7-16-1990

Some NP-Complete Problems in Linear Algebra

Santhosh Sastry '90
Illinois Wesleyan University

Some NP-Complete Problems in Linear Algebra

Santhosh Sastry

Department of Mathematics

Illinois Wesleyan University

Bloomington, IL 61702

July 16, 1990

AN OVERVIEW

This research project is aimed at studying the theory of NP-Completeness and determining the complexity of certain problems in linear algebra. The first chapter introduces the reader to Complexity theory and defines NP-Completeness. It is supported by Appendices 1 and 2. Appendix 3 lists some known NP-Complete problems.

We desired to work on the following two open questions.

(1) Is the problem of determining whether a nonnegative matrix has nonnegative rank factorization NP-Complete?

(2) Is the problem of checking whether a given nonnegative matrix is weakly monotone NP-Complete?

We also spent considerable time on another problem:

(3) Is the problem of checking whether a nonnegative matrix contains an r x r monomial submatrix NP-Complete?

Before working on these problems, we needed to study the theory of nonnegative rank factorization intensively. The papers read to achieve this objective are described in Chapters 3 and 4. Chapter 2 contains a result which might have provided a transformation to one of the problems we were working on. From linear algebra, we know that nonnegative rank factorization exists if certain conditions are met. We tried to show that checking for these conditions is NP-Complete, and hence determining whether a nonnegative matrix has nonnegative rank factorization is NP-Complete.

We spent the most time on question (3). At the moment we believe that there is a good

chance that a transformation from the Clique problem in graph theory could be provided. We are currently looking at this possibility.

1 Introduction

1.1 NP-Completeness

1.1.1 Optimization and Decision problems

A **problem** is a general question to be answered involving several parameters with unspecified values, with or without bound conditions. For example, we may be looking for the maxima of a certain function, subject to certain constraints. An **instance** is a particular case of the problem where specific values have been assigned to the parameters. An **algorithm** is a general, step by step procedure for solving a problem. We are generally interested in finding the most efficient algorithm for solving a given problem, that is, the algorithm which runs on a computer in minimum time.

A **decision problem** is a problem which has only two possible answers, "Yes" or "No." Hence, a decision problem may be regarded as two disjoint sets, a set of "Yes" instances and a set of "No" instances. Quite often, a problem can be reformulated as a decision problem, the original problem being at least as difficult as the corresponding decision problem. To illustrate this point, we cast an optimization problem as a decision problem. An **optimization problem** is a problem wherein a function has to be maximized or minimized given certain constraints. Each optimization problem can be written as a decision problem (i.e, a problem which has only two answers, "Yes" or "No"), that includes a numerical bound B. We may desire to check if a solution exists such that it is less than or greater than a given B. Clearly, the given optimization problem is at least as difficult as

the related decision problem. Let us consider the travelling salesman problem (TSP) as an example. We have a finite set $C = \{c_1, ..., c_m\}$ of "cities" and, for each pair of cities c_i, c_j in C, $d(c_i, c_j)$ is the "distance" between them. A solution is an ordering $< c_{\pi(1)}, ..., c_{\pi(m)} >$ of the given cities that minimizes

$$\sum_{i=1}^{m} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(m)}, c_{\pi(1)}),$$

that is, we want to visit all the cities once, while travelling the minimum possible distance. The decision problem is as follows:

Instance: A finite set $C = \{c_1 ... c_m\}$ of cities, a "distance" $d(c_i, c_j) \in Z^+$ for each pair of nodes $c_i, c_j \in C$ and a bound $B \in Z^+$ (where Z^+ denotes the positive integers).

Question: Is there a "tour" of all the cities in C having total length not more than B, that is, an ordering $< c_{\pi(1)}, ..., c_{\pi(m)} >$ of C such that the distance

$$\sum_{i=1}^{m} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(m)}, c_{\pi(1)}) \leq B?$$

In general, the question becomes - Given a set of constraints, is the "cost"less than or equal to a numerical bound? The answer is either "Yes" or "No." So the optimization problem is at least as difficult as the corresponding decision problem. **In this paper, the word 'problem'implies a decision problem unless specifically stated otherwise.**

A problem is denoted by the Greek letter π and a decision problem by D_π. A particular instance of a problem is denoted by I. The set of instances of a decision problem for which the answer is "Yes" is denoted by Y_π.

1.1.2 Tractable, Intractable and Unsolvable problems

Tractable problems are those which may be computed "efficiently," that is, the time needed to compute a solution using an algorithm is a polynomial in the size of the problem. It may be said that an **intractable** problem is one which requires a tremendous amount of running time, that is, the running time is an exponential in the size of the problem. On the other hand, a **noncomputable function** (also called an **unsolvable problem**) is one which cannot be evaluated whatever the power of the computer and the running time allowed.[1] The difference between an intractable problem and an unsolvable problem is that, for intractable problems we can generate all possible solutions and check each one for validity whereas unsolvable problems cannot be solved at all.

Certain problems like the Travelling Salesman problem have no "efficient" solutions. The only approach seems to be a "brute force" approach (i.e exhausting all possibilities) which is known to be highly inefficient and in many cases, almost impossible in practice. Such problems are intractable.

1.1.3 The classes P and NP

A problem π belongs to the class **P** (where **P** stands for **P**olynomial) if there exists an algorithm which can solve π in polynomial time, that is, the time needed to produce an answer is a polynomial in the size of the problem. There are many problems which cannot be solved by polynomial time algorithms. The only approach for some of these

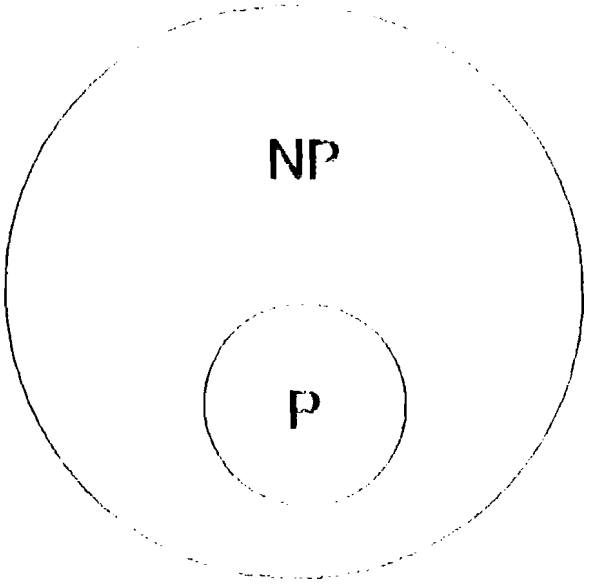
[1] For example, the **The Halting Problem** for Turing machines is unsolvable. The Halting problem is as follows: Given a Turing machine M and an input x, will M eventually halt?

problems seems to be an exhaustive search of all possibilities. A possibility is guessed and then verified, that is, it is checked to see if it is a valid solution. If this verification can be done in polynomial time, the problem belongs to the class **NP** (which stands for Nondeterministic Polynomial).

A problem does not necessarily have to be in the class **P** or **NP**. It may belong to classes like Co-**NP** or **NP**-Hard which are not discussed in this paper. We shall now briefly consider the relationship between the classes **P** and **NP**.

It is fairly obvious that **P** is a subset of **NP**. Consider a problem π solved by a polynomial time algorithm A. We can use A as a polynomial time nondeterministic algorithm by considering A as the checking stage and ignoring the guessing stage. Therefore $\pi \in \mathbf{P}$ implies that $\pi \in \mathbf{NP}$. Therefore it may be concluded that **P** is a subset of **NP**.

Figure 1.1



It is clear that **P** is a subset of **NP**; but is **NP** a subset of **P**? This is an open question. So far it has not been proved that **NP** is a subset of **P** or that **NP** is not a subset of **P**. If **NP** is a subset of **P**, then **P=NP**. Given the present state of knowledge, it is reasonable to assume that **NP** is not equal to **P** and this is the widely held belief. If **P** is not equal to **NP**, then the distinction between **P** and **NP\P** is important. All problems in **P** can be solved in polynomial time and all **NP\P** problems are intractable. If a given decision problem $\pi \in$ **NP**, and **P** is not equal to **NP**, then we would like to know whether π is in **P** or **NP\P**. Until it is proved that **P** is not equal to **NP**, it is not possible to show that a given problem lies in **NP\P**. Figure 1.1 depicts the widely accepted, hypothetical relationship between the classes **P** and **NP**.

