an integer solution. This can be a very computationally expensive procedure. In contrast, in the PVA proposed in this report, blocks of partitions are explicitly expressed in terms of the vectors. To see which block a given index point $\bar{j} \in Z^n$ belongs to, the computations required are to compute $\Pi\bar{j}(\mathrm{mod}\,\alpha)$ and $\Psi\bar{j}$. In addition, as it will be explained in next section, this method is more convenient for mapping algorithms into systolic arrays than the MDA. It is not clear which method is more suitable for mapping algorithms into MIMD systems.

Compared with the Smith normal form approach (abbreviated SNFA), the PVA has the following disadvantages and advantages. First, SNFA always provides the maximal pseudo-independent partitions for any uniform dependence algorithm. In contrast, the PVA provides the maximal pseudo-independent partitions only when the uniform dependence algorithm satisfies the condition of Theorem 5.3a. Secondly, for the complexity, when $m=n=m'$, the complexity of the PVA is $O(n^5)+O(n\,|J|)$ and the complexity of the SNFA is $O(n^{10})+O(n^2\,|J|)$. However, it is not true that the PVA always has better complexity than the SNFA. Thirdly, in MIMD systems, one problem is to find a time-optimal schedule such that the total execution time plus the total overhead caused by communications is minized. In this case the PVA is preferred because the partitioning vector $\Pi$ could also be used to specify a linear schedule [ShFo88]. The PVA can also be used to derive optimal multiple-systolic arrays which is discussed in next section.

One interesting observation is as follows. Look at Definition 3.2, if the first k elements of the displacement vector $\bar{s}$ are equal to 1, i.e., $s_1=\ldots=s_k=1$, $0\leq k\leq m'$, then only the last $n-k$ rows of the matrix U are needed to construct the U-partition. When $m'=n$ and $k=n-1$, then only one vector is needed to construct the U-partition as the PVA. The following theorem provides a sufficient condition when only one vector is needed to construct the U-partition.

**Theorem 6.1:** Given algorithm (J, D), let $\bar{s}$ be the displacement vector. If there exists a partitioning vector $\Pi$ such that $\mathrm{disp}\Pi = |\det(TD_c)|$, where $D_c$ is the determining matrix of $\Pi$ and $T=\begin{bmatrix} E_{r_1} \\ \ldots \\ E_{r_{m'}} \end{bmatrix}$ is such that $\mathrm{rank}(TD_c)=m'$, then $s_1=\ldots=s_{m'-1}=1$.

Proof: Provided in Appendix 2.

## 7. ISSUES OF MAPPING PARTITIONED ALGORITHMS INTO MIMD/MULTI-SYSTOLIC ARRAYS

This section discusses some issues of mapping the partitioned algorithms into MIMD/multi-systolic arrays. First, a new architecture called *multi-systolic array* is proposed and how to map the partitioned algorithms into it is discussed. Then, the mapping of the partitioned algorithms into MIMD systems are addressed. The basic idea is to assign one block in the maximal pseudo-independent partition to one processor (or cluster) such that there is no communication between processors (clusters) which dominates the overhead in MIMD systems.

The system of multi-systolic arrays consists of a set of identical systolic arrays. Processing elements (PE's) in the system can be described by index vectors $[x_1, ..., x_{n-1}, x_n]$ where $x_n$ indicates which array the indexed PE belongs to and $x_1$, ..., $x_{n-1}$ indicate its location inside that array. If the pseudo-partitionability of the algorithm to be mapped is $\alpha$, then $\alpha$ arrays are taken from the pool of the system for executing the algorithm. Each block of the partition resulted from the partitioning procedure described in Section 5 will be assinged to one array. Since the partition is independent, there is no communication between different arrays and the hardware structure is expected simple and regular.

There are many methodologies of mapping algorithms into a single systolic array [Che86], [KuS87], [LiWa85], [OKFo86], [Rao85], [Qui84], [MoFo86]. With a little modification, the transformation method [MoFo86], [FoMo85] can be used to map partitioned algorithms into multi-systolic arrays. Given an algorithm (J, D), this mapping can be specified by an integer matrix $T = \begin{bmatrix} \Pi \\ S \end{bmatrix} \in Z^{n \times n}$, $\Pi \in Z^{1 \times n}$ and $S \in Z^{n-1 \times n}$. The computation indexed by $\bar{j}$ will be executed at time step $\left\lfloor \frac{\Pi \bar{j}}{\text{disp}\Pi} \right\rfloor$ where $\Pi$ is a partitioning vector and $\text{disp}\Pi = \min\{\Pi \bar{d}_i : \bar{d}_i \in D\}$, at processor $(S\bar{j}, \Pi\bar{j} (\text{mod}\, \alpha))$ where $\alpha$ is the algorithm coefficient. Notice that if $\text{disp}\Pi = \beta\alpha$, $\beta \in N^+$, then $\beta$ computations are mapped into the same PE and the same time step. If the algorithm has an equal partitioning vector, i.e., $\beta = 1$, then each PE processes only one computation at one time step. If $\beta > 1$, then each PE has to process $\beta$ computations at one time step. There are two ways to achieve this. One way is to process these $\beta$ computations in serial in one time step. The other is to install $\beta$ computing elements inside one PE so that all $\beta$ computations can be executed in parallel in one time step. For an algorithm (J, D), the problem of finding an optimal transformation with respect to the total execution time consists of finding $\Pi \in Z^{1 \times n}$ such that it minimizes the following objective function:

$$f = \frac{\max\{\Pi(\bar{j}_1 - \bar{j}_2) : \bar{j}_1, \bar{j}_2 \in J\}}{\min\{\Pi \bar{d}_i : \bar{d}_i \in D\}} \tag{7.1}$$

Subject to $\Pi D > \text{disp}\Pi$ and $\Pi$ is a partitioning vector

The solution of formulation 7.1 can be found as follows. First, find all partitioning vectors $\Pi_1$, ..., $\Pi_l$, each of which corresponds to a combination of $m'$ linearly independent dependence vectors from the m dependence vectors. Clearly, there are at most $\binom{m}{m'}$ partitioning vectors. i.e., $l \leq \binom{m}{m'}$. Secondly, identify all these partitioning vectors $\Pi$ such that $\Pi D \geq \text{disp}\Pi\, \bar{1}$. These partitioning vectors are all the feasible solutions of formulation 7.1. Finally, find the feasible partitioning vector that results in the shortest execution time and is with a feasible space transformation S. This step can be achieved by comparing all execution times by these feasible partitioning vectors, respectively.

MIMD systems consist of a set of identical processors or a set of identical clusters of processors. Processors may be connected through shared memories (tightly coupled MIMD system) or interconnection network (loosely coupled MIMD system). In both cases, communication between processors is realized by expensive interconnection network and usually is much slower than the computation speed of the processors. So the communication between processors dominates the overhead of the system and minimizing the amount of communication between processors is a main goal in mapping of algorithms to MIMD systems. If one block of the independent partition of the algorithm is assinged to one processor, the communication between processors is zero.

If the MIMD system consists of a set of clusters of processors and each cluster consists of a set of processors, then one block of the independent partition can be assigned to one cluster and inside each cluster, computations can be scheduled to execute by an optimal linear schedule. Solutions of how to schedule computations by a linear schedule and how to find the optimal linear schedule with respect to the total execution time are provided in [FoPa84], [ShFo88].

## 8. FUTURE WORK AND CONCLUSIONS

There is still more research work needed to be done on how to make good use of the partitioning methods proposed in this report to map algorithms into MIMD/multi-systolic arrays. First, the time schedule Π obtained by the procedure described in Section 7 may not as good as the one obtained by [ShFo88] even if the same number of PE's is used. This is because only partitioning vectors are considered in the procedure as the solution space of Formulation 7.1. This space is actually a small subspace of the solution space of the linear schedule problem described in [ShFo88]. It is still open to find an optimal procedure which considers the whole solution space of the linear schedule problem and makes use of the partitioning vector to yield an optimal time schedule. Secondly, instead of finding the maximal independent partitions, it is also desired to find the maximal partitions such that the communications among blocks can be supported by that target machine. For non-partitionable algorithms (an algorithm is non-partitionable if its pseudo-partitionability is equal to 1), sometimes, it is desired to find the maximal partition such that the ratio of the communications between a block and other blocks with the cardinality of that block is minimized.

The main contribution of this report is computationally inexpensive methods for identifying independent partitions of algorithms with uniform dependencies. The resulting partitions are maximal. The partitioning methods proposed here can be applied in practice as one of the many analysis procedures used by optimizing compilers to detect and exploit concurrency in serial programs. It may be particularly useful in mapping of algorithms into multiprocessor machines where processors are organized in clusters with limited inter-cluster communication capabilities. In these systems, different clusters can process distinct blocks of a partition without inter-cluster communication overhead costs. Among others, such multiprocessors include Cedar [Kuetal86] and Cm* [HwBr84].

## 9. ACKNOWLEDGEMENTS

The authors thank S. Zak, M. O'Keefe and V. Van Dongen for their discussions on Section 3.

## REFERENCES

[Baetal79]
U. Banerjee, S.C. Chen, D.J. Kuck and R.A. Towle, "Time and Parallel Processor Bounds for FORTRAN-like Loops", IEEE Trans. on Comp., Vol. c-28, No. 9, pp.660-670, Sept. 1979.

[Che86]
M. Chen, "A Design Methodology for Synthesizing Parallel Algorithms and Architectures," Journal of Parallel and Distributed Computing, pp. 461-491, December, 1986.

[Cyt86]
R. Cytron, "Doacross: Beyond Vectorization for Multiprocessors," Proc. 1986 Int'l Conf. on Parallel Processing, 1986.

[FoMo85]
J.A.B. Fortes and D.I. Moldovan, "Parallel Detection and Transformation Techniques Useful for VLSI Algorithms," J. of Parallel and Distributed Computing, 2, pp.277-301, 1985.

[FoPa84]
J.A.B. Fortes, F. Parisi-Presicce "Optimal Linear Schedule for the Parallel Execution of Algorithms," Proc. of 1984 Int'l Conf. on Parallel Processing, 1984.

[For83]
J.A.B. Fortes, "Algorithm Transformations for Parallel Processing and VLSI Architecture Design" , Ph.D. Thesis, Dept of Elect. Eng.-Syst., University of Southern California, December 1983.

[GaPe85]
D. D. Gajski and J.-K. Peir, "Essential Issues in Multiprocessor Systems," IEEE Computer, Vol.18, No.6, pp. 9-27, June 1985.

[Gol73]
L.J. Goldstein, "Abstract Algebra: A First Course," Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1973.

[HwBr84]
K. Hwang and F.A. Briggs, "Computer Architecture and Parallel Processing," McGraw-Hill, New York, 1984.

[KaBa79]
R. Kannan and A. Bachem, "Polynomial Algorithms for Computing the Smith and Hermite Normal Forms of an Integer Matrix," SIAM J. on Computing, Vol. 8, No. 4, pp. 499-507, November 1979.

