On the Approximability of Linear Ordering and Related NP-optimization Problems

Thesis submitted to the Indian Statistical Institute in partial fulfillment of the requirements for the award of the degree of Doctor of Philosophy

by

Sounaka Mishra

Theoretical Statistics and Mathematics Unit

Indian Statistical Institute

203 B. T. Road

Kolkata 700 108

INDIA

e-mail: res9513@isical.ac.in

under the supervision of

Professor Kripasindhu Sikdar

Theoretical Statistics and Mathematics Unit

Indian Statistical Institute

203 B. T. Road

Kolkata 700 108

INDIA

e-mail: sikdar@isical.ac.in

Acknowledgements

A number of people have influenced me in continuing my research and writing this thesis. First, I would like to thank all of them.

I express my sincere gratitude to my supervisor Professor Kripasindhu Sikdar for his guidance, advice, encouragement and constructive criticisms of my work. He introduced me to the area of approximation algorithms and allowed me to work on problems of my choice.

I express my regards to my teachers Professor R. Barua and Professor S. M. Srivastava for their encouragements. I also like to thank all the faculties of Stat-Math Unit of Indian Statistical Institute, Kolkata, for making an enjoyable environment to work.

I would like to thank my co-author Dr. M. Satpathy, University of Reading, UK, for introducing me to the Optimal Evaluation Ordering Problem and his encouragements. I would also like to thank Dr. P. Sarkar, Professor S. B. Rao and Professor A. R. Rao of Indian Statistical Institute, Kolkata, for their useful suggestions. My sincere thanks to Dr. C. R. Subramanian and Dr. M. Mohajan of I. M. Sc., Chennai, and Dr. J. Radhakrishnan of TIFR, Mumbai, for helpful discussions with them during the course of this work. I would also like to thank Dr. T. Fujito of Nagoya University, Japan, for having some discussion through email.

I express my appreciation to my friends Snigdhansu, Subho, Parasar, Biswa, Rudra-Da, Sarif, Lingaraj, Anita, Giri, Pradip and Pinakpani for their support and help. I must thank Mr. Ashok Naik for providing me several books from abroad.

I am thankful to Indian Statistical Institute for providing me its research infrastructure and supporting my research.

Finally, I would like to express my gratitude for the support I have received from all the members of my family.

Abstract

We investigate approximability of both maximum and minimum linear ordering problems (MAX-LOP and MIN-LOP) and several related problems such as the well known feedback set problems, acyclic subdigraph problem and several others and their variants.

We show that both MAX-LOP and MIN-LOP are strongly NP-complete, and MIN-LOP, MIN-QAP(S) (a special case of minimum quadratic assignment problem) and MIN-W-FAS are equivalent with respect to *strict*-reduction. The *strict*-equivalence is also established among these problems as well as MIN-W-FVS, with weights on arcs/vertices bounded by a polynomial, and the unweighted versions of the feedback set problems. We also show that MAX-LOP is *strict*-equivalent with MAX-W-SUBDAG and, with weights bounded by a polynomial, they are *AP*-equivalent with MAX-SUBDAG, the unweighted version of MAX-W-SUBDAG. Such equivalent problems have similar approximability properties. In particular, MIN-LOP and MIN-QAP(S) have $O(\log n \log\log n)$-approximation algorithms and they are APX-hard as MIN-W-FAS has such properties. Also MAX-LOP is APX-complete as MAX-W-SUBDAG is so and has a 2-approximation algorithm.

We next concentrate primarily on LOP, and, after noting that, if P$\neq$NP, there cannot exist any absolute approximation algorithm for LOP, we examine the question whether MIN-LOP$\in$APX. We have some evidences, though not strong, that appear to suggest that MIN-LOP and other equivalent problems may not be in APX, if P$\neq$NP. In particular, we establish a connection between coordinates of extreme points of linear ordering polytope and the existance of a constant-factor approximate solution of MIN-LOP which seems to suggests that we may not be able to get a constant-factor approximation algorithm for MIN-LOP via a LP-relaxation and rounding method. Secondly, we show that for MIN-Subset-FAS, a generalization of MIN-FAS, there cannot exist a polynomial-time $(1-\epsilon)\log n$-approximation algorithm, for any $\epsilon > 0$, unless NP$\subset$DTIME($n^{\log\log n}$). We also note that the proof of this result suggests a possible way of showing that MIN-LOP $\notin$ APX. We consider a generalization of MIN-LOP, called MIN-W-k-FAS, and show that it is NPO-complete. Since MIN-W-k-FAS is MIN-W-FAS with restriction just on cardinalities of the feasible solutions, this also suggests that MIN-W-FAS is possibly hard to approximate within a constant factor.

We then consider a restricted version of LOP, called $\triangle$-LOP, where the weight function satisfies a kind of triangle inequality (stronger than the usual triangle inequality)

and show that a simple heuristic, called CT-heuristic, yields good approximate solutions. In particular, the CT-heuristic is a $\frac{4}{3}$-approximation algorithm for $\triangle$-MAX-LOP and a $\frac{3}{2}$-approximation algorithm for $\triangle$-MIN-LOP. We extend quite easily the analysis of the performance of CT-heuristic for the case when the arc weights satisfy parametrized triangle inequality (or $\triangle_t$-inequality, for $t \in (0, 2]$), and show that the CT heuristic is a $\frac{2+t}{2t}$-approximation algorithm for $\triangle_t$-MIN-LOP and a $\frac{4}{2+t}$-approximation algorithm for $\triangle_t$-MAX-LOP. Further, $\triangle_t$-LOP$\in$PO if and only if $t = 2$, and $\triangle_t$-LOP, for $0 < t < 2$, is not in PTAS, unless P=NP.