A third class of problems known as the class **NP-Complete** may shed some light on the question of whether P is equal to NP. Informally, the class NP-Complete may be thought of as a subclass of the "most difficult" problems in NP. A formal definition will be provided later on. Once again, we stress that the common belief is that **P** is not equal to **NP**.

1.1.4 Complement of a problem

Suppose we have a problem π of the form "Given I, is X true for I?" Its complementary problem π^c is of the form "Given I, is X false for I?" The following points are to be noted:

1. Membership in **P** for a problem π implies membership in P for its complement

2. Membership in **NP** for a problem π does not appear to imply membership in NP for its complement.

The Travelling Salesman problem mentioned earlier belongs to the class **NP**. We can guess the cities to be visited and check in polynomial time whether the total distance travelled by choosing the cities in this particular order is less than the numerical bound B. When we find such a tour we may stop. The complement of the Travelling Salesman problem is as follows: Given a set of cities, distances between cities and a numerical bound B, is there no tour of all the cities so that the total distance travelled is B or less? To answer this question, we may have to look at all possible tours. Hence it appears that a polynomial time nondeterministic algorithm for the complement of the Travelling Salesman problem does not exist.

1.1.5 Polynomial Transformations

A key idea in the study of **NP**-Completeness is the idea of a polynomial transformation.

Definition 1 *A* **polynomial transformation** *from a problem π_1 to a problem π_2 is a function $f : D_{\pi_1} \longrightarrow D_{\pi_2}$ that satisfies the two conditions:*

1. *For all $I \in D_{\pi_1}$, $f(I)$ can be computed in polynomial time*

2. *For all $I \in D_{\pi_1}$, $I \in Y_{\pi_1}$ if and only if $f(I) \in Y_{\pi_2}$.*

A polynomial transformation from π_1 to π_2 is written as $\pi_1 \propto \pi_2$ and is read "π_1 transforms to π_2."

Definition 2 *A problem π is defined to be* **NP-Complete** *if*

1. $\pi \in$ **NP** *and*

2. $\pi' \in$ **NP** *implies that* $\pi' \propto \pi$.

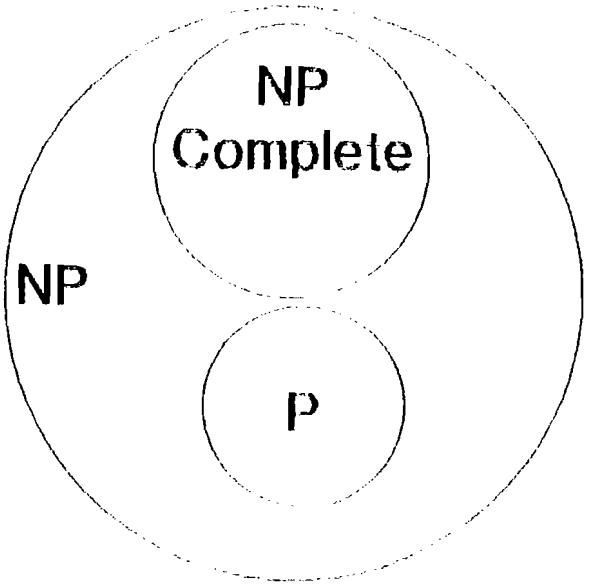
The following properties which are fairly obvious are useful in proving that a given problem is NP-Complete.

1. If $\pi_1 \propto \pi_2$, then $\pi_2 \in$ **P** implies $\pi_1 \in$ **P** and equivalently, π_1 does not belong to **P** implies that π_2 does not belong to **P**.

2. If $\pi_1 \propto \pi_2$ and $\pi_2 \propto \pi_3$, then $\pi_1 \propto \pi_3$.

Two problems π_1 and π_2 are **polynomially equivalent** if $\pi_1 \propto \pi_2$ and $\pi_2 \propto \pi_1$.

Figure 1.2



A relation R is **reflexive** on a set S if aRa for all $\in$ S. The relation is **symmetric** if aRb implies that bRa where a,b $\in$ S. The relation is **transitive** if aRb and bRc implies that aRc. If a relation is reflexive, symmetric and transitive, it is called an **equivalence relation**. From the above proposition it can be seen that polynomial equivalence is an equivalence relation. **P** is an equivalence class which consists of the computationally "easiest" decision problems, **NP**-Complete is the class which consists of the "hardest" decision problems. Notice that if the classes **P** and **NP**-Complete are not disjoint, then P=NP. Figure 1.2 depicts **P** and **NP**-Complete as disjoint; this is the widely held belief.

Finally notice that a given problem π can be shown to be **NP**-Complete in the following fashion:

1. Show that π belongs to the class **NP**,
2. Select a known **NP**-Complete problem π',
3. Construct a polynomial transformation from π' to π.

A few known **NP**-Complete problems are stated in Appendix 3.

1.2 Nonnegative Rank Factorizations

Our interest lies in deciding the complexity of certain problems in linear algebra. We studied the theory of **NP**-Completeness because we feel that the problem of determining whether a given nonnegative matrix has nonnegative rank factorization is **NP**-Complete. We proceed to describe the nonnegative rank factorization problem after explaining our

notation and stating a few definitions.

A **nonnegative matrix** is a matrix in which all the entries are greater than or equal to zero. It is assumed that all the matrices mentioned in this paper are nonnegative, unless specifically stated otherwise. A nonnegative matrix A which is $m \times n$ and has rank r is written as $A \in P_r^{m \times n}$, or as $A = [a_{i,j}]_{m,n}^r$. The column space of A is denoted by $R(A)$. The null space of A is denoted by $N(A)$ and the left null space by $N(A^T)$.

The following definitions provide an understanding of some of the key terms in this paper.

Definition 3 *A factorization $A=BC$ where $A \in P_r^{m \times n}, B \in P_r^{m \times r}$ and $C \in P_r^{r \times n}$ is called* **nonnegative rank factorization.**

Definition 4 *A nonnegative $m \times n$ matrix A is* **nonnegative weakly monotone** *if $R(A) \cap \mathbf{R}_+^m = A\mathbf{R}_+^n$ ($\mathbf{R}_+^m$ is the nonnegative orthant of $\mathbf{R}^{n \times 1}$). Alternately, A is nonnegative weakly monotone if and only if $Ax \geq 0$ implies that $x \in N(A) + \mathbf{R}_+^n$.*

Definition 5 *A* **monomial matrix** *is a square matrix with exactly one nonzero element in each row and in each column.*

The study of nonnegative rank factorization has important applications in the field of nonnegative generalized inverses of nonnegative matrices. Not all nonnegative matrices have nonnegative rank factorizations. There appears to be no simple test to determine if a given nonnegative matrix has a nonnegative rank factorization, which prompts us to pose the following question:

(1) Is the question of determining whether a given nonnegative matrix has nonnegative rank factorization **NP**-Complete.

It was the aim of this research project to investigate this question. It is highly probable that the answer is "Yes."

The chief difficulty in proving that a given problem is **NP**-Complete lies in finding a suitable problem from which a polynomial transformation can be constructed to the given problem. Problems in linear algebra appear to be more difficult than problems in other fields like graph theory. This is because, to the best of our knowledge, very few problems in this field have been shown to be **NP**-Complete. The authoritative book on **NP**-Completeness, *Computers and Intractability : A guide to the theory of NP-Completeness* by M.R Garey and D.S Johnson lists very few known NP-Complete problems in the realm of linear algebra. The few results listed are not relevant to our problem. Hence we read the paper "Degeneracy testing is NP-Complete" which may be related to the problem we are working on. This result and its relevancy to our problem is described in Chapter 2.

Since the problem under consideration appeared to be rather hard, we decided to impose a restriction, which gave us a new problem.

(2) Is the question of determining whether a nonnegative, weakly monotone matrix has nonnegative rank factorization **NP**-Complete?

Notice that this is (1) with an additional condition, the condition being that the nonnegative matrix is weakly monotone. If this question is NP-Complete, the original question is also **NP**-Complete. By imposing an additional condition, we are looking at a slightly

different problem. It is evident that if (2) is intractable, the original problem (1) is also intractable.