[Kaetal67]
R.M. Karp, R.E. Miller and S. Winograd, "The Organization of Computations for Uniform Recurrence Equations," JACM 14, 3, pp. 563-590, Jul. 1967.

[Kuetal84]
D.C. Kuck, A.H. Sameh, R. Cytron, A.V. Veidenbaum, C.D. Polychronopoulos, G. Lee, T. McDaniel, B.R. Leasure, C. Beckman, J.R.B. Davies and C. Kruskal, "The Effects of Program Restructuring, Algorithm Changes, and Architecture Choice on Program Performance," Proc. 1984 Int'l Conf. on Parallel Processing, pp.129-138, Aug. 1984.

[Kuetal86]
D. Kuck, E. Davidson, D. Lawrie and A.H. Sameh, "Parallel Supercomputing Today and the Cedar Approach," Science, Vol. 231, pp. 967-974, Feb. 28, 1986.

[KuS87]
S. Y. Kung, "VLSI Array Processors," Prentice-Hall, Englewood Cliffs, N.J. 1987.

[LiWa85]
G.-J. Li and B. W. Wah, "The Design of Optimal Systolic Arrays," IEEE Trans. Computers, Vol. C-34,pp. 66-77, Jan. 1985.

[MoFo86]
D.I. Moldovan and J.A.B. Fortes, "Partitioning and Mapping Algorithms into Fixed Size Systolic Arrays," IEEE Trans. Computers, Vol. C-35, No. 1, pp. 1-12, Jan. 1986.

[Mor69]
L.J. Mordell, *Diophantine Equations*, Academic Press, New York, pp. 30, 1969.

[New72]
M. Newman, "Integral Matrices," Academic Press, New York, 1972.

[OKFo86]
M.T. O'Keefe and J.A.B. Fortes, "A Comparative Study of Two Systematic Design Methodologies For Systolic Arrays," Proc. 1986 Int'l Conf. on Parallel Processing.

[Pad79]
D. A. Padua, "Multiprocessors: Discussion of Theoretical and Practical Problems," Ph.D Thesis, Univ. of Illinois at Urb.-Champ., Rept. No. UIUCDCS-R-79-990, Nov. 1979.

[Paetal80]
D.A. Padua, D.J. Kuck and D.L. Lawrie, "High Speed Multiprocessor and Compilation Techniques," IEEE Trans. on Computers, Vol. C-29, No. 9, pp. 763-776, Sept. 1980.

[PeCy87]
J.-K. Peir and R. Cytron, "Minimum Distance: A Method for Partitioning Recurrences for Multiprocessors," Proc. 1987 Int'l Conf. on Parallel Processing, pp.217-225, 1987.

[PeGa86]
J.-K. Peir and D.D. Gajski, "CAMP: A Programming Aid for Multiprocessors," Proc. 1986 Int'l Conf. on Parallel Processing, pp.475-482, Aug. 1986.

[Pei86]J.-K. Peir, "Program Partitioning and Synchronization on Multiprocessor Systems," Ph.D Thesis, Report No. UIUCDCS-R-86-1259, Dept. of Computer Science, Univ. of Illinois at Urb.-Champ., Urbana, Illinois, Mar. 1986.

[Poetal86]
C.D. Polychronopoulos, D.J. Kuck and D.A. Padua, "Execution of Parallel Loops on Parallel Processor Systems," Proc. 1986 Int'l Conf. on Parallel Processing , pp. 519-527,Aug. 1986.

[Qui84]
P. Quinton, "Automatic Synthesis of Systolic Arrays from uniform Recurrent Equations," Proc. 11'th Annual Symposium on Computer Architecture, pp. 208-214, 1984.

[Rao85]
Rao, Sailesh K., "Regular Iterative Algorithms and Their Implementations on Processor Arrays," Ph. D Dissertation, Stanford University, Stanford, California, Oct. 1985.

[Sch86]
A. Schrijver, "Theory of Linear and Integer Programming," John Wiley & Sons, 1986.

[ShFo88]
W. Shang and J.A.B. Fortes, "Time Optimal Linear Schedules for Algorithms with Uniform Dependencies," Proceedings of Int'l Conf. on Systolic Arrays, May 1988, pp. 393-402.

[Str80]Strang, G. "Linear Algebra and its Applications", Second Edition, Academic Press, 1980.

[VeFr21]
O. Veblen and P. Franklin, "On Matrices Whose Elements are Integers," Annals of Mathematics 23 (1921-2), pp. 1-15.

[Wol82]
M.J. Wolfe, "Optimizing Supercompilers for Supercomputers," Ph.D thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois. Report No. UIUCDCS-R-82-1105, 1982

## APPENDIX 1. NECESSARY AND SUFFICIENT CONDITIONS FOR LINEAR SYSTEMS TO HAVE INTEGER SOLUTIONS

Consider a linear system of equations as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$$

or

$$A\bar{x} = \bar{b} \qquad \text{(a1.1)}$$

where n, m $\in N^+$, $a_{ij}$, $b_i \in Q$, i=1, ..., n and j=1, ..., m, are given coefficients and $\bar{x} \in \mathbb{R}^m$ is the unknown vector. There have been some necessary and sufficient conditions of existence of integral solutions of Equation a1.1 [Sch86, pp. 51], [VeFr21]. This section provides a new such conditions based on the result of the partitioning vector method and also a procedure to identify one integer solution of Equation a1.1.

Without loss of generality, it is always assumed that $a_{ij}$, $b_i \in Z$, i=1, ..., n and j=1, ..., m. If there is a coefficient $a_{ij}$ such that $a_{ij} = \frac{a'_{ij}}{c} \in Q$, where $a'_{ij}$, $c \in Z$ and $\gcd(a'_{ij}, c)=1$, then by multiplying both sides of the ith equation $a_{i1}x_i+\dots+a_{im}x_m=b_i$ with constant c, the assumption can be satisfied. In addition, $\gcd(a_{i1}, a_{i2}, \dots, a_{im}, b_i)$ is assumed to be 1, i=1, ..., n. These assumptions are summarized as follows.

**Assumption a1.1 (Linear system):** The linear system (a1.1) satisfies the following conditions:
(1) The coefficients of Equation a1.1 are integers, i.e., $a_{ij}$, $b_i \in Z$, i=1, ..., n and j=1, ..., m; and
(2) $\gcd(a_{i1}, a_{i2}, \dots, a_{im}, b_i) = 1$, i=1, ..., n.

Consider the algorithm (J, A), where $J=Z^n$ and the dependence vectors are $\bar{a}_1, \dots, \bar{a}_m$. Let rank(A)=m′, $\Pi$ and $\Psi$ be a partitioning vector and separating matrix of (J, A), respectively, $\bar{a}_{t_1}, \dots, \bar{a}_{t_{m'}}$ be the determining vectors of $\Pi$, $A_c=[\bar{a}_{t_1}, \dots, \bar{a}_{t_{m'}}]$ and $\alpha$ be the algorithm coefficient for (J, A). Without loss of generality, let $A_c=[\bar{a}_1, \dots, \bar{a}_{m'}]$, i.e., the first m′ columns of matrix A are linearly independent and the determining vectors of $\Pi$, and $A=[A_c, A_1]$. Let $T=\begin{bmatrix} E_{r_1} \\ \dots \\ E_{r_{m'}} \end{bmatrix}$ be such that $\text{rank}(TA_c)=m'$. As explained in Section 5.1, such a matrix T is always possible. By Theorem 5.2, $\Pi=\text{disp}\Pi\,\bar{1}\,(TA_c)^{-1}T$.

**Theorem a1.1 (Necessary conditions):** If Equation a1.1 has an integer solution, then $\Pi\bar{b}(\text{mod}\,\alpha)=0$ and $\Psi\bar{b}=\bar{0}$.

Proof: Let $\bar{j}_1=\bar{b}$ and $\bar{j}_2=\bar{0}$. Then $\bar{j}_1, \bar{j}_2 \in J=Z^n$ and $(\bar{j}_1-\bar{j}_2)=\bar{b}$. Since Equation a1.1 has an integer solution $\bar{\lambda} \in Z^m$, i.e., $A\bar{\lambda}=\bar{b}=(\bar{j}_1-\bar{j}_2)$, index points $\bar{j}_1$, $\bar{j}_2$ are pseudo-connected in algorithm (J, A). By Theorem 4.1, $\Pi(\bar{j}_1-\bar{j}_2)(\text{mod}\alpha)=0$ and $\Psi(\bar{j}_1-\bar{j}_2)=\bar{0}$. Therefore, $\Pi\bar{b}(\text{mod}\,\alpha)=0$ and $\Psi\bar{b}=\bar{0}$. □

Theorem a1.1 provides necessary conditions for Equation a1.1 to have an integer solution. The following theorem identifies the cases where the conditions in Theorem a1.1 become sufficient.

**Theorem a1.2 (Necessary and sufficient conditions):** Let $\text{disp}\Pi=|\det(TA_c)|$. Then, Equation a1.1 has an integer solution if and only if $\Pi\bar{b}(\text{mod}\,\alpha)=0$ and $\Psi\bar{b}=\bar{0}$.

Proof: (=>) See Theorem a1.1.

(<=) Now, let's prove that if $\text{disp}\Pi=|\det(TA_c)|$, $\Pi\bar{b}(\text{mod}\,\alpha)=0$ and $\Psi\bar{b}=\bar{0}$, then Equation a1.1 has an integer solution. Let $\bar{j}_1=\bar{b}$ and $\bar{j}_2=\bar{0}$. Then, $\bar{b}=(\bar{j}_1-\bar{j}_2)$, $\Pi(\bar{j}_1-\bar{j}_2)(\text{mod}\,\alpha)=\Pi\bar{b}(\text{mod}\,\alpha)=0$ and $\Psi(\bar{j}_1-\bar{j}_2)=\Psi\bar{b}=\bar{0}$. By Theorem 5.3a, $\bar{j}_1$ and $\bar{j}_2$ are pseudo-connected since $\text{disp}\Pi=|\det(TA_c)|$, i.e., there exists a vector $\bar{\lambda} \in Z^m$ such that $(\bar{j}_1-\bar{j}_2)=\bar{b}=A\bar{\lambda}$. This implies that Equation a1.1 has an integer solution

$\bar{\lambda}$. □

The procedure of how to used Theorems a1.1 and a1.2 to test the existence of integer solutions of a given linear system of equations is as follows. First, find the n−m′ separating vectors $\Psi_i$, i=1, ..., n−m′ by solving equation $\Psi_i A=\bar{0}$ and try to find a partitioning vector $\Pi$ such that disp$\Pi$= |det(TA$_c$)|. Secondly, compute the algorithm coefficient $\alpha$ and $\begin{bmatrix} \Pi\bar{b}(\text{mod}\,\alpha) \\ \Psi\bar{b} \end{bmatrix}=\bar{y}$. If $\bar{y}\neq\bar{0}$, then there does not exist an integer solution for the linear system of equations. If there exists a partitioning vector $\Pi$ such that disp$\Pi$= |det(TA$_c$)| and $\bar{y}=\bar{0}$, then there exists at least one integer solution of the linear system of equations. If such $\Pi$ does not exist and $\bar{y}=\bar{0}$, then the procedure fails to answer the question of existence of integer solutions of the linear system of equations.