Next we consider a problem called Optimal Evaluation Ordering Problem (OEOP) and prove some results regarding the complexity and approximability of OEOP by relating it to MIN-LOP. We establish NP-completeness of OEOP by showing that a restricted version of OEOP, called OEOP(S), is NP-complete. OEOP(S) is obtained from OEOP by restricting the instances which satisfy an assumption, called simple sharing assumption. We prove NP-completeness of OEOP(S) by a reduction from the NP-complete problem MIN-LOP(1, 2), which is a restricted version of MIN-LOP with arc weights being 1 or 2. Regarding approximability, we show that (1) OEOP(S) $\leq_{strict}$ MIN-LOP, (2) MIN-LOP(1, 2) $\leq_{strict}$ OEOP(S) and (3) OEOP $\leq_{strict}$ MIN-LOP(set), where MIN-LOP(set) is a generalization of MIN-LOP. From these we conclude that (1) OEOP(S) $\notin$ PTAS, if P$\neq$NP, using a similar result about MIN-LOP(1, 2), and (2) OEOP(S) has an $O(\log n \log\log n)$-approximation algorithm as MIN-LOP has such an algorithm. We feel that approximating OEOP is harder than MIN-LOP, but has at least similar approximability properties as that of MIN-LOP(set). On the other hand as MIN-LOP is believed not to be in APX, it is expected that OEOP may not be in the class APX, if P$\neq$NP. We then propose some heuristics for OEOP.

We also establish various results about hardness of approximating several minimum-maximal and maximum-minimal NP-complete optimization problems on graphs as well as related maximum/minimum problems. We are led to investigation of such problems while considering a generalization of MIN-LOP, viz MIN-W-MAX-SUBDAG and its unweighted version MIN-MAX-SUBDAG problem. As MIN-LOP is APX-hard, MIN-W-MAX-SUBDAG is so. However, even though the unweighted version of MIN-LOP can be solved in polynomial time, we show that MIN-MAX-SUBDAG is APX-hard. We also show that the complementary problem MAX-MIN-FAS is also APX-hard. We establish these results by L-reductions from MAX-SUBDAG problem which is APX-complete.

Then, using the results of Håstad concerning hardness of approximating MAX-IS and appropriate reductions, we prove similar results about MAX-MIN-VC, for arbitrary graphs, and about MAX-MIN-FVS, for arbitrary graphs and digraphs. In particular, we show that, unless NP = ZPP (respectively, P=NP), for any $\epsilon > 0$, there exists no polynomial-time algorithm to approximate MAX-MIN-VC within a factor of $\frac{1}{2\sqrt{2}} n^{\frac{1}{2}-\epsilon}$ (respectively, $\frac{1}{2\sqrt[4]{2}} n^{\frac{1}{4}-\epsilon}$), where n is the number of vertices in an instance. For MAX-MIN-FVS we have, unless NP=ZPP (respectively, P=NP), for any $\epsilon > 0$, there exists no polynomial-time algorithm to approximate MAX-MIN-FVS within a factor of $\frac{1}{4} n^{\frac{1}{2}-\epsilon}$ (respectively, $\frac{1}{2\sqrt{2}} n^{\frac{1}{4}-\epsilon}$), where n is the number of vertices in an instance. For undirected graphs, we have similar results for MAX-MIN-FVS.

Finally, we consider approximability of some of these problems restricted to bounded degree graphs and digraphs. By considering bounded degree graphs and digraphs, we show that MIN-FVS is APX-complete for 6-regular graphs and MAX-MIN-FVS is APX-hard for all graphs of maximum degree 9. Then, we prove that MAX-MIN-VC is k-approximable for all graphs without any isolated vertex and having maximum degree k, $k \geq 1$, and is APX-complete for all graphs of maximum degree 5. We also establish APX-hardness of MIN-FAS and MAX-SUBDAG for k-total-regular digraphs, for all $k \geq 4$, MIN-MAX-SUBDAG for digraphs of maximum total degree 12 and MAX-MIN-FVS for all digraphs of maximum total degree 6.

Contents

List of Notations and Symbols

Symbol	Brief explanation of symbol
Numbers:	
$\mathbb{N}$	the set of non-negative integers
$\mathbb{R}$	the set of real numbers
Sets:	
$\subset$	proper subset
$\subseteq$	subset
$\cap$	intersection
$\cup$	union
2^A	the set of all subsets of A
$\|A\|$	the cardinality of set A
Asymptotic Notations:	
$O(f(n))$	class of functions asymptotically dominated by $f(n)$
$\Omega(f(n))$	class of functions which asymptotically dominate $f(n)$
Graphs:	
$G = (V, E)$	graph with vertex set V and edge set E
$G = (V, A)$	digraph with vertex set V and arc set A
$G_n = (V, A_n)$	complete digraph with n vertices
$N(v)$	open neighborhood of vertex $v \in V$
$N[v]$	closed neighborhood of vertex $v \in V$
SUBDAG	directed acyclic subdigraph
FAS	feedback arc set
FVS	feedback vertex set
VC	vertex cover
IS	independent set
Optimization problems :	
π	optimization problem $\pi = (I, sol, m, goal)$

π^p	restriction of π to instances of x such that the sum of weights is at most $p(\|x\|)$, where p is a polynomial
π_d	decision version of π
L_π	language associated with the optimization problem π and corresponds to π_d
I	set of instances of π
$\|x\|$	size of $x \in I$
$max(x)$	largest number appearing in $x \in I$
$sol(x)$	set of feasible solutions for the instance x
$m(x, y)$	measure of $y \in sol(x)$
$m^*(x)$	value of an optimal solution for x
$opt(x)$	set of optimal solutions for x

Reductions:

$\leq_m^p$	polynomial-time many-one reduction
$\leq_{AP}$	AP-reduction
$\leq_{strict}$	*strict*-reduction
$\leq_L$	L-reduction
$\equiv_r$	equivalence with respect to the r-reduction

Linear ordering problem

LOP	linear ordering problem
$\triangle$-LOP	LOP satisfying a triangle inequality
$\triangle_t$-LOP	LOP satisfying a parametrized triangle inequality
P_{LO}^n	integral acyclic tournament polytope
P_c^n	generalized acyclic tournament polytope

Symbols related to logic

$\wedge$	logical conjunction operator
$\vee$	logical disjunction operator
$\overline{\sigma}$	negation of a literal

Chapter 1

Introduction

How well can we solve an optimization problem? Can we obtain an *optimal* solution, or at least a *good approximate* solution, for such a problem using a *reasonable* amount of computation time or space? If not, can we provide some qualitative explanation as to why we cannot possibly find such solutions? These are the types of questions we investigate in this thesis regarding some specific combinatorial optimization problems such as linear ordering and related problems.