This line of thought led us to carefully consider some of the work of Jeter and Pye which is described in Chapter 2. A perusal of this paper raised another question:

(3) Is the problem of determining whether a given nonnegative matrix of rank r has an $r \times r$ monomial submatrix NP-Complete?

This is the problem we have spent most of our time working on.

From our readings in linear algebra (described in the pages to come), we gathered that a nonnegative matrix has nonnegative rank factorization if certain conditions are met. We tried to show that checking for these conditions is NP-Complete, for this would then be the same as saying that checking for nonnegative rank factorization is NP-Complete. We develop this line of thought more fully in the coming chapters. When this approach did not yield the results we desired, we decided to look at the dual of the problem. In mathematics, a better approach to a problem may sometimes be obtained by looking at the dual of a problem.

In conclusion, we mention an idea which might provide the transformation we are looking for. We intend to pursue this matter further and we are currently working on it.

2 Nonnegative Rank Factorizations

2.1 Nonnegative Rank Factorizations of Nonnegative Weakly Monotone Matrices

In the previous chapter, we introduced the problems we are working on. Rather than attacking the problem of determining whether a given nonnegative matrix has a nonnegative rank factorization, we considered restricting our attention to nonnegative weakly monotone matrices and then trying to determine whether a given nonnegative weakly monotone matrix has nonnegative rank factorization. The second question (stated in the previous chapter) was suggested by Jeter and Pye's paper, "A Note on Nonnegative Rank Factorizations,"published in *Linear Algebra and its applications 38:171-173 (1981).* Here we describe a result from the above paper which says that a nonnegative weakly monotone matrix of rank r has nonnegative rank factorization if and only if it has a $r \times r$ monomial submatrix. We are studying the conditions that must be met for a nonnegative weakly monotone matrix to have nonnegative rank factorization and trying to decide if checking for these conditions is NP-Complete.

It is to be noted that all matrices considered are nonnegative matrices and the notation described in the previous chapter holds. We will need a few more definitions for what follows.

Definition 6 *Let $A \in \mathbf{R}^{m \times n}$. Then any matrix $X \in \mathbf{R}^{n \times m}$ that satisfies $A=AXA$ is called a $\{1\}$-inverse of A. [2]*

Definition 7 *A real matrix* $A \in \mathbf{R}^{m\times n}$ *is said to be* **monotone** *if* $Ax \geq 0$ *implies that* $x \geq 0$ *for all* $x \in \mathbf{R}^{n\times 1}$. *[7]*

Definition 8 *A set C is said to be a* **convex cone with vertex zero** *if* x_1 *and* $x_2 \in C$ *imply that every semipositive combination of* x_1 *and* x_2 *belong to C. [1]*

Definition 9 *Let C be a convex cone with vertex the origin. Let* $x \in C$. *Then x is an* **extremal element of C** *(or generator of C) if and only if x=y+z, where* $y, z \in C$ *implies that* $y = \alpha x$ *and* $z = \beta y$ *for scalars* α *and* β.

The following lemma is useful in proving Theorem 2, which is the main result in this subsection.

Lemma 1 *If A is a nonnegative weakly monotone matrix with nonnegative rank factorization A=BC, then* $A\mathbf{R}_+^n = B\mathbf{R}_+^r$.

PROOF:

It is known that A=BC. Therefore $A\mathbf{R}_+^n = B(C\mathbf{R}_+^n)$. From the definition of nonnegative rank factorization we know that $C \in \mathbf{R}^{r\times n}$ and has rank r. Hence $A\mathbf{R}_+^n = B(C\mathbf{R}_+^n) \subseteq B\mathbf{R}_+^r$. Therefore

$$A\mathbf{R}_+^n \subseteq B\mathbf{R}_+^r \qquad \ldots(1)$$

Since C has full row rank, it has a right inverse C^-. It is given that A = BC. Multiplying both sides by C^-, we have $AC^- = BCC^- = BI_r = B$.

Notice that C^- is an $n \times r$ matrix. Multiplying both sides by $\mathbf{R}_+^r$, we have $AC^-\mathbf{R}_+^r = B\mathbf{R}_+^r$. Therefore $B\mathbf{R}_+^r \subseteq R(A) \cap \mathbf{R}_+^m$. Using definition 4,

$$B\mathbf{R}_+^r \subseteq A\mathbf{R}_+^n \qquad \ldots(2)$$

From (1) and (2),

$$A\mathbf{R}_+^n = B\mathbf{R}_+^r.$$

Theorem 2 *If $A \in P_r^{m \times n}$ is weakly monotone and has nonnegative rank factorization A=BC, then each column of B is proportional to a column of A, and each row of C is proportional to a row of A.*

PROOF:

By Lemma 1, we have the two equal cones $A\mathbf{R}_+^n = B\mathbf{R}_+^r$. The generators of $A\mathbf{R}_+^n$ must come from the columns of A, denoted by $A^{(1)}, \ldots, A^{(n)}$. In cone $B\mathbf{R}_+^r$, all the columns of B are used as generators because B has full column rank. Since the two cones are equal, $B^{(1)}, \ldots, B^{(r)}$ (or some multiple) must also be the generators of $A\mathbf{R}_+^n$. Therefore, each column in B is a scalar multiple of some column in A.

It is now to be shown that each row in C is proportional to some row in A. We know that A=BC. Therefore, if B has an $r \times r$ monomial submatrix, it follows that each row in C is proportional to some row in A. We must still show that B has an $r \times r$ monomial submatrix. Assume that $By \geq 0$. We know that $B = AC^-$. Multiplying both sides by y, $By = A[C^-y]$. Let

$C^{-}y = x$. Then By=Ax. Since $By \geq 0, Ax \geq 0$. Since A is weakly monotone, $Ax \geq 0$ implies that $x \in N(A) + \mathbf{R}^n_+$. Let $x = \alpha + \beta$ where $\alpha \in N(A)$ and $\beta \in \mathbf{R}^n_+$. Since $By = Ax$, $By = A\alpha + A\beta = A\beta = BC\beta$. But $N(B) = \{0\}$ and $B(y - C\beta) = 0$. It follows that $y = C\beta$ where C and β are both nonnegative. Therefore $y \geq 0$. Since $By \geq 0$ implies that $y \geq 0$, B is monotone (from definition 7, the definition of a monotone matrix), which ensures that B has a nonnegative {1}-inverse. From the paper "Inverses of nonnegative matrices" (*Linear and Multilinear Algebra 2:161-172, 1974*) by A. Berman and R.J Plemmons, we know that if B has a nonnegative {1}-inverse, it has an $r \times r$ monomial submatrix where r is the rank of the matrix. Hence each row in B is proportional to some row in A.

Corollary 3 *A nonnegative weakly monotone matrix of rank r has a nonnegative rank factorization if and only if it has a $r \times r$ monomial submatrix.*

From the corollary, we can say that checking a nonnegative weakly monotone matrix for nonnegative rank factorization is **NP**-Complete if and only if checking to see whether a nonnegative weakly monotone matrix has an $r \times r$ monomial submatrix is **NP**-Complete. If the problem of determining whether a nonnegative weakly monotone matrix has an $r \times r$ monomial submatrix is **NP**-Complete, then the problem of determining whether any nonnegative matrix has nonnegative rank factorization is **NP**-Complete. This is problem (1) stated in the previous chapter. If the answer to problem (2) is "Yes", then the answer to problem (1) is also "Yes." It is fairly obvious that the problem belongs to the class

NP since we can guess an $r \times r$ submatrix and verify if the submatrix is monomial in polynomial time. We tried to establish a polynomial transformation from a known **NP**-Complete problem (the **degeneracy testing problem**) to problem (3). We chose this particular problem (described in the pages to come) because we believe that both problems are equally "hard." Therefore if the degeneracy testing problem is **NP**-Complete, problem (3) could also be **NP**-Complete. This idea is explained in more detail later on.

2.2 Degeneracy testing is NP-Complete

This result first appeared in the paper, "Some NP-Complete problems in linear programming" by R. Chandrasekaran, S.N Kabadi and K.G Murty (Operation Research Letters, Vol.1, Number 3).

Degeneracy testing in the linear programming problem.

Instance: The linear programming problem with integer data

$$\text{Minimize } Z(x) = cx$$

$$\text{subject to } Ax = b, x \geq 0$$

where A is an $m \times n$ matrix of rank m.