If one integer solution $\bar{\delta}$ of Equation a1.1 can be identified, then all integer solutions of Equation a1.1 can be expressed as linear combinations of $\bar{\delta}$ and the n−m′ linearly independent solutions of equation $A\bar{x}=0$. The following theorem describes one integer solution of Equation a1.1 for the case where the algorithm (J, A) has a partitioning vector $\Pi$ such that disp$\Pi$= | det(TA$_c$) |.

Consider the following equation

$$\beta\,\text{disp}\Pi + \beta_1 \Pi\,\bar{a}_1 + \cdots + \beta_m \Pi\,\bar{a}_m = \alpha \qquad \text{(a1.2)}$$

Let $\Pi\bar{a}_i$ (mod disp$\Pi$)=$\alpha_i$, i=1, ..., m. Since $\alpha$=gcd(disp$\Pi$, $\alpha_1$,...,$\alpha_m$)=$\alpha$, by Lemma 1 in Appendix 2, gcd(disp$\Pi$, $\Pi\bar{a}_1$,..., $\Pi\bar{a}_m$)=$\alpha$. By [Mor69], Equation a1.2 has at least one integer solution.

**Theorem a1.3:** Let disp$\Pi$= | (det(TA$_c$) |, $\bar{\beta}'=[\beta,\beta_1,...,\beta_m]^T$ be an integer solution of Equation a1.2, $\bar{\beta}$= $[(\beta+\beta_1), \beta_2, ..., \beta_m]^T$ and $\bar{v}=A\,\bar{\beta}$. If $\Pi\bar{b}$(mod$\alpha$)=0= $\gamma_1$disp$\Pi$ + $\gamma_2\alpha$, $0\leq\gamma_2\alpha<$ disp$\Pi$ and $\gamma_1$, $\gamma_2 \in$ Z, and $\Psi_i\bar{b}=0$, i=1, ..., n−m′, then an integer solution of Equation a1.1 is $\bar{\delta}=\gamma_2\bar{\beta}$ + $\bar{\lambda}$, where $\bar{\lambda}=[\lambda_1,...,\lambda_{m'}, 0,...,0]^T$ is such that

$$(TA_c)^*T(\bar{b}-\gamma_2\bar{v}) = \det(TA_c)\begin{bmatrix}\lambda_1\\ \cdots \\ \lambda_{m'}\end{bmatrix} \qquad \text{(a1.3)}$$

where (TA$_c$)$^*$ is the adjoint matrix of matrix (TA$_c$) which is defined in Appendix 2.

Proof: Since $\bar{\beta}'$ is an integer solution of Equation a1.2, then

$$\Pi\,\bar{v} = \Pi A\bar{\beta} = \beta\Pi\bar{a}_i + \beta_1\Pi\,\bar{a}_1 + \cdots + \beta_m\Pi\,\bar{a}_m$$
$$= \beta\,\text{disp}\Pi + \beta_1\Pi\,\bar{a}_1 + \cdots + \beta_m\Pi\,\bar{a}_m = \alpha,$$

where $\bar{a}_i \in A_c$ is a determining vector of $\Pi$ and $\Pi\bar{a}_i$ = disp$\Pi$. $\Pi(\bar{b}-\gamma_2\bar{v})$=$\Pi\bar{b}-\gamma_2\Pi\bar{v}$ = $\gamma_1$disp$\Pi$ + $\gamma_2\alpha-\gamma_2\alpha$. So, $\Pi(\bar{b}-\gamma_2\bar{v})$(mod disp$\Pi$)=0 and $\Psi_i(\bar{b}-\gamma_2\bar{v})$ = $\Psi_i\bar{b}$ − $\gamma_2\Psi_i\bar{v}$= $0-\gamma_2\Psi_iA\bar{\beta}$=0 since $\Psi_iA=0$, i=1, ..., n−m′. So, by Theorem 5.3a, $\bar{b}$, $\gamma_2\bar{v} \in$ J are pseudo-connected and belong to the same block of the $\Psi$-partition. Let $\bar{b}$, $\gamma_2\bar{v}$ belong to $J_{\bar{y}} \in P_\Psi$, $L_{\bar{y}}$= {T$\bar{j}_l$: $\bar{j}_l \in J_{\bar{y}}$} and $\Delta$=TA. Consider the algorithm ($L_{\bar{y}}$, $\Delta$), let $\Gamma \in Z^{1\times m'}$ be the partitioning vector for algorithm ($L_{\bar{y}}$, $\Delta$) determined by T$\bar{a}_1$, ..., T$\bar{a}_{m'}$. Then $\Gamma$= a$\bar{1}$(TA$_c$)$^{-1}$ where a$\in N^+$ is such that greatest common divisor of the m′ components of $\Gamma$ is equal to unity. Notice that $\Pi$=$\Gamma$T is a partitioning vector determined by $\bar{a}_1$, ..., $\bar{a}_{m'}$ and disp$\Pi$ = disp$\Gamma$ = a. Since $\Pi(\bar{b}-\gamma_2\bar{v})$(mod disp$\Pi$)=0, $\Gamma$(T$\bar{b}-\gamma_2$T$\bar{v}$)(mod disp$\Gamma$)=0. By Lemma 3 in Appendix 2,

$(TA_c)^*\,(T\bar{b}-\gamma_2T\bar{v}) = \det(TA_c)\begin{bmatrix}\lambda_1\\ \cdots \\ \lambda_{m'}\end{bmatrix}$ where $\lambda_1$, ..., $\lambda_{m'} \in$ Z. So,

$$T(\bar{b}-\gamma_2\bar{v}) = (TA_c^*)^{-1}\det(TA_c)\begin{bmatrix}\lambda_1\\ \cdots \\ \lambda_{m'}\end{bmatrix}$$

$$= TA_c\begin{bmatrix}\lambda_1\\ \cdots \\ \lambda_{m'}\end{bmatrix} = T[A_c\,A_l]\begin{bmatrix}\lambda_1\\ \cdots \\ \lambda_{m'}\\ 0\\ \cdots \\ 0\end{bmatrix} = TA\bar{\lambda}$$

In summary, $T(\bar{b}-\gamma_2\bar{v})=T[A_c\,A_1]\bar{\lambda}$. Since $\Psi_i(\bar{b}-\gamma_2\bar{v})=0$, $i=1, \ldots, n-m'$, $(\bar{b}-\gamma_2\bar{v})$ belongs to $sp\{\bar{a}_1, \ldots, \bar{a}_m\}$. By Lemma 2 in Appendix 2, $\bar{b}-\gamma_2\bar{v}=A\lambda$ or $\bar{b}=\gamma_2\bar{v}+A\bar{\lambda}=\gamma_2 A\bar{\beta}+A\bar{\lambda}=A(\gamma_2\bar{\beta}+\bar{\lambda})=A\bar{\delta}$. So $\bar{\delta}$ satisfies Equation a1.1. This implies that $\bar{\delta}$ is an integer solution of Equation a1.1. □

The next procedure describes how to find an integer solution by Theorem a1.3 when the given linear system satisfies the conditions in Theorem a1.2.

**Procedure a1.2 (Finding an integer solution for Equation a1.1):**
Step 1: Find a transforming matrix T, the partitioning vector $\Pi$ and the algorithm coefficient $\alpha$.
Step 2: Write $\Pi\,\bar{b}$ as $\Pi\,\bar{b}=\gamma_1\,disp\Pi+\gamma_2\alpha$, $0\le\gamma_2\alpha<disp\Pi$.
Step 3: Solve Equation a1.2 and obtain $\bar{\beta}$.
Step 4: Compute $\bar{v}=A\,\bar{\beta}$ and find $(TA_c)^*\,T(\bar{b}-\gamma_2\bar{v})=\bar{b}'=[b'_1, \ldots, b'_{m'}]^T$.
Step 5: By Theorem a1.3, each component of $\bar{b}'$ is a multiple of $\det(TA_c)$. Set $\lambda_i=\frac{b'_i}{\det(TA_c)}$, $i=1, \ldots, m'$.
Step 6: $\bar{\delta}=\gamma_2\bar{\beta}+\bar{\lambda}$ where $\bar{\lambda}=[\lambda_1, \ldots, \lambda_{m'}, 0, \ldots, 0]^T$.

**Example a1.1:** Consider the linear system:

$$\begin{bmatrix} 2 & 1 & 3 \\ 0 & 1 & -1 \end{bmatrix}\bar{x}=\bar{b} \tag{a1.4}$$

The corresponding algorithm (J, A) has an equal partitioning vector $\Pi=[1,1]$ and $\Psi=\bar{0}$. Let $A_c=\begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}$. $\alpha=disp\Pi=2=\det A_c$. If $\bar{b}=\begin{bmatrix} 1 \\ 3 \end{bmatrix}$, then $\Pi\,\bar{b}(\mathrm{mod}\,\alpha)=4(\mathrm{mod}\,\alpha)=0$. By Theorem a1.2, Equation a1.4 has an integer solution. Now, let's follow procedure a1.2 to find this integer solution. Step 2: $\Pi\,\bar{b}=2\bullet disp\Pi$, $\gamma_1=2$ and $\gamma_2=0$. Step 3: $\bar{\beta}=\bar{0}$. Step 4: $\bar{v}=\bar{0}$, $A_c^*=\begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix}$ and $\bar{b}'=\begin{bmatrix} -2 \\ 6 \end{bmatrix}$. Step 5: $\lambda_1=\frac{-2}{2}=-1$, $\lambda_2=\frac{6}{2}=3$ and $\lambda_3=0$. Step 6, $\bar{\delta}=\bar{\lambda}=\begin{bmatrix} -1 \\ 3 \\ 0 \end{bmatrix}$. It can be verified that $\bar{\delta}$ is an integer solution of Equation a1.4. Notice that if $A_c$ is chosen as $\begin{bmatrix} 1 & 3 \\ 1 & -1 \end{bmatrix}$, then $disp\Pi \neq |\det A_c|$ and Theorems a1.2 and a1.2 can not be applied. So, it is important to choose the right $A_c$. End of example.

## APPENDIX 2

Before the proofs are presented, some mathematical notations are introduced. These notations are based on [Str80]. Let a matrix $A \in \mathbb{R}^{n\cdot n}$ and

$$A=\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}=\begin{bmatrix} \bar{a}_1\,\bar{a}_2 & \cdots & \bar{a}_n \end{bmatrix}$$

The cofactors of A are denoted $A_{ij}$, i, j=1, ..., n. The cofactors of matrix $A_c$ are denoted by $(A_c)_{ij}$, i,j=1, ..., n. The adjugate (or adjoint) matrix of A is denoted $A^*$. Some facts [Str80] are also listed here which will be used later in some of the proofs.