The purpose of this chapter is to give a short informal introduction to the developments in response to above questions concerning combinatorial optimization problems and then an outline of the specific contributions made in this thesis as well as the organization of the rest of the thesis.

1.1 Efficient Algorithms and Combinatorial Optimization Problems

A central concern in Computer Science is to design *efficient* algorithms for solving various computational problems. Following Edmonds [51], an algorithm for a problem is regarded as efficient if, for any instance of the problem, the running time of the algorithm is bounded by a polynomial in the *size* of the instance.

A combinatorial optimization problem is specified by a set of instances such that for each instance there is a finite set of feasible solutions and for each feasible solution of an instance there is a positive integer cost. The goal is to obtain, for

any instance, a feasible solution that has optimal (i.e. maximum or minimum) cost [131,2,9,148,119]. Examples of such problems (see [69,9]) include the well known optimization problems on graphs and networks such as minimum spanning tree (MST), maximum flow (MAX-FLOW), maximum matching (MAX-MATCH), traveling salesman (MIN-TSP), maximum independent set (MAX-IS), minimum vertex cover (MIN-VC) problems and many other problems arising in diverse contexts such as 0-1 knapsack (01-KNAP), scheduling independent tasks (SIT) and others.

During the past few decades there have been significant developments in designing efficient algorithms and understanding of computational complexity of such problems. While very efficient algorithms have been designed for some of these problems [138,108, 60,93,54] such as MST, MAX-FLOW, MAX-MATCH, etc, no efficient algorithm has been discovered, despite intensive investigation by many researchers, for many other problems such as MIN-TSP, MAX-IS, MIN-VC, 01-KNAP, SIT, etc.

We are concerned with computationally hard problems that are of the second kind. For such a problem, as the set of feasible solutions for any instance, though finite, is very large and has size that grows exponentially (or still worse) with the size of the instance, the straight forward (or naive) algorithm, based on exhaustive search of the feasible set, is very inefficient. For example, for an n-city MIN-TSP instance, the naive algorithm that examines all the possible tours to find a minimum cost tour has time complexity $O(n!)$. As the factorial function grows very very fast as n becomes large, much faster than any exponential function of n, even for small value of n, such as $n = 50$, such an algorithm cannot be executed within any reasonable amount of time. Similarly, for the maximum linear ordering problem (MAX-LOP), for any instance, that consists of a complete digraph on a set V of n vertices with positive integer costs associated with the arcs and the objective being to find a minimum cost acyclic tournament on V, the set of feasible solutions has $n!$ elements as each such tournament is specified by a permutation of the vertices.

However, using clever algorithm design principles, often algorithms with much

improved running time have been designed for many such problems. For example, Held and Karp [91] designed an algorithm with time complexity $O(n^2 2^n)$ for n-city MIN-TSP based on the dynamic programming principle. While this is much better than the naive algorithm, no considerably better exact algorithm is known for this problem. Similarly, one can design a dynamic programming algorithm for linear ordering problem. As another example, we may mention that, using a divide and conquer strategy, a recursive algorithm for MAX-IS has been designed by Tarjan and Trojanowski [149] with time complexity, $O(2^{\frac{n}{3}})$, which is, of course, much better than the complexity $O(2^n)$ of the naive algorithm. But these reflects the worst-case time complexity and algorithms based on techniques such as branch and bound, cutting plane methods, etc, have been found to solve such hard problems of reasonable size arising in practice within affordable time [111, 77]. However, the urge for providing satisfactory explanation of their intrinsic hardness and possible intractability, in the sense of possible nonexistence of algorithms of polynomial worst-case time complexity, has been a primary motivation for the development of the theory of computational complexity, specially the theory of NP-completeness.

1.2 NP-completeness and NP-hard Optimization Problems

The possible intractability of combinatorial optimization problems like MIN-TSP, MAX-IS, 01-KNAP, etc, is best explained by the theory of NP-completeness developed by Cook [40], Levin [110] and Karp [103]. Though their theory explicitly deals with decision problems, i.e., problems which require to answer 'yes' or 'no' to a question asked about any instance, its implication easily extends to optimization problems as well.

The essence of their theory is quite simple. Let P denote the class of all decision problems which can be solved by polynomial-time algorithms. Let NP denote the class of all decision problems which can be solved by polynomial-time non-deterministic algorithms. Deciding that an instance x is an 'yes' instance of a decision problem in

polynomial time by a nondeterministic algorithm can be thought of being composed of two steps: (i) first the algorithm guesses a *certificate* y of x being an 'yes' instance among potentially exponential many possible solutions, and (ii) then verifies in polynomial time that y is indeed a certificate for x. Clearly, P$\subseteq$NP, but whether P=NP is the central open question in computer science. However, the general belief is that P$\neq$NP. A decision problem π is NP-hard, if every problem in NP can be solved in polynomial time by *reducing* it to π. An NP-hard problem that is also in NP, is an NP-complete problem. It follows that if, P$\neq$NP, then no NP-hard problem can be solved in polynomial time.