Question: Is the problem **degenerate**; that is, does there exist a basis B for the column space of A, such that at least one component in $B^{-1}b$ is zero?

Before trying to determine the complexity of this problem, we need to establish other results.

Subset Sum problem (SSP):

Instance: Given positive integers $\{a_1, ..., a_p, b\}$.

Question: Does there exist $\emptyset \neq I \subseteq \{1, ..., p\}$, where $\sum_{i \in I} a_i = b$?

This problem is known to be **NP-Complete** (Garey and Johnson).

Equal Partial Sums Problem (EPSP):

Instance: A pair of nonempty sets of positive integers $\{a_1, ..., a_m\}$ and $\{b_1, ..., b_n\}$.

Question: Are there nonempty subsets I and J (I is a subset of $\{1, ..., m\}$ and J is a

subset of $\{1, ..., n\}$) such that $\sum_{i \in I} a_i = \sum_{j \in J} b_j$?

Proposition 4 *Equal Partial Sums problem is NP-Complete.*

PROOF:

We can guess subsets I and J and verify in polynomial time if $\sum_{i \in I} a_i = \sum_{j \in J} b_j$. Hence, EPSP belongs to the class **NP**.

We have to show that SSP $\propto$ EPSP. Select an instance $\{a_1, ..., a_p, b\}$ of SSP. This can be written as $\{\{a_1, ..., a_p\}, \{b\}\}$. This is the EPSP problem with n=1. In the mapping

$$\{a_1, ..., a_p, b\} \longrightarrow \{\{a_1, ..., a_p\}, \{b\}\},$$

it is clear that

$$\{a_1, ..., a_p, b\} \in Y_{SSP}$$

if and only if

$$\{\{a_1, ..., a_p\}, \{b\}\} \in Y_{EPSP}.$$

This transformation can be computed in polynomial time. Hence the Equal Partial Sums problem is **NP**-Complete.

Equal Proper Partial Sums Problem (EPPSP):

Instance: A pair of nonempty sets of positive integers $\{a_1, ..., a_m\}$ and $\{b_1, ..., b_n\}$.

Question: Are there nonempty, proper subsets I and J (I is a subset of $\{1, ..., m\}$ and J is a subset of $\{1, ..., n\}$ such that $\sum_{i \in I} a_i = \sum_{j \in J} b_j$?

Proposition 5 *The Equal Proper, Partial Sums Problem is NP- Complete.*

PROOF:

We can guess proper, subsets I and J and verify in polynomial time if

$$\sum_{i \in I} a_i = \sum_{j \in J} b_j.$$

Hence, EPPSP belongs to the class NP.

We must establish that SSP $\propto$ EPPSP. Let $\{a_1, ..., a_p, b\}$ be an instance of SSP. Define positive integers a_{p+1} and b' by $a_{p+1} = 1 + \sum_{i \in I} a_i + b$ and $b' = b + 4a_{p+1}$. Consider the polynomial transformation

$$f : D_{SSP} \longrightarrow D_{EPPSP}$$

where

$$f(a_1, ..., a_p; b) = (\{a_1, ..., a_p, a_{p+1}\}, \{b, b'\}).$$

If $(a_1, ..., a_p, b) \in Y_{SSP}$, then there exists a nonempty, proper subset I of $\{1, ..., p\}$ (which is a subset of $\{1, ..., p+1\}$) such that $\sum a_i = b$. Let $J = \{1\}$. Then J is nonempty and is a proper, subset of $\{1, 2\}$ and $\sum_{i \in I} a_i = \sum_{j \in J} b_j$. Hence $(\{a_1, ..., a_p, a_{p+1}\}, \{b, b'\}) \in Y_{EPPSP}$. Assume that

$$f(a_1, ..., a_p, b) = (\{a_1, ..., a_p, a_{p+1}\}, \{b, b'\}) \in Y_{EPPSP}.$$

Then there exists nonempty, proper subsets I (a subset of $\{1, ..., p+1\}$) and J (a subset of $\{1, 2\}$) for which $\sum_{i \in I} a_i = \sum_{j \in J} b_j$. Since J is a proper, subset of $\{1, 2\}$

then $J = \{1\}$ or $J = \{2\}$. Suppose $J = \{2\}$. Then

$$\begin{aligned} b + 4a_{p+1} = b' &= \sum_{i \in I} a_i \\ &\leq \sum_{i \in I \setminus \{p+1\}} a_i + a_{p+1} \\ &\leq \sum_{i \in \{1,\ldots,p\}} a_i + a_{p+1} \\ &< 2a_{p+1} \end{aligned}$$

This implies that $b \leq b + 2a_{p+1} < 0$, which is a contradiction. Therefore, $J = \{1\}$. So $b = \sum_{i \in I} a_i$. Suppose $p+1 \in I$. Then

$$b = \sum_{i \in I} a_i = \sum_{i \in I \setminus \{p+1\}} a_i + a_{p+1} = \sum_{i \in I \setminus \{p+1\}} a_i + 1 + \sum_{i \in \{1,\ldots,p\}} a_i + b.$$

This implies that $0 = \sum_{i \in I \setminus \{p+1\}} a_i + 1 + \sum_{i \in \{1,\ldots,p\}} a_i > 1$, which is also a contradiction. Therefore p+1 is not an element of I. Hence I is a proper subset of $\{1, \ldots, p+1\}$ and $J = \{1\}$ is a proper subset of $\{1, 2\}$. So I is nonempty and it is a proper subset of $\{1, \ldots, p\}$ such that $\sum_{i \in I} a_i = b$. Thus

$$f(a_1, \ldots, a_p, b) = (\{a_1, \ldots, a_p, a_{p+1}\}, \{b, b'\}) \in Y_{EPPSP}$$

implies that $(a_1, \ldots, a_p, b) \in Y_{SSP}$. Hence SSP $\propto$ EPPSP and EPPSP is **NP**-Complete.

Theorem 6 *Degeneracy testing is NP-Complete.*

PROOF:

(1) It is easy to see that degeneracy testing belongs to the class **NP**. We can guess a basis and look for a zero element in $B^{-1}b$ in polynomial time.

(2) Consider the following transportation problem.

$$\text{Minimize } \sum_{i,j} c_{i,j}x_{i,j} \text{ subject to the constraints}$$

$$\sum_{j=1}^{n} x_{i,j} = a_i, \text{ for } i = 1, ..., m$$

$$\sum_{i=1}^{m} x_{i,j} = b_j, \text{ for } j = 1, ..., n$$

$$x_{i,j} \geq 0$$

for all i,j where $\{a_1, ..., a_m\}$ and $\{b_1, ..., b_n\}$ are positive integers satisfying $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. It is degenerate if and only if there exists proper, nonempty subsets I (a subset of $\{1, ..., m\}$) and J (a subset of $\{1, ..., n\}$) such that $\sum_{i \in I} a_i = \sum_{j \in J} b_j$ (Theorem 13.11, pp 398, *Linear Programming* by K.G Murty). This is the same as EPPSP, which is known to be **NP**-Complete (Proposition 5). Therefore degeneracy testing for the transportation problem is **NP**-Complete. Hence degeneracy testing for the linear programming problem is **NP**-Complete.

Note: The transportation problem is a special case of the linear programming problem. If we are unable to test degeneracy for a special case, we will not be able to do so for the general problem. Hence degeneracy testing for the linear programming problem is **NP**-Complete.

We were looking for ways to tackle problems (1), (2) and (3). In other words, we were searching for suitable problems which could be transformed into these problems. Hence we studied the degeneracy testing problem. We hoped to transform the degeneracy testing problem into problem (3). In the degeneracy testing problem, we were looking for

occurrences of zero entries in column matrices. In problem (3) we were looking for a nonzero entry in $r \times r$ monomial submatrices. Notice that in both problems, we were searching for a particular element in a matrix. Hence we thought it probable that they both belong to the same Complexity class. Since we just showed that degeneracy testing is **NP**-Complete, problem (3) might also be **NP**-Complete.

We started out by showing that problem (3) belongs to the class **NP**. For a given nonnegative matrix, we can "guess" an $r \times r$ submatrix. Obviously, such a submatrix is square. It is possible to verify in polynomial time whether each row and column contains exactly one nonzero element. Hence problem (3) belongs to the class **NP**. But the polynomial transformation could not be worked out satisfactorily because the sets of "Yes" instances in the two problems did not correspond.