**Fact 1:**

$$A^*=\begin{bmatrix} A_{11} & A_{21} & \ldots & A_{n1} \\ A_{12} & A_{22} & \ldots & A_{n2} \\ \ldots & \ldots & \ldots & \ldots \\ A_{1n} & A_{2n} & \ldots & A_{nn} \end{bmatrix}$$

**Fact 2:** $A^{-1}=\frac{A^{*}}{\det A}$.

**Lemma 1:** If $\gcd(z_1, ..., z_s)=z$, $z_i \in Z$, $i=1, ..., s$, then $\gcd(z_1, c_2 z_1 + z_2, ..., c_s z_1 + z_s)=z$, $c_i \in Z$, $i=2, ..., s$.

**Proof:** Let's prove it by contradiction. Suppose that $\gcd(z_1, c_2 z_1 + z_2, ..., c_s z_1 + z_s)=z' \neq z$, then z divides $z'$, i.e., there exists a positive integer $\beta$ such that $z'=z\beta$ and $\beta \neq 1$ (See [Gol73, pp.26]). Let $z_1=z\beta\gamma_1$, $z_i=z\gamma_i$, $i=2, ..., s$, where $\gamma_i \in (Z-\{0\})$, $i=1, ..., s$. Since $z'$ divides $z_1$, $c_i z_1 + z_i$, $i=2, ..., s$, $\beta$ divides $\beta\gamma_1$, $c_i\beta\gamma_1+\gamma_i$, $i=2, ..., s$. Let $c_i\beta\gamma_1+\gamma_i=\beta\delta_i$ $i=2, ..., s$. Then $\gamma_i=\beta\delta_i - c_i\beta\gamma_1 = \beta(\delta_i - c_i\gamma_1)$. So, $\beta$ divides $\gamma_i$, $i=2, ..., s$ and $z_1$. This means that $\gcd(z_1, z_2, ..., z_s)=z'$. So $z'=z$, i.e., $\beta=1$ which is contradict to the assumption. So $\gcd(z_1, c_2 z_1 + z_2, ..., c_s z_1 + z_s)=z$. □

**Lemma 2:** Let $A \in \mathbb{R}^{n \cdot m}$, $A=[\bar{a}_1, ..., \bar{a}_m]$, rank(A)=m', $\bar{b}=\begin{bmatrix} b_1 \\ ... \\ b_n \end{bmatrix} \in sp\{\bar{a}_1, ..., \bar{a}_m\}$ and $T \in \mathbb{R}^{m' \cdot n}$ be such that rank(TA)=m'. Then $\bar{x}$ is a solution of equation $TA\bar{x}=T\bar{b}$ if and only if it is a solution of equation $A\bar{x}=\bar{b}$.

**Proof:** (=>). Let $M=[A\ \bar{b}]=\begin{bmatrix} M_1 \\ ... \\ M_n \end{bmatrix}$. Since $\bar{b} \in sp\{\bar{a}_1, ..., \bar{a}_m\}$, rank(M)=m' and rank(TM)=m'. Let $TM=[TA\ T\bar{b}]\ =K=\begin{bmatrix} K_1 \\ ... \\ K_{m'} \end{bmatrix}$. Since $K_1$, ..., $K_{m'}$ are linear combinations of $M_1$, ..., $M_n$ and rank(K)=rank(M)=m', $sp\{M_1, ..., M_n\}= sp\{K_1, ..., K_{m'}\}$, i.e., $M_i$, i=1, ..., n, can be expressed as linear combinations of $K_1$, ..., $K_{m'}$. Let

$$M = \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1m'} \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2m'} \\ \cdots & \cdots & \cdots & \cdots \\ \gamma_{n1} & \gamma_{n2} & \cdots & \gamma_{nm'} \end{bmatrix} \begin{bmatrix} K_1 \\ \cdots \\ K_{m'} \end{bmatrix}$$

or $[A\ \bar{b}]=\Gamma K=\Gamma[TA\ T\bar{b}]$. Then $A=\Gamma TA$ and $\bar{b}=\Gamma T\bar{b}$. Let $\bar{x}$ be a solution of equation $TA\bar{x}=T\bar{b}$, then $\Gamma TA\bar{x}=\Gamma T\bar{b}$. i.e., $A\bar{x}=\bar{b}$. Therefore, $\bar{x}$ is also a solution of $A\bar{x}=\bar{b}$.

(<=). Let $\bar{x}$ be a solution for equation $A\bar{x}=\bar{b}$. Then $TA\bar{x}=T\bar{b}$ which implies $\bar{x}$ is also a solution of equation $TA\bar{x}=T\bar{b}$. □

**Proof of Theorem 4.1 :**

(1). Suppose $\bar{j}_1$ and $\bar{j}_2$ are pseudo-connected, then there exists a vector $\bar{\lambda}=[\lambda_1, ..., \lambda_m]^T \in Z^m$ such that $\bar{j}_1 + D\bar{\lambda}=\bar{j}_2$. Therefore,

$$\Pi\bar{j}_1 + \Pi D\bar{\lambda} = \Pi\bar{j}_2$$

or

$$\Pi\bar{j}_1 + \sum_{i=1}^{m}\lambda_i\Pi\bar{d}_i = \Pi\bar{j}_2$$

Let $\Pi\bar{d}_i=\alpha_i + a_i\,\mathrm{disp}\Pi$, $\alpha_i$, $a_i \in Z$, $0 \le \alpha_i < \mathrm{disp}\Pi$, i=1, ..., m. So,

$$\Pi\bar{j}_2 - \Pi\bar{j}_1 = (\sum_{i=1}^{m} a_i\lambda_i)\,\mathrm{disp}\Pi + \sum_{i=1}^{m}\alpha_i\lambda_i$$

$\sum_{i=1}^{m}\lambda_i a_i$ is an integer because $\lambda_i$ and $a_i$, i=1, ..., m, are integers. Since $\gcd(\mathrm{disp}\Pi, \alpha_1, ..., \alpha_m)=\alpha$, $\alpha_i=\alpha\gamma_i$ and $\mathrm{disp}\Pi=\alpha\gamma$, $\gamma_i$, $\gamma \in Z$, i=1, ..., m. Then,

$$\Pi \bar{j}_2 - \Pi \bar{j}_1 = \alpha \left( \gamma \sum_{i=1}^{m} a_i \lambda_i + \sum_{i=1}^{m} \gamma_i \lambda_i \right) \quad \text{and} \quad \Pi(\bar{j}_2 - \bar{j}_1)(\text{mod } \alpha) = 0$$

i.e., $\Pi \bar{j}_1(\text{mod}\alpha) = \Pi \bar{j}_2$ (mod $\alpha$). This contradicts to the assumption. So $\bar{j}_1$ and $\bar{j}_2$ are not pseudo-connected.

Consider the $\alpha$-partition $P_\alpha$. Since $\Pi \bar{j}(\text{mod } \alpha) = i$, $\bar{j} \in J_i \in P_\alpha$, $i=0, \ldots, \alpha-1$, for any two arbitrary index points $\bar{j} \in J_i$, $\bar{j}' \in J_l$, $i \neq l$, $\Pi \bar{j}(\text{mod } \alpha) \neq \Pi \bar{j}'(\text{mod } \alpha)$ and they are not pseudo-connected. By Definition 2.5, P is a pseudo-independent partition.

(2). Suppose that $\bar{j}_1$, $\bar{j}_2$ are pseudo-connected, then there exists a vector $\bar{\lambda} \in Z^m$ such that $D\bar{\lambda} = (\bar{j}_1 - \bar{j}_2)$. So, $\Psi D\bar{\lambda} = \Psi(\bar{j}_1 - \bar{j}_2)$. By Definition 4.3, $\Psi_i D = \bar{0}$, $i=1, \ldots, n-m'$ which implies that $\Psi(\bar{j}_1 - \bar{j}_2) = 0$, i.e., $\Psi \bar{j}_1 = \Psi \bar{j}_2$. This is contradict to the assumption. So, $\bar{j}_1$, $\bar{j}_2$ are not pseudo-connected. For the $\Psi$-partition $P_\Psi$, let $\bar{j}_1 \in J_{\bar{y}_i}$ and $\bar{j}_2 \in J_{\bar{y}_l}$, $\bar{y}_i \neq \bar{y}_l$, $J_{\bar{y}_i}, J_{\bar{y}_l} \in P_\Psi$. The fact that $\bar{y}_i \neq \bar{y}_l$ implies that $\Psi \bar{j}_1 \neq \Psi \bar{j}_2$. So, $\bar{j}_1$, $\bar{j}_2$ are not pseudo-connected. By Definition 2.5, $P_\Psi$ is pseudo-independent.

(3). Similarly, let $\bar{j}_1 \in J_{\bar{y}_i} \in P_{\alpha\Psi}$ and $\bar{j}_2 \in J_{\bar{y}_l} \in P_{\alpha\Psi}$, $\bar{y}_i \neq \bar{y}_l$, where $\bar{y}_i = [y_{0i}, y_{1i}, \ldots, y_{(n-m')i}]^T$ and $\bar{y}_l = [y_{0l}, y_{1l}, \ldots, y_{(n-m')l}]^T$. Since $\bar{y}_i \neq \bar{y}_l$, there exists at least one dimension $t \in \{0, 1, \ldots, n-m'\}$ such that $y_{ti} \neq y_{tl}$. If $t=0$, then $\Pi \bar{j}_1 (\text{mod } \alpha) \neq \Pi \bar{j}_2 (\text{mod } \alpha)$ and by (1) of Theorem 4.1, $\bar{j}_1$ and $\bar{j}_2$ are not pseudo-connected. If $1 \leq t \leq n-m'$, then $\Psi_t \bar{j}_1 \neq \Psi_t \bar{j}_2$ and by (2) of Theorem 4.1, $\bar{j}_1$ and $\bar{j}_2$ are not pseudo-connected. So, by Definition 2.5, $P_{\alpha\Psi}$ is pseudo-independent. □

**Proof of Theorem 4.2 :**

Let $a \in Z$ and $0 \leq a < \alpha$. If it can be shown that for any arbitrary value of a, there exists at least one index point $\bar{j} \in J$ such that $\Pi \bar{j}$ (mod $\alpha$ )=a, then by Theorem 4.1, there are at least $\alpha$ index points in J which are not connected to each other. Therefore, $J_i \neq \emptyset$, where $J_i \in P_\alpha$, $i=0, \ldots, \alpha-1$. This implies that the maximal pseudo-independent partition contains at least $\alpha$ blocks. So, $|P_{\max}| \geq \alpha$.