This important negative conclusion about polynomial-time solvability of a NP-hard decision problem can be extended to optimization problems by noting that a decision problem can be naturally associated to any optimization problem. For an optimization problem π we can associate a decision problem π_d, which given an instance x of π and a bound k, requires to find whether x has a solution of cost $\geq k$ (or $\leq k$) if π is a maximization (or minimization) problem.

An optimization problem is an NP-optimization problem if its associated decision problem is in NP. An optimization problem π is NP-hard if every decision problem in NP can be solved in polynomial time using an algorithm that solves π. An NP-optimization problem is NP-complete if the associated decision problem is NP-complete. The class of all NP-optimization problems is denoted by NPO. If we denote by PO, the class of NP-optimization problems which can be solved in polynomial time, then it can be easily seen that P$\neq$NP implies that PO$\neq$NPO. If P$\neq$NP, no NP-hard optimization problem can be solved in polynomial time.

1.3 Approximation Algorithms for NP-hard Optimization Problems

Most of the NP-hard optimization problems are of considerable practical interest, and, therefore, researchers have attempted different possible approaches to find efficient

'practical' solutions to such problems. One of the approaches is to find efficient algorithms for interesting subclasses of any such problems. Many interesting results have been obtained in this direction, for example, the efficient algorithms for MAX-IS, MIN-COLOR, etc for perfect graphs [70,74]. Another important direction is to find good approximate solutions, i.e. solutions that are 'close' to optimal solution, by efficient algorithms. The field of approximation algorithms initiated by Garey, Graham and Ullman [66] and Johnson [96], is now well developed [90,9], though approximation algorithms for some scheduling problems were designed by Graham [75] even before the formulation of the theory of NP-completeness. Several paradigms for designing good approximation algorithms have been developed which include apart from greedy and dynamic programming paradigms, randomized methods and associated derandomization (see Chapter 11 in [90]), primal-dual method on relaxed linear program formulation (see Chapter 4 in [90]) and algorithms based on semidefinite relaxation [73].

An interesting thing to note is that while all known NP-complete problems are structurally similar (or *polynomial-time isomorphic*) and all NP-complete problems are conjectured to be so (the Berman-Hartmanis Conjecture) [22], the NP-hard optimization problems exhibit a great variations with respect to approximability by efficient algorithms.

Various approximability properties of NP-optimization problems correspond to various classes of approximation algorithms with respect to a natural measure of approximation quality. A common measure of approximation quality is the performance ratio [9] of the cost of an approximate solution and the cost of an optimal solution. For a function $f : \mathbb{N} \to [1, \infty)$, an $f(n)$-approximation algorithm for a problem is a polynomial-time algorithm obtaining a solution, for any instance x, with performance ratio bounded by $f(|x|)$. For a class F of such functions, F-APX denotes the class of all NP-optimization problems having a polynomial-time $f(n)$-approximation algorithm where $f(n) \in F$. For example, APX, log-APX, poly-APX, exp-APX refer to classes where $F = O(1), O(\log n), O(\text{poly}(n)), O(\exp(n))$, respectively. Several problems are

known to be in APX such as MAX-CUT, MIN-VC [132], whereas many problems such as MAX-IS, MIN-TSP are known not to be in APX. Of course, all these problems restricted to appropriate subclasses of instances are even in PO, and MIN-TSP with symmetric cost matrix satisfying triangle inequality is in APX [38].

Efforts have been made to obtain approximate solutions with performance ratio bounded by any arbitrary constant $r \geq 1$. An approximation algorithm which, for any instance x and for any $r \geq 1$, yields solution with performance ratio bounded by r and having time complexity bounded by a polynomial in the size of x is called a *polynomial-time approximation scheme* (ptas). The class of all NP-optimization problems having a ptas is denoted by PTAS. Very few problems have been shown to be in PTAS. A drawback of a ptas is that as the approximate solution gets closer to an optimal solution, i.e., r becomes closer to 1, the time complexity of the algorithm, which may depend exponentially on $\frac{1}{r-1}$, though polynomial in size of x, may blow up making it impractical. So, it may be desirable to have a *fully polynomial-time approximation scheme* (fptas) which is a ptas with running time bounded by a polynomial in both the size of the instance and $\frac{1}{r-1}$. The class of NP-optimization problems that admit of an fptas is denoted by FPTAS. The 01-KNAP is in FPTAS [94] while SIT is in PTAS [75, 119]. Clearly, PO $\subseteq$ FPTAS $\subseteq$ PTAS $\subseteq$ APX $\subseteq$ log-APX $\subseteq$ poly-APX $\subseteq$ exp-APX $\subseteq$ NPO, and it is known that this is a strict hierarchy if P$\neq$NP [42,130,28,9]. The *central concern* is to locate any given NP-optimization problem to an approximation class as low as possible in the above hierarchy.

1.4 Characterization of Approximation Classes, Reductions and Hardness of Approximation

In response to the above concern regarding NP-optimization problems, several alternative approaches can be made. If an appropriate characterization of an approximation class $\mathcal{C}$ can be obtained by giving necessary and sufficient conditions for an NP-optimization problem to be in $\mathcal{C}$, then it may be possible to verify efficiently whether a

problem satisfies such conditions and belongs to $\mathcal{C}$. But very few results of this type are known. Korte and Schrader [106] obtained some necessary and sufficient algorithmic conditions for certain problems to be in PTAS (and in FPTAS). But Paz and Moran [134] obtained characterization of the classes PTAS and FPTAS in terms of certain combinatorial and algorithmic conditions. Such characterizations of other approximation classes, such as APX, are not known. However, to the best of our knowledge, no application of such characterization is known. But we shall show an application of PAZ and Moran's result about PTAS [134] in Chapter 5 showing that a restricted version of linear ordering problem cannot be in PTAS, if P$\neq$NP.