In addition to this, we tried various other approaches as well. We tried to transform the **Subset Sum problem** into problem (3) (please refer to the statements of the problems). We tried to come up with a correspondence between the set $\{a_1, ..., a_p\}$ and a row in the nonnegative matrix. The integer p which gives the size of the set would have to equal n, the number of columns, and i would have to equal r, the size of the monomial submatrix.

We also tried to transform **EPSP** into problem (3). We tried to establish a correspondence between the set $\{a_1, ..., a_m\}$ and a row in the matrix and $\{b_1, ..., b_n\}$ and a column in the matrix. The matrix would then have to be $n \times m$. But we could not ensure that the sets $\{a_1, ..., a_m\}$ and $\{b_1, ..., b_n\}$ had a common element, which would be the case in problem (3).

A serious approach was made to transform the **Clique problem** (see Appendix 3) into problem (3). Earlier it was shown that problem (3) belongs to the class **NP**. We imposed an additional condition on the Clique problem; the condition being that no vertex be connected to itself. The problem remains **NP**-Complete even then. We shall refer to this modified Clique problem as the Clique problem.

We write the graph in the Clique problem as a matrix in the standard way (an edge between two nodes being represented by a 1 in the matrix, the absence of an edge between two nodes being represented by a 0). Notice that the Clique in the graph shows up as a submatrix of the matrix. If the Clique has r nodes, we obtain an $r \times r$ submatrix. In the submatrix all the entries are 1's, except for one zero in each row and column. We can replace all the 1's by 0's and all the 0's by 1's. Then the submatrix is a diagonal submatrix, which is a special case of the monomial submatrix. The same result is obtained with less effort if an edge between two nodes is represented by 0, and the absence of an edge by 1.

These approaches did not take us where we had hoped they would. With some more careful work it is possible that a polynomial transformation can be constructed to problem (3) from one of these problems. Rather than stick to one path, we decided to consider certain duality theorems of the nonnegative rank factorization problem in the hope that they would throw fresh light on the problem. Our efforts in this direction are discussed in the next chapter.

3 Some duality theorems for nonnegative rank factorizations

This chapter discusses some of the results which first appeared in the paper "Some Duality theorems for nonnegative rank factorizations" by Jeter and Pye (*Industrial Mathematics, Vol. 33, Part 1, 1983*). Throughout this chapter, A denotes an $m \times n$ matrix of rank r with nonnegative, real entries and M represents a $p \times q$ matrix.

We need to know more about nonnegative rank factorizations before attempting to settle the complexity of determining if nonnegative rank factorization exists for a given nonnegative matrix. Sometimes it is advantageous to study the dual of a problem since we often get a clearer picture by doing so. Also we may be able to solve the dual of a problem in less computer time than the original problem. Hence we decided to study some duality theorems for the nonnegative rank factorization problem. Once again, we are interested in the conditions which, if met, ensure that there exists a nonnegative rank factorization for a nonnegative matrix. We can then attempt to show that checking for these conditions is an NP-Complete problem. We are particularly interested in theorems which guarantee a nonnegative rank factorization for a nonnegative matrix A if certain properties are exhibited by the cone(s) generated by the columns of A. This approach requires the concept of a cone which was defined in the previous chapter. We also need the additional concepts of polar and polyhedral cones.

Definition 10 *The* **polyhedral cone** *C(M)is the cone generated by the columns of M*

and is defined to be

$$C(M) = M\mathbf{R}_q^+.$$

Definition 11 *The* **polar cone** *of of C(M) is denoted by* $C(M)^*$ *and is defined to be*

$$C(M)^* = \{y : M^T y \leq 0\}.$$

The following theorems are stated without proof. The proofs may be found in *Foundations of Optimization*, Lecture Notes in Economics and Mathematical Systems by M.S Bazaraa and C.M Shetty (Springer-Verlag, 1976). We will need these theorems to prove some of the results in this chapter.

Theorem 7 *Let* $C = \{x : Ax \leq 0\}$ *where A is an* $m \times n$ *matrix. Then C is a polyhedral cone.[1]*

Theorem 8 *Let* $C = \{x : x \geq 0\}$ *and* $A^T x = 0$. *This is a polyhedral cone.[1]*

The next two propositions (stated without proofs) are also from Bazaraa and Shetty.

Proposition 9 *If* $C(M) \subseteq C(T)$, *then* $C(T)^* \subseteq C(M)^*$. *[1]*

Proposition 10 *The polar cone of a polar cone of M is its polyhedral cone, that is,* $C(S)^{**} = C(S)$.*[1]*

Proposition 11 *A has nonnegative rank factorization if and only if there exists a nonnegative* $m \times r$ *matrix B of rank r such that* $C(A) \subseteq C(B)$.

This is evident from the lemma in the previous chapter.

Proposition 12 *A has nonnegative rank factorization if and only if there exists a nonnegative $m \times r$ matrix B of rank r such that $C(B)^* \subseteq C(A)^*$.*

PROOF:

Assume that A has nonnegative rank factorization. Let $z \in C(B)^*$. Then $B^T z \leq 0$. Therefore $A^T z = (BC)^T z = C^T B^T z \leq 0$. Since $B^T z \leq 0$ and C^T is nonnegative. Hence $C(B)^* \subseteq C(A)^*$.

Let $C(B)^* \subseteq C(A)^*$. Then $C(A) = C(A)^{**} \subseteq C(B)^{**} = C(B)$. From the earlier proposition, A has nonnegative rank factorization.

We are interested in certain duality theorems which may throw new light on problem (1). We have studied the definitions of polar and polyhedral cones. We wish to connect the question of the existence of nonnegative rank factorization with the existence of cones with certain properties. More specifically, we are interested in finding out if nonnegative rank factorization for a nonnegative matrix is guaranteed if the columns of the matrix generate a cone with certain properties. Such a result would give us another approach to the problems we are interested in. Then, if we can show that finding a cone with the specified properties is NP-Complete, the nonnegative rank factorization question is also NP-Complete. We attempt to describe such a theorem in the pages to come.

The following is a brief explanation of the notation to be used. Consider a $p \times q$ matrix M. Recall that the polar cone of the cone generated by the columns of M is the polyhedral cone $C(M)^* = \{y : M^T y \leq 0\}$. Since the intersection of two polyhedral cones is again a polyhedral cone, $M^T R_p \cap R_q$ is also a polyhedral cone [1]. A polyhedral cone is generated

by its extremal elements, which may be considered as the columns of a matrix. Hence there exists a $q \times k_1$ matrix U_M which generates the cone which is obtained by the intersection of $M^T\mathbf{R}_p$ and $\mathbf{R}_q^+$, that is, $M^T R_p \cap \mathbf{R}_q^+ = U_M \mathbf{R}_{k1}^+ = C(U_M)$, where the columns of U_M are the extremal elements of $M^T\mathbf{R}_p \cap \mathbf{R}_q^+$. Since each column of U_M belongs to $M^T\mathbf{R}_p$ there exists a $p \times k_1$ matrix V_M such that $M^T V_M = U_M$. Next let $N(M^T) = \{y \in \mathbf{R}_p : M^T y = 0\}$. Then there exists a $p \times k_2$ matrix W_M such that $N(M^T) = \{W_M z : z \geq 0\} = C(W_M)$.

The following result (from [1]) is stated without proof.

Theorem 13 *:* **Minkowski's Theorem -** $C(M)^* = C([-V_M W_M])$.

We shall next establish a lemma which in turn will allow us to establish the main duality result for the problem in question.

Lemma 14 *A has nonnegative rank factorization if and only if there exists a polyhedral cone*

$$K = \{Fz : z \geq 0\} \in C(A)^* = C([-V_A W_A])$$

such that F is an $m \times k$ matrix of rank m and

(1) $C(A) \in K^* = \{y : F^T y \leq 0\} = C([-V_F W_F])$,

(2) $[-V_F W_F] \geq 0$,

(3) rank $[-V_F W_F]$ = *rank* A,

(4) $[-V_F W_F]$ *is* $m \times r$.

PROOF:

Assume that the polyhedral cone K exists. This is the cone generated by the block matrix $[-V_A W_A]$. C(A) is contained in $K^* = \{z : F^T z \leq 0\}$. So $A = [-V_F W_F]D$ for $D \geq 0$. It follows from properties (1) through (4) that $[U_F W_F]$ is a nonnegative rank factorization for A.