Let $\Pi = [\pi_1\ \pi_2 \ldots \pi_n]$. Since $\Pi$ is a non-zero vector, it has at least one non-zero component. Without loss of generality, let $\pi_1 \neq 0$. Let $M = t\,\alpha + a$ (i.e., M (mod $\alpha$)=a), $M \in Z - \{0\}$, $t \in Z$, and $M \cdot \pi_1 > 0$. Since $\gcd(\pi_1, \ldots, \pi_n) = 1$, by [Mor69] there exists at least one integer solution of the following equation

$$\pi_1 x_1 + \cdots + \pi_n x_n = M \tag{1}$$

Let $\bar{z} = \begin{bmatrix} z_1 \\ \ldots \\ z_n \end{bmatrix}$ be such an integer solution of Equation 1. If $z_1, \ldots, z_n \geq 0$, then $\bar{z} \in J$ and it has been proven that for any arbitrary integer a, $0 \leq a < \alpha$, there exists at least one index point $\bar{z} \in J$ such that $\Pi \bar{z}(\text{mod } \alpha) = M$ (mod $\alpha$ )=a. Now suppose not all $z_1, \ldots, z_n \geq 0$. It is clear that all the solutions of Equation 1 take the form

$$\bar{x} = \bar{z} + \begin{bmatrix} -\pi_2 \\ \pi_1 \\ 0 \\ 0 \\ \ldots \\ 0 \end{bmatrix} t_2 + \begin{bmatrix} -\pi_3 \\ 0 \\ \pi_1 \\ 0 \\ \ldots \\ 0 \end{bmatrix} t_3 + \begin{bmatrix} -\pi_n \\ 0 \\ 0 \\ \ldots \\ 0 \\ \pi_1 \end{bmatrix} t_n \tag{2}$$

where $t_2, t_3, \ldots, t_n$ are constants. This can be verified as follows.

$$\Pi \bar{x} = \Pi \bar{z} + \Pi \begin{bmatrix} -\pi_2 \\ \pi_1 \\ 0 \\ 0 \\ \ldots \\ 0 \end{bmatrix} t_2 + \Pi \begin{bmatrix} -\pi_3 \\ 0 \\ \pi_1 \\ 0 \\ \ldots \\ 0 \end{bmatrix} t_3 + \Pi \begin{bmatrix} -\pi_n \\ 0 \\ 0 \\ \ldots \\ 0 \\ \pi_1 \end{bmatrix} t_n$$

$$= M + (-\pi_1\pi_2 + \pi_2\pi_1)t_2 + (-\pi_1\pi_3 + \pi_3\pi_1)t_3 + \cdots + (-\pi_1\pi_n + \pi_n\pi_1)t_n = M$$

Therefore, $\bar{x}$ is a solution of Equation 1. Next, it shows how a non-negative integer solution of Equation 1 is constructed from Equation 2. Let,

$$\bar{x} = \begin{bmatrix} z_1 - \pi_2 t_2 - \pi_3 t_3 - \cdots - \pi_n t_n \\ z_2 + \pi_1 t_2 \\ z_3 + \pi_1 t_3 \\ \cdots \\ z_n + \pi_1 t_n \end{bmatrix} = \begin{bmatrix} x_1 \\ \cdots \\ x_n \end{bmatrix}$$

If $t_i \geq -\dfrac{z_i}{\pi_1}$, then $x_i \geq 0$, i=2, ..., n. Let

$$t_i = \left\lceil -\frac{z_i}{\pi_1} \right\rceil = -\frac{z_i}{\pi_1} + \beta_i \quad \text{where } 0 \leq \beta_i < 1 \quad i=2, ..., n. \tag{3}$$

Now it is shown that $x_1$ is also greater than or equal to zero if $t_i$, i=2, ..., n, are defined by (3).

$$\begin{aligned} x_1 &= z_1 - \sum_{i=2}^{n} \pi_i t_i = z_1 - \sum_{i=2}^{n} (\beta_i - \frac{z_i}{\pi_1}) \pi_i \\ &= z_1 - \sum_{i=2}^{n} \beta_i \pi_i + \sum_{i=2}^{n} \pi_i \frac{z_i}{\pi_1} \\ &= z_1 - \sum_{i=2}^{n} \beta_i \pi_i + \frac{1}{\pi_1} ( M - z_1 \pi_1 ) \\ &= \frac{M}{\pi_1} - \sum_{i=2}^{n} \beta_i \pi_i \end{aligned}$$

Notice that $\sum_{i=2}^{n} |\pi_i| > \sum_{i=2}^{n} \beta_i |\pi_i| \geq \sum_{i=2}^{n} \beta_i \pi_i$. If M is selected such that $M = t\alpha + a$ and $\dfrac{M}{\pi_1} \geq \sum_{i=2}^{n} |\pi_i|$, then $x_1 = \dfrac{M}{\pi_1} - \sum_{i=2}^{n} \beta_i \pi_i \geq \dfrac{M}{\pi_1} - \sum_{i=2}^{n} |\pi_i| \geq 0$. So, there exists an index point $\bar{x} \in J$ such that $\Pi \bar{x} (\bmod \alpha) = a$. □

**Proof of Theorem 5.1 :**

(=>). Let $\Pi$ be an equal partitioning vector of (J, D). Then by Definition 4.1, there exists a set of $m'$ linearly independent dependence vectors such that

$$\Pi \bar{d}_{t_1} = \cdots = \Pi \bar{d}_{t_{m'}} = \text{disp}\Pi > 0$$

and

$$\Pi \bar{d}_j = a_j \,\text{disp}\Pi, \quad a_j \in Z, \quad j = 1, ..., m \tag{4}$$

Since rank(D)=$m'$ and $\bar{d}_{t_1}, \bar{d}_{t_2}, ..., \bar{d}_{t_{m'}}$ are linearly independent, $\bar{d}_j$, $1 \leq j \leq m$, can be expressed as a linear combination of $\bar{d}_{t_1}, ..., \bar{d}_{t_{m'}}$. i.e.,

$$\bar{d}_j = \left[ \bar{d}_{t_1}, ..., \bar{d}_{t_{m'}} \right] \begin{bmatrix} a_{1j} \\ a_{2j} \\ \cdots \\ a_{m'j} \end{bmatrix} \quad 1 \leq j \leq m \tag{5}$$

So,

$$\begin{aligned} \Pi \bar{d}_j &= \Pi \left[ \bar{d}_{t_1}, ..., \bar{d}_{t_{m'}} \right] \begin{bmatrix} a_{1j} \\ a_{2j} \\ \cdots \\ a_{m'j} \end{bmatrix} \\ &= \text{disp}\Pi \, [1\,1 \cdots 1] \begin{bmatrix} a_{1j} \\ a_{2j} \\ \cdots \\ a_{m'j} \end{bmatrix} \end{aligned}$$

$$= \text{disp}\Pi \sum_{i=1}^{m'} a_{ij}$$

By (4), $\Pi \bar{d}_j = a_j \,\text{disp}\Pi$, where $a_j \in Z$, $1 \le j \le m$. So $\sum_{i=1}^{m'} a_{ij} = a_j$, $1 \le j \le m$, are integers.

(<=). Consider the following two systems of equations:

$$\begin{cases} \Pi(\bar{d}_{t_1} - \bar{d}_{t_2}) = 0 \\ \Pi(\bar{d}_{t_1} - \bar{d}_{t_3}) = 0 \\ \dots \\ \Pi(\bar{d}_{t_1} - \bar{d}_{t_{m'}}) = 0 \end{cases} \tag{6}$$

and

$$\begin{cases} \Pi \bar{d}_{t_1} = 0 \\ \Pi \bar{d}_{t_2} = 0 \\ \dots \\ \Pi \bar{d}_{t_{m'}} = 0 \end{cases} \tag{7}$$

Let $N_1$ be the solution space of Equation 6 and $N_2$ be the solution space of Equation 7. If $\Pi$ is a solution of Equation 7, then it is a solution of Equation 6. Thus $N_2 \subseteq N_1$. The dimension of $N_1$ is $n - m' + 1$ and the dimension of $N_2$ is $n - m'$. This implies that $N_2 \subset N_1$. So, there exists at least one solution $\Pi'$ of Equation 6 such that $\Pi' \bar{d}_{t_j} \ne 0$, $1 \le j \le m'$. Let $\Pi \in Z^{1 \times n}$ be such a solution of Equation 6 that $\Pi \bar{d}_{t_j} = a > 0$, $j = 1 \dots, m'$, and the greatest common divisor of the n components of $\Pi$ is equal to one. Then

$$\Pi \bar{d}_j = \Pi \left[ \bar{d}_{t_1}, \dots, \bar{d}_{t_{m'}} \right] \begin{bmatrix} a_{1j} \\ a_{2j} \\ \dots \\ a_{m'j} \end{bmatrix} = a \sum_{i=1}^{m'} a_{ij},\ j = 1, \dots, m.$$

Since $\sum_{i=1}^{m'} a_{ij}$, $j = 1, \dots, m$, are integers, the following holds.

$$\Pi \bar{d}_j (\text{mod } a) = 0, \quad j = 1, \dots, m \ \text{ and } \text{disp}\Pi = a > 0$$

By Definition 4.1, $\Pi$ is an equal partitioning vector of (J, D). □.

**Proof of Corollary 5.1:**

(1). Let's first prove that if algorithm (J, D) satisfies the first condition, then it has an equal partitioning vector. Consider the following two systems of equations:

$$\begin{cases} \Pi(\bar{d}_1 - \bar{d}_2) = 0 \\ \Pi(\bar{d}_1 - \bar{d}_3) = 0 \\ \dots \\ \Pi(\bar{d}_1 - \bar{d}_m) = 0 \end{cases} \tag{8}$$

and

$$\begin{cases} \Pi \bar{d}_1 = 0 \\ \Pi \bar{d}_2 = 0 \\ \dots \\ \Pi \bar{d}_m = 0 \end{cases}$$

With the similar reasoning as in the second part of the proof of Theorem 5.1, it can be shown that there exists at least one solution $\Pi'$ of Equation 8 such that $\Pi'\bar{d}_j \neq 0$, $1 \leq j \leq m$. Let $\Pi \in Z^{1\times n}$ be such a solution of Equation 8 that $\Pi\,\bar{d}_j = \mathrm{disp}\Pi \geq 0$, $j=1, \ldots, m$, and the greatest common divisor of the n components of $\Pi$ is equal to one. Then $\Pi\,\bar{d}_j = \mathrm{disp}\Pi > 0$, $j=1, \ldots, m$. Therefore, $\Pi$ is an equal partitioning vector.