A characterization of the class APX of a quite different flavor has, however, been obtained. In [132], Papadimitriou and Yannakakis defined syntactically, in terms of second order logic, a class MAX-SNP of NP-optimization problems, inspired by the work of Fagin [55] and Kolaitis and Vardi [104], and showed that MAX-SNP $\subseteq$ APX. But, after a series of interesting developments in interactive proof systems culminating to the PCP characterization of NP (see [57,7,6,87]), finally, Khanna, Motwani, Sudan and Vazirani [102] established that APX=MAX-SNP. Such syntactic characterization for other approximation classes are not known though some progress towards such a characterization of PTAS has been made by Khanna and Motwani [101].

In order to show that an NP-optimization problem belongs to some approximation class, we may design an approximation algorithm having the required approximation properties. While a good number of such positive results are known (see [90]), researchers have also tried to establish negative results in showing that certain problems cannot be in some approximation class, if P$\neq$NP or similar complexity theoretic condition holds.

Perhaps the first such result establishing hardness of approximating an NP-hard optimization problem with desired approximation property was obtained by Garey and Johnson [68]. They formulated the concept of strongly NP-complete problem and showed that no strongly NP-complete optimization problem with some easily verifiable

condition can belong to FPTAS, if P≠NP The next important result was due to Sahni and Gongalez [141] who showed that MIN-TSP cannot be in the class APX, if P≠NP, by using a 'gap' producing reduction from the NP-complete Hamiltonian cycle problem. However, such a technique has been of very limited use.

Researchers have also investigated various approximation preserving reductions and complete problems for approximation classes in a spirit similar to the theory of NP-completeness (for a list of definitions of these reductions see [98,9]). In particular, under appropriate approximation preserving reductions, Orponen and Manila [128] proved NPO-completeness of MIN-W-3SAT, MIN-TSP and MIN-01-PROGRAMMING and Cre-scenzi and Panconesi [42] proved APX-completeness and PTAS-completeness of several problems such as Bounded-SAT is APX-complete and Lower-bounded-SAT is PTAS-complete. Also Berman and Schnitger [23] considered a class of polynomially bounded NP-optimization problems (NPO-PB), and established that certain natural problems, are NPO-PB-complete under an approximation preserving reduction. They also showed that, if P≠NP, there exists an $\epsilon > 0$ such that, any NPO-PB-complete cannot be approximated within a factor of $O(n^{\epsilon})$. In the same direction, Kann [99,100] and Crescenzi, Kann, Silvestri and Trevisan [41] found other NPO-PB-complete problems such as Shortest Computation, Shortest Path with Forbidden Pairs, Independent Dominating set, MIN-Ones, MAX-Ones and many other problems.

Papadimitriou and Yannakakis [132], on the other hand, established that some natural problems such as MAX-3SAT, MAX-CUT, MIN-VC, MAX-IS, MIN-$\triangle$TSP are MAX-SNP-complete with respect to L-reduction, and showed that a MAX-SNP-complete problem is in PTAS if and only if MAX-SNP $\subseteq$ PTAS. Further, with successive developments in the interactive proof systems [57,7,5] leading to the PCP characterization of NP, Arora, Lund, Motwani, Sudan and Szegedy [5] finally showed that no MAX-SNP-complete problem can be in PTAS, unless P=NP. With the identification of APX with MAX-SNP [5], as mentioned earlier, this shows that no APX-complete problem can be in PTAS, unless P=NP.

PCP characterization of NP and subsequent developments led to many other important results. For many years it was not known whether MAX-CLIQUE (or MAX-IS) is in APX. Though Feige, Goldwasser, Lovász, Safra and Szegedy [57] provided first evidence that MAX-CLIQUE is not in APX, Arora and Safra [7] finally proved that if MAX-CLIQUE is in APX, then P=NP. There is a series of important papers regarding hardness of approximating problems like MAX-CLIQUE, MAX-IS, etc, (see [19,58,20]), but perhaps the strongest result about MAX-CLIQUE and MAX-IS is due to Håstad [87] who showed that these problems cannot be approximated within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$, under an appropriate complexity theoretic hypothesis. Such hardness results for other problems can be established by use of appropriate approximation preserving reductions from known hard problems, but such reductions can often be quite nontrivial, as we shall see in this thesis.

1.5 Contributions in This Thesis

In this thesis, we are concerned with approximate solutions and hardness of approximation of linear ordering and related NP-optimization problems. A detailed discussion about these problems, their applications and review of known results about their complexity and approximability together with their relationships are presented in Chapter 3. Our main contributions consists of (1) a few relatively simple positive results having efficient approximation algorithms for some of the problems with certain approximation qualities, and (2) many negative results describing hardness of approximating many of the problems using the standard reducibility arguments, some of which are quite non-trivial. However, before we describe our results, in Section 1.5.2 through Section 1.5.5, we explain briefly what these problems are in Section 1.5.1.

1.5.1 The Problems Considered

Given a complete digraph $G_n = (V, A_n)$ on a set $V = \{1, 2, \ldots n\}$ of n vertices with non-negative integer arc weights, the maximum (respectively, minimum) linear order-