Next assume that A has nonnegative rank factorization, that is, A=BC where B is an $m \times r$ nonnegative matrix of rank r and C is an $r \times n$ nonnegative matrix of rank r. Then $C(B)^* \subseteq C(A)^*$. Let $K = C(B)^* = C(F)$, where $F = [-V_B W_B]$. Note that $K = C(F) \subseteq C(A)^*$. Now consider the four conditions which must be met.

(1) From proposition 10 *(Bazaraa and Shetty)*, $C(A)^{**} = C(A)$. Hence

$$C(A) = C(A)^{**} \in K^* = \{y : F^T y \leq 0\} = C(F)^*.$$

From Minkowski's theorem, $C(F)^* = C([-V_F W_F])$. The first condition is proved.

(2) Again from Minkowski's theorem, $C(F)^* = C([-V_F W_F])$. Notice that $C(F)^* = K^* = C(B)^{**} = C(B)$. Therefore, $C([-V_F W_F]) = C(B)$. The nonnegative matrix C(B) is contained in $\mathbf{R}_m^+$. Hence $[-V_F W_F] \geq 0$.

(3) We already know that $C([-V_F W_F]) = C(B)$. Thus there exist nonnegative matrices D and E such that $[-V_F W_F] = BD$ and $B = [-V_F W_F]E$. Hence rank $[-V_F W_F]$ = rank B which is equal to the rank of A. It follows that rank $[-V_F W_F]$ = rank A.

(4) It has been shown that $[-V_F W_F] \geq 0$. The vector space $N(F^T) = C(W_F)$ must contain the additive inverse of all its component vectors. If $N(F^T) \neq \{0\}$, W_F would have to contain nonpositive vectors, which is not the case. So $N(F^T) = \{0\}$. Therefore Rank (F) = m. The columns of U_M (upto nonnegative scalar multiplication) are the extremal elements of the polyhedral cone obtained by the intersection of $F^T\mathbf{R}_m$ and $\mathbf{R}_k^+$. As stated in Lemma 13, F is an $m \times k$ matrix of rank m. The columns of V_F are the extremal elements of the polyhedral cone $C(V_F)$. This is proved in the following manner.

Let

$$V_F^{(1)} = (\sum \alpha_i V_F^{(i)}) + (\sum \beta_i V_F^{(i)}) = \sum(\alpha_i + \beta_i)V_F^{(i)}$$

where α_i and β_i are nonnegative and $V_F^{(i)}$ is column i in V_F. Since $U_F = F^T V_F$, each column in U_F has a proportional column in V_F. To see this, consider

$$U_F^{(1)} = F^T V_F^{(1)} = F^T \sum(\alpha_i + \beta_i)V_F^{(i)} = \sum(\alpha_i + \beta_i)U_F^{(i)}.$$

Therefore for each i,

$$(\alpha_i + \beta_i)U_F^{(i)} = \gamma_i U_F^{(1)}$$

for some nonnegative γ_i. Obviously,

$$(\alpha_i + \beta_i)U_F^{(i)} = (\alpha_i + \beta_i)F^T V_F^{(i)}$$

and $\gamma_i U_F^{(1)} = \gamma_i F^T V_F^{(1)}$. So $(\alpha_i+\beta_i)F^T V_F^{(i)} = \gamma_i F^T V_F^{(1)}$. Taking out the common factor, $F^T((\alpha_i + \beta_i)V_F^{(i)} - \gamma_i V_F^{(1)}) = 0$. Since the left null space of F is the zero vector, $(\alpha_i + \beta_i)V_F^{(i)} = \gamma_i V_F^{(1)}$ for all i. So the columns of V_F are the

elements of $C(V_F)$. But $C(B) = C([-V_F W_F])$. This is also equal to $C(-V_F)$ because $N(F^T) = \{0\}$. But the columns of B are up to nonegative scalar multiplication, the extremal elements of C(B). Hence the columns of $-V_F$ and B are proportional. Since B is an $m \times r$ matrix, it follows that $-V_F$ is $m \times r$. The matrix $W_F = \phi$, so $[-V_F W_F]$ is also $m \times r$.

Using this lemma, we can establish the main duality theorem.

Theorem 15 *A has nonnegative rank factorization if and only if there exists an $m \times k$ matrix F of rank m $(k > m)$ such that*

$(1')\{Fz : z \geq 0\} \subseteq C(A)^*$

$(2')V_F$ *is a positive $m \times r$ matrix of rank r.*

PROOF:

Assume that A has nonnegative rank factorization. From lemma 13, there always exists an $m \times k$ matrix of rank m having properties (1) through (4). The lemma assures us that there exists a special matrix having properties (1),...,(4). While proving lemma 13, we showed that $N(F^T) = C(W_F) = \{0\}$, that is, $W_F = \phi$, rank F = m, $-V_F = [-V_F W_F] \geq 0$, $[-V_F W_F]$ is $m \times r$ and that $rank(-V_F) = r$. Therefore (1') and (2') hold for F.

We shall prove that $k > m$ by contradiction. It is evident that k cannot be less than m. This leaves us with two possibilities, k=m or $k > m$. Assume that k=m. Then F is an $m \times m$ square matrix whose rank equals its row

rank and column rank. Then $F^T\mathbf{R}_m \cap \mathbf{R}_m^+ = \mathbf{R}_m \cap \mathbf{R}_m^+ = \mathbf{R}_m^+$. Let $U_F = I_M$ where I_m is the $m \times m$ identity matrix. The matrix I_m can be written in terms of V_F and F as $I_m = F^T V_F$. Multiplying both sides by $(F^T)^{-1}$, we have $(F^T)^{-1} I_m = (F^T)^{-1} F^T V_F$. Therefore $(F^T)^{-1} = V_F$. Since F has rank m, the inverse of its transpose also has rank m. This implies that V_F has rank m. This is a contradiction since rank $V_F = r$. Therefore $k \neq m$. We have exhausted all other possibilities, hence we can assert that $k > m$.

Finally, assume that there exists an $m \times k$ matrix F of rank m ($k > m$) such that conditions (1') and (2') are met. Conditions (2), (3) and (4) of the lemma follow readily. Condition (1) follows from the fact that $C(A) = C(A)^{**}$. Hence A has nonnegative rank factorization.

Considering the dual of the problem has given us a new theorem to work with. It seems reasonable to suppose that checking to see if $\{Fz : z \geq 0\} \subseteq C(A)^*$ is **NP**-Complete. This prompts us to pose another question:

> Is the problem of determining whether the polyhedral cone $\{Fz : z \geq 0\}$ is contained in the polar cone $C(A)^*$ **NP**-Complete?

So far we have not done much work on this hypothesis.

Our research efforts and the transformations we tried were described briefly in the previous chapter. We are currently concentrating on transforming the Clique problem in graph theory into problem (3). We hope to look at the duality result in the near future as it appears to be promising. Our work in this area has led us to believe that problems (1),

(2) and (3) may be **NP**-Complete, we are trying to prove the same.

This study lays the foundation for a serious investigation of the questions which have been raised concerning the computational complexity of certain problems in linear algebra. The resolution of one or more of these questions could serve as the foundation of more advanced research. Also, the successful resolution of one of these problems could open the door to a number of related problems in linear algebra. æ

APPENDIX 1

Strings, encoding schemes and languages

A **string** is an arrangement of symbols in a linear order. For any finite set of symbols $\sum$, we denote by $\sum^*$ the union of the set of all finite strings of symbols over $\sum$ and the empty string. For example, if $\sum = \{0,1\}$ then $\sum^*$ consists of an empty string "$\in$", the strings 0,1,00,01,10,11,000,001 ... and all other finite strings of 0's and 1's. A subset L, of $\sum^*$, is a **language** over the alphabet . So $\{0, 01, 111, 011101\}$ is a language over $\{0, 1\}$.