(2). By the assumption of condition 2, there exists a set of $m'$ linearly independent dependence vectors $\bar{d}_{t_1}, \ldots, \bar{d}_{t_{m'}}$ such that all dependence vectors can be expressed as an integer linear combination of $\bar{d}_{t_1}, \ldots, \bar{d}_{t_{m'}}$. i.e., $\bar{d}_j = \sum_{i=1}^{m'} a_{ij}\,\bar{d}_{t_i}$, $j=1, \ldots, m$, where $a_{ij}$, $i=1, \ldots, m'$ and $j=1, \ldots, m$, are integer constants. Clearly, $\sum_{i=1}^{m'} a_{ij}$, $j=1, \ldots, m$, are integers. According to Theorem 5.1, (J, D) has an equal partitioning vector. □

**Lemma 3:** Let $m'=n$, $\Pi$ be a partitioning vector of algorithm (J, D) determined by $\bar{d}_{t_1}, \ldots, \bar{d}_{t_n}$, $D_c=[\bar{d}_{t_1}, \ldots, \bar{d}_{t_n}]$ and $\bar{j}_1, \bar{j}_2 \in J$. If $|\det D_c| = \mathrm{disp}\Pi$ and $\Pi(\bar{j}_1-\bar{j}_2)(\mathrm{mod}\ \mathrm{disp}\Pi)=0$, then $\bar{j}_1, \bar{j}_2$ are pseudo-connected and $D_c^*(\bar{j}_1-\bar{j}_2) = \det D_c \begin{bmatrix} \lambda_1 \\ \ldots \\ \lambda_n \end{bmatrix}$, where $\lambda_1, \ldots, \lambda_n \in Z$

**Proof:** Consider algorithm (J, $D_c$). The algorithm coefficient is equal to $|\det D_c|$, i.e., $\alpha = \gcd(\Pi\bar{d}_{t_1}, \ldots, \Pi\bar{d}_{t_n}) = \gcd(\mathrm{disp}\Pi, \ldots, \mathrm{disp}\Pi) = \mathrm{disp}\Pi = |\det D_c|$. By Corollary 3.1, there are $|\det D_c|$ blocks in the maximal pseudo-partition of algorithm (J, $D_c$) and by Corollary 4.2, $|P_\alpha| = \alpha = |\det D_c|$. So, the $\alpha$-partition is the maximal pseudo-independent partition. Because $\Pi(\bar{j}_1-\bar{j}_2)(\mathrm{mod}\ \mathrm{disp}\Pi)=0$, $\bar{j}_1, \bar{j}_2$ are in the some block of $\alpha$-partition. Therefore, $\bar{j}_1$ and $\bar{j}_2$ are pseudo-connected and $\bar{j}_1-\bar{j}_2 = D_c\bar{\lambda}$, $\bar{\lambda} \in Z^n$. By Fact 1 and Fact 2, $D_c^*(\bar{j}_1-\bar{j}_2) = \det D_c\bar{\lambda}$. □

**Proof of Theorem 5.3:**

(1). By Theorem 4.1, $\bar{j}_1, \bar{j}_2$ are pseudo-connected only if $\Pi\bar{j}_1(\mathrm{mod}\,\alpha) = \Pi\bar{j}_2(\mathrm{mod}\,\alpha)$. Let's prove that $\bar{j}_1, \bar{j}_2$ are pseudo-connected if $\Pi\bar{j}_1(\mathrm{mod}\,\alpha) = \Pi\bar{j}_2(\mathrm{mod}\,\alpha)$.

Let $\Pi(\bar{j}_1-\bar{j}_2) = \gamma_1\,\mathrm{disp}\Pi + \gamma_2\alpha$, $0 \leq \gamma_2\alpha < \mathrm{disp}\Pi$ and $\gamma_1, \gamma_2 \in Z$. If $\gamma_2 = 0$, by Lemma 3, $\bar{j}_1$ and $\bar{j}_2$ are pseudo-connected. Suppose $\gamma_2 \neq 0$ and let $\Pi\bar{d}_j = a_j\mathrm{disp}\Pi \pm \alpha_j$, $a_j$, $\alpha_j \in Z$, $0 \leq \alpha_j < \mathrm{disp}\Pi$, $i=1, \ldots, m$. Since $\gcd(\mathrm{disp}\Pi, \alpha_1, \ldots, \alpha_m) = \alpha$, by Lemma 1, $\gcd(\mathrm{disp}\Pi, \Pi\,\bar{d}_1, \ldots, \Pi\,\bar{d}_m) = \alpha$. By [Mor69], there exists at least one integer solution of the following equation:

$$\lambda\,\mathrm{disp}\Pi + \lambda_1\,\Pi\,\bar{d}_1 + \cdots + \lambda_m\,\Pi\,\bar{d}_m = \alpha \tag{9}$$

Let $\bar{\lambda} = [\lambda, \lambda_1, \ldots, \lambda_m]$ be an integer solution of Equation 9. Then $\gamma_2\bar{\lambda}$ is an integer solution of the following equation:

$$\gamma_2\lambda\,\mathrm{disp}\Pi + \gamma_2\lambda_1\,\Pi\,\bar{d}_1 + \cdots + \gamma_2\lambda_m\,\Pi\,\bar{d}_m = \gamma_2\alpha$$

Let

$$\bar{j} = \bar{j}_2 + \gamma_2\lambda\,\bar{d}_{t_1} + \gamma_2\lambda_1\,\bar{d}_1 + \cdots + \gamma_2\lambda_m\,\bar{d}_m$$

where $\bar{d}_{t_1} \in D_c$ is a determining vector of $\Pi$. Clearly, $\bar{j}$ and $\bar{j}_2$ are pseudo-connected. Then,

$$\Pi(\bar{j}-\bar{j}_2) = \gamma_2\lambda\,\mathrm{disp}\Pi + \gamma_2\lambda_1\,\Pi\,\bar{d}_1 + \cdots + \gamma_2\lambda_m\,\Pi\,\bar{d}_m = \gamma_2\alpha$$

$\Pi(\bar{j}_1-\bar{j}) = \Pi\bar{j}_1 - \Pi\bar{j} + \Pi\bar{j}_2 - \Pi\bar{j}_2 = \Pi(\bar{j}_1-\bar{j}_2) - \Pi(\bar{j}-\bar{j}_2) = \gamma_1\mathrm{disp}\Pi + \gamma_2\alpha - \gamma_2\alpha$. So, $\Pi(\bar{j}_1-\bar{j})(\mathrm{mod}\ \mathrm{disp}\Pi)=0$. By Lemma 3, $\bar{j}_1$ and $\bar{j}$ are pseudo-connected. Since $\bar{j}$, $\bar{j}_2$ and $\bar{j}$, $\bar{j}_1$ are pseudo-connected, respectively, $\bar{j}_1$ and $\bar{j}_2$ are also pseudo-connected. Therefore, if $\Pi(\bar{j}_1-\bar{j}_2)(\mathrm{mod}\,\alpha)=0$, then $\bar{j}_1$ and $\bar{j}_2$ are pseudo-connected.

(2). Consider the $\alpha$-partition $P_\alpha$, since $\Pi\bar{j}_1(\mathrm{mod}\,\alpha) = \Pi\bar{j}_2(\mathrm{mod}\,\alpha)$ where $\bar{j}_1, \bar{j}_2 \in J_i \in P_\alpha$ are arbitrary index points, $0 \leq i < \alpha$, by the results in (1) of Theorem 5.3, $\bar{j}_1, \bar{j}_2$ are pseudo-connected. By Definition 2.5,

$P_\alpha$ is the maximal pseudo-independent partition. i.e., $P_\alpha = P_{max}$ and $|P_{max}| = |P_\alpha| = \alpha$. □

**Lemma 4:** Let $J_{\bar y} \in P_\Psi$, then $J_{\bar y} = \{\bar j: \bar j \in J, \bar j = \bar p + D\bar x, \bar x \in \mathbb{R}^m, \Psi\bar p = \bar y\}$.

**Proof:** Let S denote $\{\bar j: \bar j \in J, \bar j = \bar p + D\bar x, \bar x \in \mathbb{R}^m, \Psi\bar p = \bar y\}$. If $\bar j \in J_{\bar y}$, then $\Psi\bar j = \bar y$ and $\Psi(\bar j - \bar p) = \bar 0$. By the *Fundamental Theorem of Linear Algebra* (See [Str80], pp. 75 and pp. 87), $\bar j - \bar p$ belongs to $sp\{\bar d_1, ..., \bar d_m\}$ because $\Psi D = 0$. This implies that there exists an $\bar x \in \mathbb{R}^m$ such that $\bar j - \bar p = D\bar x$, i.e., $\bar j = \bar p + D\bar x$. So, $\bar j \in S$ and $J_{\bar y} \subseteq S$. Now, let $\bar j \in S$, then $\bar j = \bar p + D\bar x$ and $\Psi\bar j = \Psi\bar p + \Psi D\bar x = \Psi\bar p = \bar y$. So $\bar j \in J_{\bar y}$ and $J_{\bar y} = S$. □

**Lemma 5:** Let $\bar d_{t_1}, ..., \bar d_{t_{m'}}$ be linearly independent, $D_c = [\bar d_{t_1}, ..., \bar d_{t_{m'}}]$, $T \in Z^{m' \times n}$ be such that $\text{rank}(TD_c) = m'$, $J_{\bar y} \in P_\Psi$ and $L_{\bar y} = \{T\bar j: \bar j \in J_{\bar y}\}$. Then,
(1) the mapping $\mathbf{T}: J_{\bar y} \to L_{\bar y}$, $\mathbf{T}(\bar j) = T\bar j$, $\bar j \in J_{\bar y}$ is bijective and
(2) $T\bar j_1, T\bar j_2 \in L_{\bar y}$ are pseudo-connected in algorithm $(L_{\bar y}, TD)$ if and only if $\bar j_1, \bar j_2 \in J_{\bar y}$ are pseudo-connected in algorithm $(J_{\bar y}, D)$, i.e., if $\bar j_1 - \bar j_2 = D\bar\lambda$ if and only if $T(\bar j_1 - \bar j_2) = TD\bar\lambda$.

**Proof:** (1). Consider the mapping $\mathbf{T}: J_{\bar y} \to L_{\bar y}$, $\mathbf{T}(\bar j) = T\bar j$, $\bar j \in J_{\bar y}$. Since $L_{\bar y} = \{T\bar j: \bar j \in J_{\bar y}\}$, $\mathbf{T}$ is surjective. By Lemma 4, $J_{\bar y} = \{\bar j: \bar j \in J, \bar j = \bar p + D\bar x, \bar x \in \mathbb{R}^m, \Psi\bar p = \bar y\}$. Since $\bar d_{t_1}, ..., \bar d_{t_{m'}}$ are linearly independent and rank(D)=m', each dependence vector can be written as a linear combination of $\bar d_{t_1}, ..., \bar d_{t_{m'}}$, i.e., $D = D_c \times \Lambda$ where $\Lambda \in \mathbb{R}^{m' \times m}$. So, $J_{\bar y}$ can be rewritten as $J_{\bar y} = \{\bar j: \bar j = \bar p + D_c\bar z, \bar z = \Lambda\bar x, \bar j \in J, \bar x \in \mathbb{R}^{m'}, \Psi\bar p = \bar y\}$. Let $\bar j_1, \bar j_2 \in J_{\bar y}$ and $\bar j_1 = \bar p + D_c\bar z_1$, $\bar j_2 = \bar p + D_c\bar z_2$, then, $T(\bar j_1 - \bar j_2) = TD_c(\bar z_1 - \bar z_2) = \bar 0$ if and only if $\bar z_1 = \bar z_2$, or equivalently, $\bar j_1 = \bar j_2$, since rank(TD)=m', i.e., $T\bar j_1 = T\bar j_2$ if and only if $\bar j_1 = \bar j_2$. So $\mathbf{T}$ is injective which implies $\mathbf{T}$ is bijective.

(2). (<=). If $\bar j_1, \bar j_2 \in J_{\bar y}$ are pseudo-connected, then there is a vector $\bar\lambda \in Z^m$ such that $\bar j_1 = \bar j_2 + D\bar\lambda$. So, $T\bar j_1 = T\bar j_2 + TD\bar\lambda$ and $T\bar j_1$, $T\bar j_2$ are pseudo-connected.