ing problem (MAX-LOP) (respectively, MIN-LOP) is to find an acyclic (i.e. without any dicycle) tournament on V [83] of maximum (respectively, minimum) total arc weight. We shall write LOP when referring to either of MAX-LOP and MIN-LOP. A problem arising in the context of implementation of eager functional languages [137, 120], which also provides some motivation for considering the MIN-LOP problem, is the Optimal Evaluation Ordering Problem (OEOP). In essence, the OEOP requires to find an ordering for evaluation of a set of arithmetic expressions having common subexpressions so as to minimize the number of values of common subexpressions to be stored in memory for the evaluation of other expressions. Two well known optimization problems on digraphs related to LOP are the maximum weight acyclic subdigraph (MAX-W-SUBDAG) and the minimum weight feedback arc set (MIN-W-FAS) problems. Given an arc weighted digraph $G = (V, A)$, the MAX-W-SUBDAG problem is to find a subset $B \subseteq A$ of maximum weight such that (V, B) is an acyclic subdigraph of G, and the MIN-W-FAS problem is to find a subset $F \subseteq A$ of minimum weight such that $(V, A - F)$ is an acyclic subdigraph of G. The unweighted versions (i.e. when arc weights are all 1) of these problems are denoted by MAX-SUBDAG and MIN-FAS, respectively. A vertex version corresponding to MIN-W-FAS is the minimum weight feedback vertex set (MIN-W-FVS) problem which, for a given vertex weighted digraph (or graph) $G = (V, A)$ (or $G = (V, E)$), requires to find a minimum weight subset $F \subseteq V$ such that F is a feedback vertex set of G, i.e. F contains at least one vertex from each dicycle (or cycle) of G. The unweighted version of this problem is denoted by MIN-FVS. A generalized version of MIN-W-FAS is denoted by MIN-Subset-FAS which, for any arc weighted digraph $G = (V, A)$ and a subset $\mathcal{C}$ of interesting dicycles, requires to find a minimum weight FAS F that destroys all dicycles in $\mathcal{C}$. Similarly, the generalized version MIN-Subset-FVS of MIN-W-FVS is defined.

We consider two problems, which can be viewed as generalizations of MIN-LOP, by noting that, for any complete digraph $G_n = (V, A_n)$, an acyclic tournament on V is indeed a maximal SUBDAG (i.e. a SUBDAG of G_n not contained in any other

SUBDAG of G_n) as well as a minimal FAS of G_n (i.e. a FAS of G_n not containing any other FAS of G_n) and has $\frac{1}{2}n(n-1)$ arcs. Thus, one generalization of MIN-LOP is the minimum weight maximal SUBDAG (MIN-W-MAX-SUBDAG) problem that requires to find a maximal SUBDAG of minimum total arc weight in any given arc weighted digraph (which is not necessarily a complete digraph). The complementary problem corresponding to MIN-W-MAX-SUBDAG is the maximum weight minimal feedback arc set (MAX-W-MIN-FAS) problem which requires to find a minimal FAS of maximum weight in a given arc weighted digraph. Similarly, the MAX-W-MIN-FVS problem is defined. Another related problem that we consider is the MAX-W-MIN-VC which requires to find, given a vertex weighted graph, a minimal VC of maximum weight. The unweighted versions of these problems are denoted by the same symbols without 'W' such as MIN-MAX-SUBDAG. As another generalization of MIN-LOP, we consider the MIN-W-k-FAS problem, which, for a given an arc weighted digraph $G = (V, A)$ and a positive integer k, it is required to find a minimum weight FAS F of G with $|F| = k$.

We consider some of these weighted problems with restriction on the weight function. For an NP-optimization problem π and a polynomial p, we denote by π^p the restriction of π to instances of x such that the sum of weights is at most $p(|x|)$. We also consider many of these problems on graphs (or digraphs) with restriction on the vertex degrees. In general, given any optimization problem π on a graph we denote by π-$\leq k$ (respectively, π-$= k$) as the problem π on graphs whose vertices have degree at most (respectively, equal to) k. Similar, notations are used for problems on digraphs where the restriction is put on total degrees (indegree plus outdegree) of the vertices.

1.5.2 Preliminary Observations on the Complexity and Approximability of the Problems

In Chapter 3, in addition to detailed discussion about above problems and their applications, we present a quick overview of the known results about their complexity and

approximability. Further, we make some simple observations regarding complexity and approximability of these problems and establish equivalences among several of them with respect to *strict*-reduction or AP-reduction.

In particular, we show that LOP(p, q), a restricted version of LOP with arc weights on input digraph restricted to the integers p and q, is NP-complete. Thus, LOP is strongly NP-complete and is easily seen not to be in FPTAS, if P$\neq$NP, by the result of Garey and Johnson [68].

Then we show that MIN-LOP$\equiv_{strict}$MIN-W-FAS$\equiv_{strict}$MIN-QAP(S). We also show similar equivalences for several weighted problems with polynomial restrictions on weight function and their unweighted versions. In particular, MIN-LOP^p, MIN-W-FAS^p and MIN-FAS are *strict*-equivalent and MAX-LOP^p, MAX-W-SUBDAG^p and MAX-SUBDAG are AP-equivalent. Similar equivalences also hold for MIN-W-Subset-FAS^p, MIN-W-Subset-FVS^p, MIN-Subset-FVS and MIN-Subset-FAS problems. Because of such equivalences, these problems have similar approximability properties. In particular, MIN-LOP admits of an $O(\log n \log\log n)$-approximation algorithm as such an algorithm exists for MIN-W-FAS problem [53]. These observations are contained in the joint work [121] with K. Sikdar.

1.5.3 On the Approximability of LOP

In view of the equivalences of MIN-LOP and MAX-LOP with several other problems we next confine our attention primarily to LOP and examine various issues regarding its approximability, the main issue being whether MIN-LOP $\in$ APX.