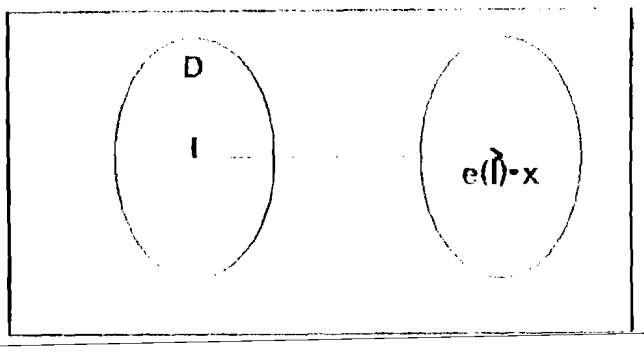
Encoding Schemes

An encoding scheme is a way of representing a problem for a machine; it is a link between the decision problem and the language. An **encoding scheme** e, is a function $e : D_\pi \longrightarrow \sum^*(I \in D_\pi, x = e(I))$. The function e is one-one. It may be noted that $e(D_\pi)$ is a subset of $\sum^*$. Moreover, $e(Y_\pi)$ is a subset of $\sum^*$ (since Y_π is a subset of D_π). Thus $e(Y_\pi)$ is a language which is denoted by $L[\pi, e]$.

$L[\pi, e] = \{x \in \sum^* : \sum$ *is the alphabet used by e and x is the encoding scheme under e of an instance* $I \in Y_\pi\}$

A pictorial representation of encoding schemes is given below.

Figure A1.1



We assume that one particular scheme has been chosen in advance and that each problem has an encoding scheme associated with it. For example, if we need to encode a graph we may use a vertex and edge list representation or an incidence matrix or an edge list. Descriptions of the graph G = (V,E) where $V = \{V_1, ..., V_4\}$ and E=$\{\{V_1, V_2\}, \{V_2, V_3\}$ under three different schemes are given in Table 1. Let the number of vertices be v and the number of edges be e. As seen in Table 2, the input lengths differ at most polynomially from each other.

Encoding Scheme	String	Length
Vertex List,Edge List	$v[1]v[2]v[3]v[4](v[1]v[2]v[2]v[3])$	36
Neighbour List	$(v[2])(v[1]v[3])(v[2])()$	24
Adjacency Matrix Rows	0100/1010/0010/0000	19

Table 1 (Garey and Johnson)

Encoding Scheme	Lower Bound	Upper Bound
Vertex List,Edge List	$4v + 10e$	$4v + 10e + (v + 2e).logv$
Neighbour List	$2v + 8e$	$2v + 8e + 2e.logv$
Adjacency Matrix	$v^2 + v - 1$	$v^2 + v - 1$

Table 2 (Garey and Johnson)

From the tables it can be noted that bounds on input lengths are not affected by the encoding scheme as long as "reasonable encoding schemes" are used. We define a reasonable encoding scheme as one which is polynomially equivalent to any other encoding scheme

which may be chosen. Encodings e_1 and e_2 of a problem are polynomially equivalent if given e_1, e_2 can be computed in polynomial time and vice versa. The input length for an instance I_π of a problem π is defined to be the number of symbols in the description of I_π obtained from the encoding scheme for π. Suppose $e(I_\pi) = X$, then the input length is length(X).

APPENDIX 2

TURING MACHINES

The physical apparatus on which the various computations described in this paper are carried out is called a Turing Machine. The Turing Machine (TM) is a hypothetical machine proposed by Alan Turing in 1936. It has an infinitely long tape with blank squares. There is a read-write head which can move bidirectionally on the tape. It can write symbols on the tape, change and delete symbols, functioning according to the instructions given.

A TM is formally defined as a 6-tuple $M = (Q, \sum, T, P, q_o, F)$ where

$Q =$ finite set of control states

$q_o =$ initial state

$T =$ alphabet of the program

$\sum = T \cup \in$ (tape alphabet with blank)

$F =$ final state, a subset of Q

$P = (Q \backslash F) \times \sum \longrightarrow Q \times \sum \times \{L, R, O\}$ a partial function where

L denotes "move one square to the left"

R denotes "move one square to the right"

O denotes "stay in the same square"

All the squares initially contain a blank. The input to a TM is a string $x \in \sum^*$. One symbol is placed in each square. The program starts at q_o, the initial state. The read-write head scans the first square and progresses square by square, either to the right or the left, depending on the instruction set. It may change the symbol to another symbol in the alphabet. The computation ends in the terminating states q_Y or q_N with an answer

of "Yes" or "No." A computation may never end. a task may be executed repeatedly without stopping, resulting in an **infinite loop.** Such an operation is highly undesirable. Generally, a loop is an operation involving repeated execution of a task. The TM described above is a **Deterministic Turing Machine** (to be henceforth called a **DTM**) because from each state it is only possible to go to another particular state specified by the program.

CHURCH'S THESIS : If a function is computable, it can be computed on a Turing Machine. If a function is Turing computable, it can be computed on another machine.

Church came to this conclusion based on a literature survey. Though this assertion has no formal proof there is little reason to doubt it. Church's thesis says nothing about time and space requirements, it is only concerned with computability. But the recent introduction of parallel machines raises the question – Are there functions which can be evaluated on parallel machines but are not Turing computable? The answer is No, as argued below. In other words, we are contending that if a parallel algorithm exists for a problem, then a sequential algorithm exists.

A parallel or concurrent algorithm specifies two or more sequential processes that may be executed simultaneously. (from Kronsjo)

The problem, for which we have a parallel algorithm P, is partitioned into subproblems $P_1, ..., P_n$. A parallel algorithm P assigns subproblems $P_1, ..., P_n$ to processors numbered $1, ..., n$. The assignment function 'f' is a bijection. So there exists some problem which is solved sequentially.

We may use one processor and implement the same algorithm. We compute P_1, store

the result R_1, compute P_2, store the result R_2 and so on. We now work with $R_1, R_2, ..., R_n$ and come up with the solution.

For every sequential algorithm, there is a TM. So there is a TM for every parallel algorithm. From this we learn that if a parallel algorithms exists for a problem, then a sequential algorithm exists. So Church's thesis is valid for parallel algorithms too.

At first glance it might seem incredible that so simple a machine as the TM can perform all the functions of a sophisticated computer. A computer is a powerful device because we can perform monotonous tasks quickly and without errors. Most computers today are sequential, that is, they perform a series of tasks one after the other. They can perform the same task repeatedly until a prespecified condition is met (this is called looping). They can also "choose" between different paths. For example, we can program a computer so that if the value of a variable x is greater than 100 it multiplies it by 2; if the value of x is less than 100, it divides by 2; if the value of x is 100 it does nothing. To put it concisely, a computer derives its power from its ability to perform

(1) Sequential operations - perform a set of tasks one after the other

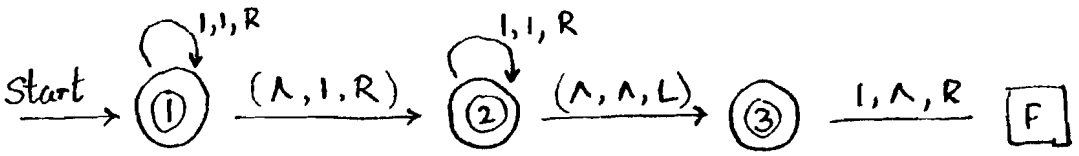
(2) Looping operations - perform the same task repeatedly

(3) Branching operations (decisions) - "choose" between two or more options

A TM may be represented by a directed graph. The nodes are denoted by circles and the final state by a square. The arrow indicates the transition from one state to another. Suppose the symbols $\wedge, \wedge, R$ appear on the arrow. This is read as "the read-write head reads a blank, writes a blank and moves right." The entire graph is read in this manner.

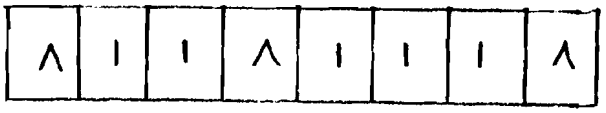
The following examples illustrate the ability of a Turing Machine to perform tasks (1), (2) and (3).

Example (A) The following TM adds two numbers $x, y \in N$ such that $z = x + y \in N$. It demonstrates sequential operations and looping.



Explanation: Let us assume that we wish to compute the sum of 2 and 3.

(1) At node 1, the tape looks like this.