(=>). If $T\bar j_1$, $T\bar j_2$ are pseudo-connected, then there exists a vector $\bar\lambda \in Z^m$ such that $T(\bar j_1 - \bar j_2) = TD\bar\lambda$. $\bar j_1, \bar j_2 \in J_{\bar y}$ implies that $\bar j_1 - \bar j_2 \in sp\{\bar d_1, ..., \bar d_m\}$. By Lemma 2, $\bar\lambda$ is also a solution of equation $D\bar\lambda = (\bar j_1 - \bar j_2)$ which implies that $\bar j_1$ and $\bar j_2$ are pseudo-connected. □

**Proof of Theorem 5.3a:**
(1). (=>) See Theorem 4.1.

(<=). Since $\Psi\bar j_1 = \Psi\bar j_2$, $\bar j_1$ and $\bar j_2$ belong to the same block of the $\Psi$-partition, i.e., $\bar j_1, \bar j_2 \in J_{\bar y} \in P_\Psi$, where $\bar y = \Psi\bar j_1$. Let $L_{\bar y} = \{T\bar j: \bar j \in J_{\bar y}\}$ and $\Delta = TD$. Consider the algorithm $(L_{\bar y}, \Delta)$, let $\Gamma$ be the partitioning vector of $(L_{\bar y}, \Delta)$ determined by $T\bar d_{t_1}, ..., T\bar d_{t_{m'}}$ and $\alpha'$ be the coefficient for algorithm $(L_{\bar y}, \Delta)$. By Theorem 5.2, $\Gamma = \beta'\bar 1(TD_c)^{-1}$ and disp $\Gamma = \beta'$, where $\beta' \in N^+$ is such that the greatest common divisor of the m' components of $\Gamma$ is equal to one and $\Gamma \in Z^{1\times m'}$. Now, consider the row vector $\Pi = (1/\beta'')\Gamma T = \beta\bar 1(TD_c)^{-1}T$ where $\beta = (\beta'/\beta'')$ is such that the greatest common divisor of the n components of $\Pi$ is equal to unity and $\Pi \in Z^{1\times n}$. By Theorem 5.2, $\Pi$ is a partitioning vector determined by vectors $\bar d_1, ..., \bar d_{m'}$ for algorithm (J, D) and disp$\Pi = (1/\beta'')$disp$\Gamma$. This implies that $\alpha = (\alpha'/\beta'')$ is the algorithm coefficient for algorithm (J, D) which can be proven as follows. $\alpha' = \gcd(\beta', \Gamma T\bar d_1, ..., \Gamma T\bar d_m)$ and $\alpha = \gcd(\beta, \Pi\bar d_1, ..., \Pi\bar d_m)$ (see Definition 4.1 and Lemma 1). Now, $\alpha = \gcd(\beta'/\beta'', (1/\beta'')\Gamma T\bar d_1, ..., (1/\beta'')\Gamma T\bar d_m) = \alpha'/\beta''$.

By assumption $\Pi(\bar j_1 - \bar j_2)(\text{mod}\,\alpha) = 0$, i.e., $\Pi(\bar j_1 - \bar j_2) = \lambda\alpha$ where $\lambda \in Z$. Because $\alpha = \alpha'/\beta''$, $\Pi\bar j_1 = (1/\beta'')\Gamma T\bar j_1$ and $\Pi\bar j_2 = (1/\beta'')\Gamma T\bar j_2$, $\Pi(\bar j_1 - \bar j_2) = (1/\beta'')\Gamma(T\bar j_1 - T\bar j_2) = \lambda\alpha'/\beta''$, or $\Gamma(T\bar j_1 - T\bar j_2) = \lambda\alpha'$ which means $\Gamma(T\bar j_1 - T\bar j_2)(\text{mod}\,\alpha') = 0$. That is, if $\Pi(\bar j_1 - \bar j_2)(\text{mod}\,\alpha) = 0$, then $\Gamma(T\bar j_1 - T\bar j_2)(\text{mod}\,\alpha') = 0$. Notice that the dimension of the index points of algorithm $(L_{\bar y}, \Delta)$ is m' and the rank of its dependence matrix $\Delta$ is m'. By applying Theorem 5.3 (1), for any two arbitrary index points $T\bar j_1, T\bar j_2 \in L_{\bar y}$, if $\Gamma(T\bar j_1 - T\bar j_2)(\text{mod}\,\alpha') = 0$, then $T\bar j_1$, $T\bar j_2$ are pseudo-connected. According to Lemma 5, $\bar j_1, \bar j_2 \in J_{\bar y}$ are also pseudo-connected in algorithm $(J_{\bar y}, D)$. In summary, it has been proven that for any two arbitrary index points $\bar j_1, \bar j_2 \in J$, if $\Pi(\bar j_1 - \bar j_2)(\text{mod}\,\alpha) = 0$ and $\Psi\bar j_1 = \Psi\bar j_1$, then $\bar j_1, \bar j_2$ are pseudo-connected.

(2). Consider any two arbitrary points $\bar j_1, \bar j_2 \in J_{\bar y_i} \in P_{\alpha\Psi}$, i=1, ..., k. Since $\Pi(\bar j_1 - \bar j_2)(\text{mod}\,\alpha) = 0$ and $\Psi(\bar j_1 - \bar j_2) = \bar 0$, by the results in Theorem 5.3a (1), they are pseudo-connected. By Definition 2.5, $P_{\alpha\Psi}$ is the maximal pseudo-independent partition.

(3). Let $P_\alpha^{\bar{y}}$ denote the $\alpha$-partition for algorithm $(L_{\bar{y}}, \Delta)$. First, let's show that $|P_\alpha^{\bar{y}}| \le \alpha$. There is one complication caused by the mapping $\underline{T}$, i.e., for some value of $a \in Z$, $0 \le a < \mathrm{disp}\Gamma$, there may not exist an index point $T\bar{j} \in L_{\bar{y}}$ such that $\Gamma T\bar{j} (\mathrm{mod}\ \mathrm{disp}\Gamma) = a$ even if $J = N^n$. However, because there are at most $\alpha$ nonempty blocks in $P_\alpha^{\bar{y}}$, $|P_\alpha^{\bar{y}}| \le \alpha$.

For $P_\Psi$, clearly, there at most $\prod_{i=1}^{n-m'} (x_i+1)$ nonempty blocks, where $x_i = \max\{\Psi_i(\bar{j}_1 - \bar{j}_2): \bar{j}_1, \bar{j}_2 \in J\}$, $i=1, \ldots, n-m'$. So, $|P_\Psi| \le \prod_{i=1}^{n-m'} (x_i+1)$. $|P_\alpha^{\bar{y}}| \le \alpha$ implies that $|P_{\max}| \le \max\{|P_\alpha^{\bar{y}}|: J_{\bar{y}} \in P_\Psi\} |P_\Psi| = \alpha |P_\Psi|$. By Theorem 4.1, $P_\alpha$ is a pseudo-independent partition of (J, D) and $|P_\alpha| = \alpha$. So $\alpha \le |P_{\max}| \le \alpha |P_\Psi|$.
□

**Proof of Theorem 6.1:** Without loss of generality, let $\det(TD_c) > 0$. Because $\mathrm{disp}\underline{\Pi} = \det(TD_c)$, by Theorem 5.2, $\Pi = \mathrm{disp}\Pi\bar{1}(TD_c)^{-1}T = \det(TD_c)\bar{1}(TD_c)^{-1}T$. By Fact 2, $\Pi = \bar{1}(TD_c)^*T$. Let $\Gamma = [\gamma_1, \ldots, \gamma_{m'}] = \bar{1}(TD_c)^*$, then $\gcd(\gamma_1, \ldots, \gamma_{m'}) = \gcd(\pi_1, \ldots, \pi_n) = 1$. Let $(TD_c)_{ij}$, $i,j=1, \ldots, m'$ be the cofactors of matrix $(TD_c)$. Then by Fact 1, $\Gamma = (\sum_{j=1}^{m'} (TD_c)_{1j}, \ldots, \sum_{j=1}^{m'} (TD_c)_{m'j})$. So, $\gcd(\sum_{j=1}^{m'} (TD_c)_{1j}, \ldots, \sum_{j=1}^{m'} (TD_c)_{m'j}) = 1$ and $\gcd((TD_c)_{ij}, i,j=1, \ldots, m') = 1$. This implies that the greatest common divisor of all subdeterminants of order $m'-1$ is equal to 1, i.e., $\prod_{i=1}^{m'-1} s_i = 1$. Therefore, $s_1 = \ldots = s_{m'-1} = 1$. □