After noting that the complexity of an exact algorithm for LOP, based on dynamic programming, is $O(n^2 2^n)$ and that, if P$\neq$NP, no polynomial-time absolute approximation algorithm for LOP can exist, we address the question whether MIN-LOP $\in$ APX. While all our efforts to show that MIN-LOP $\in$ APX have failed, we have some evidences, though not strong, that appear to suggest that MIN-LOP and other equivalent problems may not be in APX, if P$\neq$NP. In particular, we show that, if non-zero

coordinates of the extreme points of the linear ordering polytope are all greater than $\frac{1}{k}$, for some constant $k > 1$, then by rounding an optimal solution of an LP-relaxation of MIN-LOP, we get a k-approximate solution of MIN-LOP. But, as Nutov and Penn [127] have shown, the non-zero coordinates can be arbitrarily close to 0 and this leads us to believe that we may not be able to get a constant-factor approximation algorithm for MIN-LOP via a LP-relaxation and rounding method. Secondly, we show that for MIN-Subset-FAS, a generalization of MIN-FAS, there cannot exist a polynomial-time $(1-\epsilon)\log n$-approximation algorithm, for any $\epsilon > 0$, unless $NP \subset DTIME(n^{\log\log n})$. We also note that the proof of this result suggests a possible way of showing that MIN-LOP $\notin$ APX. We also consider a generalization of MIN-LOP, called MIN-W-k-FAS, and show that it is NPO-complete. Since MIN-W-k-FAS is MIN-W-FAS with restriction just on cardinalities of the feasible solutions, this also suggests that MIN-W-FAS is possibly hard to approximate within a constant factor.

Then, we consider a restricted version of LOP, called $\triangle$-LOP, where the weight function satisfies a kind of triangle inequality, that is stronger than the usual triangle inequality, and show that a simple heuristic, as suggested in [132] for MAX-SUBDAG, adapted for $\triangle$-LOP, called CT-heuristic, yields much better approximate solution. In particular, the CT-heuristic is a $\frac{4}{3}$-approximation algorithm for $\triangle$-MAX-LOP and a $\frac{3}{2}$-approximation algorithm for $\triangle$-MIN-LOP. We extend quite easily the analysis of the performance of CT-heuristic for the case when the arc weights satisfy parametrized triangle inequality (or $\triangle_t$-inequality, for $t \in (0, 2]$), and show that the CT heuristic is a $\frac{2+t}{2t}$-approximation algorithm for $\triangle_t$-MIN-LOP and a $\frac{4}{2+t}$-approximation algorithm for $\triangle_t$-MAX-LOP. Further, $\triangle_t$-LOP$\in$PO if and only if $t = 2$. Finally, we show that $\triangle_t$-LOP, for $0 < t < 2$, is not in PTAS, unless P=NP. These results are taken from a joint work [121] with K. Sikdar.

1.5.4 On the Complexity and Approximability of OEOP

We then prove some results regarding the complexity and approximability of OEOP by relating it to MIN-LOP. We establish NP-completeness of OEOP by showing that a restricted version of OEOP, called OEOP(S), is NP-complete. OEOP(S) is obtained from OEOP by restricting the instances which satisfy an assumption, called simple sharing assumption. We prove NP-completeness of OEOP(S) by a reduction from the NP-complete problem MIN-LOP(1, 2). Regarding approximability, we show that (1) OEOP(S) $\leq_{strict}$ MIN-LOP, (2) MIN-LOP(1, 2) $\leq_{strict}$ OEOP(S) and (3) OEOP $\leq_{strict}$ MIN-LOP(set), where MIN-LOP(set) is a generalization of MIN-LOP. From these we conclude that (1) OEOP(S) $\notin$ PTAS, if P$\neq$NP. using a similar result about MIN-LOP(1, 2), and (2) OEOP(S) has an $O(\log n \log\log n)$-approximation algorithm as MIN-LOP has such an algorithm. We feel that approximating OEOP is harder than MIN-LOP, but has at least similar approximability properties as that of MIN-LOP(set). On the other hand as MIN-LOP is believed not to be in APX, it is expected that OEOP may not be in the class APX, if P$\neq$NP. Finally, we propose some heuristics for OEOP. These results are from the joint work [120] with K. Sikdar and M. Satpathy.

1.5.5 Hardness of Approximating Some NPO Problems Related to MIN-LOP

We also establish various results about hardness of approximating several minimum-maximal and maximum-minimal NP-complete optimization problems on graphs as well as related maximum/minimum problems. We are led to investigation of such problems while considering a generalization of MIN-LOP, viz MIN-W-MAX-SUBDAG and its unweighted version MIN-MAX-SUBDAG problem. As MIN-LOP is APX-hard, MIN-W-MAX-SUBDAG is so. However, even though the unweighted version of MIN-LOP can be solved in polynomial time, we show that MIN-MAX-SUBDAG is APX-hard. We also show that the complementary problem MAX-MIN-FAS is also APX-hard. We

establish these results by L-reductions from MAX-SUBDAG problem which is APX-complete.

Then, using the results of Håstad concerning hardness of approximating MAX-IS and appropriate reductions, we prove similar results about MAX-MIN-VC, for arbitrary graphs, and about MAX-MIN-FVS, for arbitrary graphs and digraphs. In particular, we show that, unless NP=ZPP (respectively, P=NP), for any $\epsilon > 0$ there exists no polynomial-time algorithm to approximate MAX-MIN-VC within a factor of $\frac{1}{2\sqrt{2}} n^{\frac{1}{2}-\epsilon}$ (respectively, $\frac{1}{2\sqrt[4]{2}} n^{\frac{1}{4}-\epsilon}$), where n is the number of vertices in an instance. For MAX-MIN-FVS we have, unless NP=ZPP (respectively, P=NP), for any $\epsilon > 0$, there exists no polynomial-time algorithm to approximate MAX-MIN-FVS within a factor of $\frac{1}{4} n^{\frac{1}{2}-\epsilon}$ (respectively, $\frac{1}{2\sqrt{2}} n^{\frac{1}{4}-\epsilon}$), where n is the number of vertices in an instance. For undirected graphs, we have similar results for MAX-MIN-FVS.

Next, we consider approximability of some of these problems restricted to bounded degree graphs and digraphs. By considering bounded degree graphs and digraphs, we show that MIN-FVS is APX-complete for 6-regular graphs and MAX-MIN-FVS is APX-hard for all graphs of maximum degree 9. Then, we prove that MAX-MIN-VC is k-approximable for all graphs without any isolated vertex and having maximum degree k, $k \geq 1$, and is APX-complete for all graphs of maximum degree 5. We also show APX-hardness of MIN-FAS and MAX-SUBDAG for k-total-regular digraphs, for all $k \geq 4$. Finally, we show that MIN-MAX-SUBDAG is APX-hard for digraphs of maximum total degree 12 and MAX-MIN-FVS is APX-hard for all digraphs of maximum total degree 6. These results are taken from the joint work [122] with K. Sikdar.