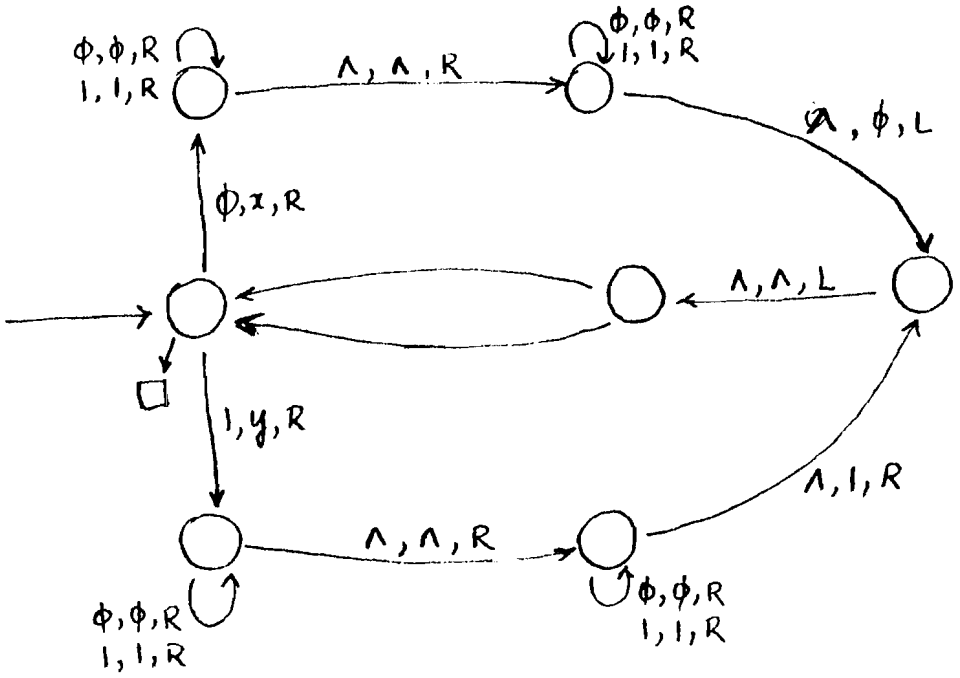
(i) As long as the read-write head sees the symbol 1, it replaces it with a 1 and moves right. The "curving arrow" with its head and tail at the same node denotes a loop.

(ii) When the read-write head sees a symbol $\wedge$ it replaces it with a 1 and moves right.

(2) As long as the read-write head sees the symbol 1, it replaces it with a 1 and moves right. When it sees a $\wedge$ (blank), it replaces it with a $\wedge$ (blank) and moves left.

(3) When the read-write head sees a 1, it replaces it with a $\wedge$ (blank) and moves right.

Example (B) This is an algorithm to make a copy of a string. it demonstrates branching. The graph may be read as described in the previous example.



NONDETERMINISM

When we use a DTM to solve a decision problem, we have a definite algorithm. Hence from each state it is possible only to go to the next predetermined state. On the other hand, a **Nondeterministic Turing machine (NDTM)** guesses a solution to the given problem and then checks the validity of the guessed solution. If it is valid, it is accepted, else another solution is guessed and so on. A nondeterministic Turing Machine (to be henceforth referred to as an NDTM) may be thought of as a TM with an attached guessing module. In such a machine, there is more than one state the machine can go to after a particular state. Hence when it is said that an NDTM "solves" a problem, it is to be noted that the word "solves" is being used in a very weak sense. It merely guesses a solution and then checks the answer.

A nondeterministic algorithm "solves" a decision problem π if the following hold for all instances $I \in D_\pi$

(i) If $I \in Y_\pi$, then there exists some guess S, that when guessed for input I will lead the checking stage to respond "Yes" for I and S

(ii) If I does not belong to Y_π, then there exist no guess S that when guessed for input I, will lead the checking stage to respond "Yes" for I and S.

APPENDIX 3

Some NP-Complete results

The first problem shown to be NP-Complete was the SATISFIABILITY problem which is as follows:

Instance: A set U of variables and a set C of clauses over U.

Question: Is there a satisfying truth assignment for C?

Each clause in C is satisfied if one of its components has a value of true. If all the clauses in C are satisfied, we have a satisfying truth assignment.

In 1971, Cook showed that Satisfiability is NP-Complete. An year later, Karp published twenty one NP-Complete results. Given below are six known NP-Complete problems (from Garey and Johnson) which are often used in proving that a given problem is NP-Complete.

3-SATISFIABILITY (3SAT):

Instance: A collection $C = \{c_1, ..., c_m\}$ of clauses on a finite set U of variables such that $|c_i| = 3$ for $1 \leq i \leq m$.

Question: Is there a truth assignment for U that satisfies all the clauses in C?

3-DIMENSIONAL MATCHING (3DM):

Instance: A set M which is a subset of $W \times X \times Y$, where W, X, Y are disjoint sets with the same number of elements, q.

Question: Does M contain a matching, that is, a subset $M'$of M such that $|M'| = q$ and no two elements of M' are the same in any coordinate?

VERTEX COVER (VC):

Instance: A graph $G = \{V, E\}$ and a positive integer $K \leq |V|$.

Question: Is there a vertex cover of size K or less for G, that is, a subset V' of V such that $|V'| \leq K$ and, for each edge $\{u, v\} \in E$, u or v or u and v belongs to V'?

CLIQUE:

Instance: A graph $G = \{V, E\}$ and a positive integer $J \leq |V|$.

Question: Does G contain a clique of size J or more, that is, is there V' a subset of V, such that $|V'| \geq J$ and every two vertices in V' are joined by an edge in E?

HAMILTON CIRCUIT (HC):

Instance: A graph $G = \{V, E\}$

Question: Does G contain a Hamilton circuit, that is, an ordering $< v_1, ..., v_n >$ of the vertices of G, where $n = |V|$, such that $\{v_n, v_1\} \in E$ and $\{v_i, v_{i+1}\} \in E$ for all $1 \leq i \leq n$?

PARTITION:

Instance: A finite set A and a "size" $s(a) \in Z^+$ for each $a \in$ A.

Question: Is there A', a subset of A such that $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$?

There are three basic techniques for proving that a problem is **NP**-Complete.

(1) Restriction: Given a problem π, we first show that $\pi \in NP$. We then show that π contains a known **NP**-Complete problem π' as a special case. If we are unable to solve a special case of π efficiently, then we cannot hope to solve π efficiently. Hence π is **NP**-Complete.

(2) Local Replacement: Given a problem π, we first show that $\pi \in NP$. We then

pick a problem π' which is known to be **NP**-Complete. We choose an instance of π' and look at it as a collection of basic units. We replace each basic unit in a uniform way with a different structure and obtain the problem π. We then show that this transformation can be done in polynomial time and the proof is complete.

(3) Component Design: Given a problem π, we first show that $\pi \in NP$. We then pick a problem π' which is known to be **NP**-Complete. The problem π' is seen as a collection of components, which are put together in a different way to yield π. We then show that this transformation can be done in polynomial time and the proof is complete. Sometimes a combination of two or more techniques may be used. Often it is not possible to draw a clear distinction between the last two techniques. æ

References

[1] M.S. Bazaraa and C.M. Shetty, *Foundations of Optimization*, Lecture Notes in Economics and Mathematical Systems, Springer Verlag, 1976.

[2] A. Ben-Israel and T.N.E Greville, *Generalized Inverses*, Wiley, 1974.

[3] R. Chandrasekaran, S.N. Kabadi and K.G. Murty, Some NP-Complete problems in Linear Programming, *Operations Research Letters*, Vol. 1, No. 3, pp.101-103, July 1982

[4] M.R. Garey and D.S. Johnson, *Computers and Intractability*, W.H. Freeman and Company, 1979

[5] M.W. Jeter and W.C. Pye, A Note on Nonnegative Rank Factorizations, *Linear algebra and its applications*, 38:171-73, 1981

[6] M.W. Jeter and W.C. Pye, Some Duality Theorems for Nonnegative Rank Factorization, *Industrial Mathematics*, pp. 63-71, Vol. 33, part 1, 1983

[7] O.L. Mangasarian, Characterization of real matrices of monotone kind, *SIAM Review* 10:439-441 (1968)

[8] K.G. Murty, *Linear Programming*, Wiley, 1983

[9] V.J. Rayward Smith, *A first course in computability*, Blackwell Scientific Press, 1989

[10] G. Strang, *Linear algebra and its applications*, Harcourt Brace Jovanovich, 1988