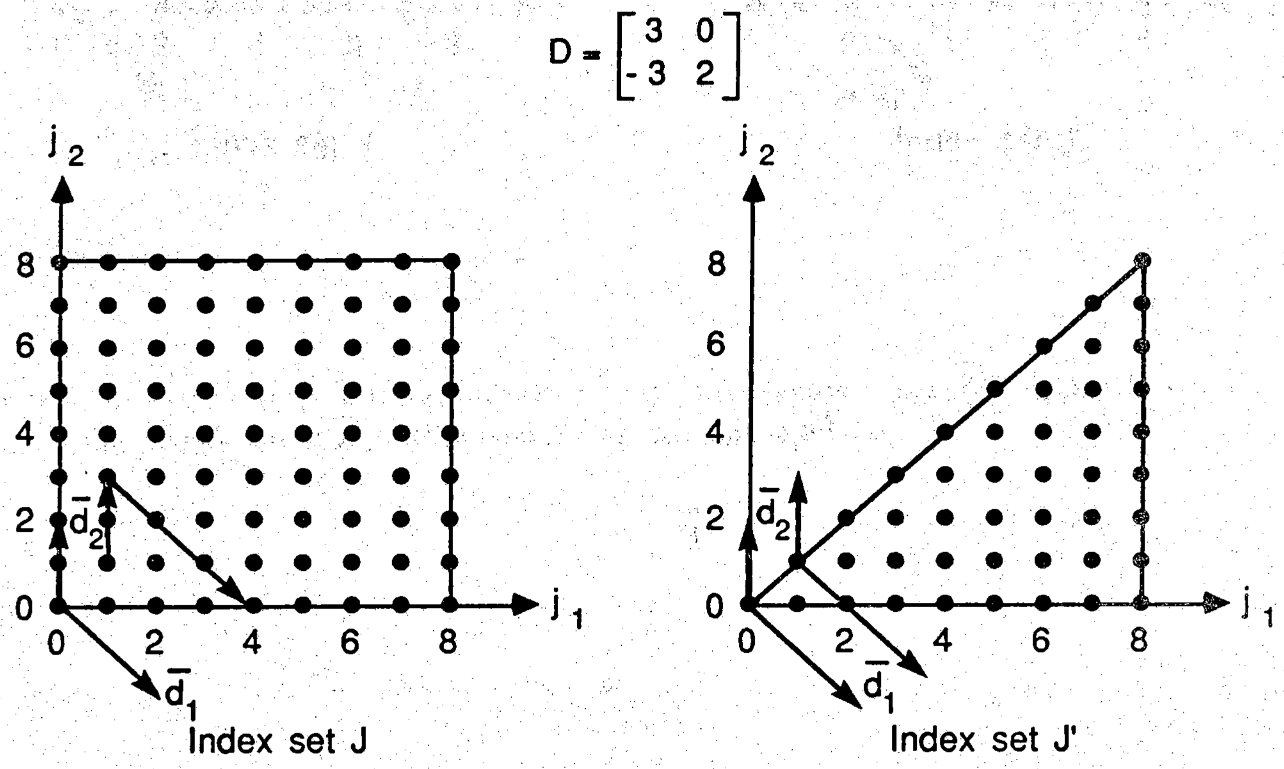
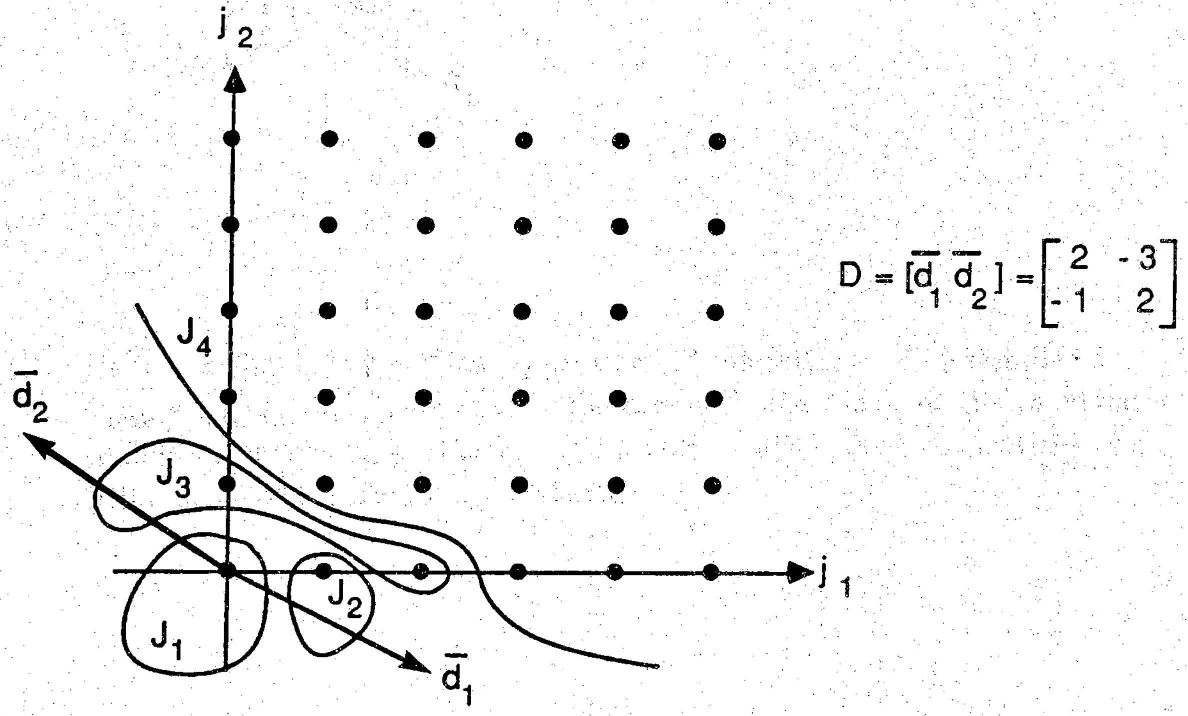


Figure 2.1 In J′, $[1,1]^T$ is not connected to any other points. However, in J it is connected to many other points such as point $[4,0]^T$.



$j_2$
$j_1$
$J_1$
$J_2$
$J_3$
$J_4$
$\bar{d}_1$
$\bar{d}_2$
$D = [\bar{d}_1\ \bar{d}_2] = \begin{bmatrix} 2 & -3 \\ -1 & 2 \end{bmatrix}$

Figure 2.2 The maximal independent partition of algorithm of Example 2.2. $P_{\max} = \{J_1, J_2, J_3, J_4\}$. However, there is only one block in the maximal pseudo-independent partition. Pictorially, only the connectivities of points near boundaries of J are affected.

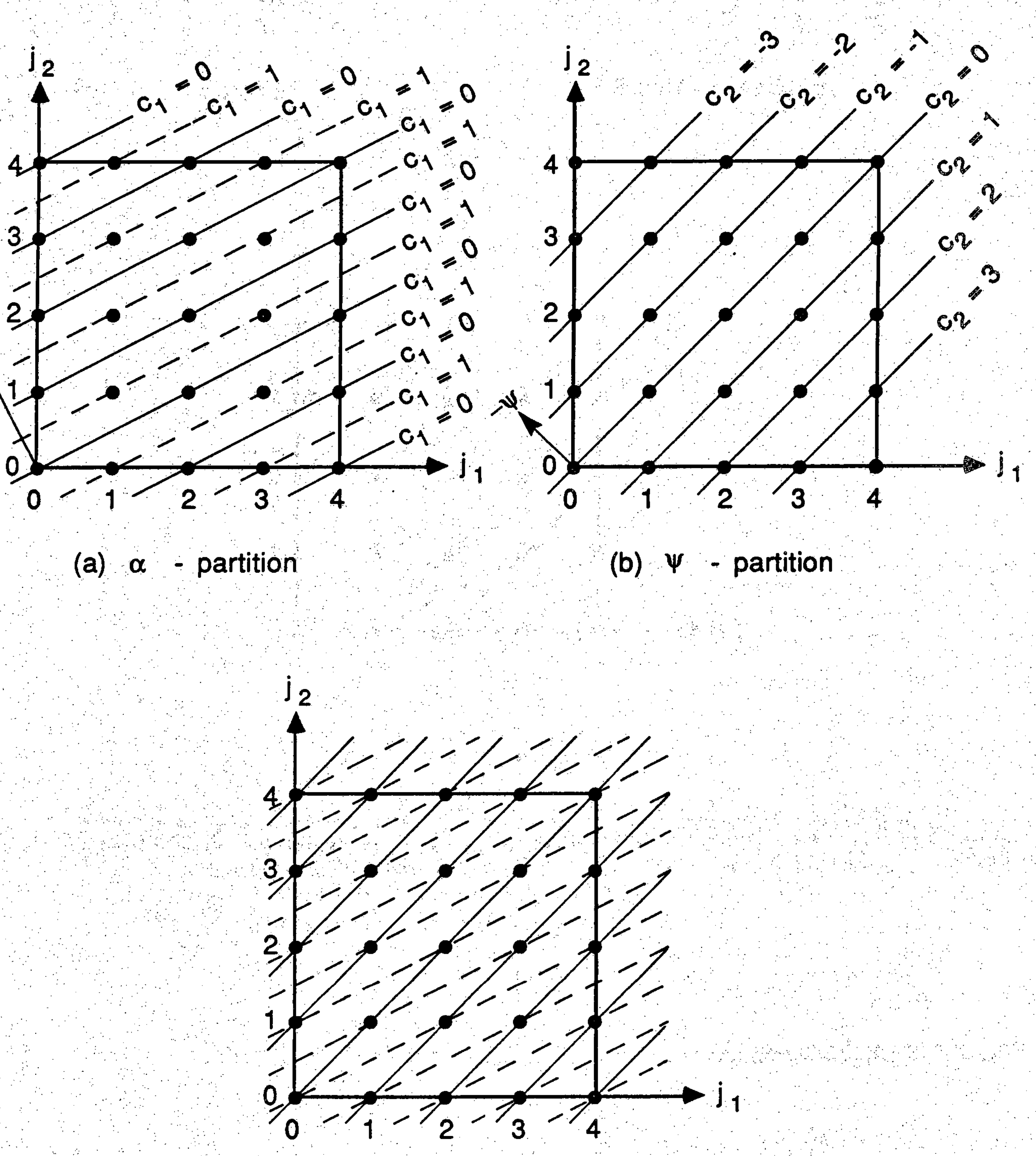


**Figure 4.1**: Partitions of algorithm of Example 4.1 where $D=[2, 2]^T$, $\Pi=[-1, 2]$ and $\Psi=[1, -1]$. (a) $\alpha$-partition: the hyperplanes are described by $\Pi\bar{j}(\text{mod}\,2)=c_1$. Points lying on dotted lines belong to $J_0 \in P_\alpha$ and points lying on solid lines belong to $J_1 \in P_\alpha$. (b) $\Psi$-partition: the hyperplanes are described by $\Psi\bar{j}=c_2$. Points lying on hyperplane $\Psi\bar{j}=c_2$ belong to $J_{(c_2)} \in P_\Psi$. (c) $\alpha\Psi$-partition: dotted lines specify $\alpha$-partition and solid lines specify $\Psi$-partition.

| Block | $J_{\begin{bmatrix}0\\-3\end{bmatrix}}$ | $J_{\begin{bmatrix}1\\-3\end{bmatrix}}$ | $J_{\begin{bmatrix}0\\-2\end{bmatrix}}$ | $J_{\begin{bmatrix}1\\-2\end{bmatrix}}$ | $J_{\begin{bmatrix}0\\-1\end{bmatrix}}$ | $J_{\begin{bmatrix}1\\-1\end{bmatrix}}$ | $J_{\begin{bmatrix}0\\0\end{bmatrix}}$ | $J_{\begin{bmatrix}1\\0\end{bmatrix}}$ | $J_{\begin{bmatrix}0\\1\end{bmatrix}}$ | $J_{\begin{bmatrix}1\\1\end{bmatrix}}$ | $J_{\begin{bmatrix}0\\2\end{bmatrix}}$ | $J_{\begin{bmatrix}1\\2\end{bmatrix}}$ | $J_{\begin{bmatrix}0\\3\end{bmatrix}}$ | $J_{\begin{bmatrix}1\\3\end{bmatrix}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Index Point | $\begin{bmatrix}0\\3\end{bmatrix}$ | $\phi$ | $\begin{bmatrix}0\\2\end{bmatrix}$ | $\begin{bmatrix}1\\3\end{bmatrix}$ | $\begin{bmatrix}0\\1\end{bmatrix}$ $\begin{bmatrix}2\\3\end{bmatrix}$ | $\begin{bmatrix}1\\2\end{bmatrix}$ | $\begin{bmatrix}0\\0\end{bmatrix}$ $\begin{bmatrix}2\\2\end{bmatrix}$ | $\begin{bmatrix}1\\1\end{bmatrix}$ $\begin{bmatrix}3\\3\end{bmatrix}$ | $\begin{bmatrix}2\\1\end{bmatrix}$ | $\begin{bmatrix}1\\0\end{bmatrix}$ $\begin{bmatrix}3\\2\end{bmatrix}$ | $\begin{bmatrix}2\\0\end{bmatrix}$ | $\begin{bmatrix}3\\1\end{bmatrix}$ | $\phi$ | $\begin{bmatrix}3\\0\end{bmatrix}$ |

**Table 4.1:** List of blocks and their corresponding index points of the $\alpha\Psi$-partition of algorithm of Example 4.1.