1.6 Organization of the Rest of the Thesis

In Chapter 2, we review the basic concepts and results, relevant to our work, regarding approximation algorithms and hardness of approximating NP-optimization problems. We begin the presentation of our work in Chapter 3, where we describe the linear ordering and related NP-optimization problems and their possible applications, review some

known results about their complexity and make some simple observations. In Chapter 4, we present our contributions regarding approximability of LOP, and in Chapter 5, we present some results about the complexity and approximability of OEOP. In Chapter 6, we present various results about hardness of approximating NP-optimization problems related to MIN-LOP and, finally, in Chapter 7, we make some concluding remarks and present some unresolved problems arising out of our work.

Chapter 2

On the Approximability of NP-hard Optimization Problems - the Basics

In this chapter, we review some of the basic concepts and results, that are relevant to our work, about approximation algorithms and hardness of approximating NP-hard optimization problems. More detailed survey can be obtained in several survey articles such as Ausiello et. al. [8] and Shmoys [143], and books, such as Papadimitriou and Steiglitz [131], Garey and Johnson [69], Papadimitriou [130], Bovet and Crescenzi [28], Ausiello et. al. [9], Hochbaum (Ed.) [90] and Mayr, Prömel and Steger (Ed.) [117].

We organize this chapter as follows. In Section 2.1, we review the basic concepts about NP-complete problems, and in Section 2.2, we present the concepts of NP-optimization problems, various kinds of approximation algorithms and associated approximation classes and discuss briefly about characterization of approximation classes, specifically, a characterization of the class PTAS due to Paz and Moran which will be used by us in Chapter 4. In Section 2.3, we introduce some approximation preserving reductions and the concept of completeness in approximation classes. Finally, in Section 2.4, we briefly discuss various types of results regarding hardness of approximation and methodology for establishing such results.

2.1 On NP-completeness

As mentioned in Chapter 1, the theory of NP-completeness plays a central role in explaining why for many problems it is hard to find efficient algorithms. The formal theory involves the concept of deciding languages over some alphabet by deterministic and nondeterministic Turing machines in polynomial time and the notion of polynomial-time reduction of one language to another. Deciding a language by a Turing machine corresponds to solving decision problems by intuitively conceived algorithms. Languages correspond to decision problems via appropriate encodings of instances by strings over an alphabet and a Turing machine (respectively, polynomial-time computability by a Turing machine) is a formal counter part of the intuitive notion of an algorithm (respectively, polynomial-time computability by an algorithm) via the classical Church-Turing Thesis (respectively, by the Extended Church-Turing Thesis) [144,50].

2.1.1 Decision Problems and Languages

A decision problem is specified by a set of instances and a question asked about them and it is required to answer, for an instance, 'yes' or 'no' in response to the question i.e., to check whether or not the instance satisfies a certain property. Let us give some standard examples of decision problems.

Example 2.1

1. Satisfiability of a CNF-formula (SAT)

   Instance : A boolean formula ϕ in conjunctive normal form

   Question : Is ϕ satisfiable?

2. Primality Testing (PRIME)

   Instance : A natural integer n

   Question : Is n a prime?

3. Hamiltonian Cycle (HC)

   Instance : A graph G

   Question : Does G have a Hamiltonian Cycle?

Given an alphabet $\sum$, containing at least 2 symbols, and a 'reasonable' encoding scheme [144], we can encode every instance of a decision problem π as a string over $\sum$ and to π we can associate the language L_π consisting of encodings of all 'yes' instances of π. Solving a decision problem π is to decide, for any instance x of π, whether x is an 'yes' instance or 'no' instance of π. Deciding a language L is to check, for any string x, whether $x \in L$ or $x \notin L$. Clearly, solving a decision problem π is equivalent to deciding the corresponding language L_π associated with π. Further, by the above connection, any decision problem can be specified by a language and every language specifies a decision problem.

From now on, we shall talk of decision problems and languages interchangeably. Further, by appealing to Church-Turing Thesis, we shall find it convenient to deal with the intuitive notion of an algorithm and its informal description. Also, for any instance x of a problem, we shall talk of its size, $|x|$, which is formally the number of symbols in an encoding of x but, informally, we shall take it to be a natural measure of the amount of storage space needed to store x, as is customary in informal discourse.

2.1.2 The Classes P and NP

An algorithm performs certain operations which are assumed to be *deterministic* in the sense that each operation yields a unique result. On an input x, an algorithm $\mathcal{A}$ performs a finite sequence of operations, called a *computation sequence*, and the length of the sequence is taken as a measure of *computation time* of $\mathcal{A}$ on input x. $\mathcal{A}$ is a polynomial-time algorithm if there exists a polynomial p such that the computation time of $\mathcal{A}$ is $O(p(n))$ for every input of length n.

An algorithm $\mathcal{A}$ solves a decision problem L if, for any input x, $\mathcal{A}$ *accepts* x (i.e. it outputs $\mathcal{A}(x)$=yes) if and only if $x \in L$.

The class of all decision problems which can be solved by polynomial-time algorithms is denoted by P.

The central concept of the theory of NP-completeness is the notion of *nondeterministic algorithm* which, unlike usual deterministic algorithms, can perform, apart from usual deterministic operations, certain operations which are *nondeterministic* in the sense that the result of such an operation is not uniquely determined but is chosen arbitrarily (or guessed) from a set of alternatives. The computation of such an algorithm. on any input, can be viewed as a tree of computation sequences, each corresponding to a sequence of possible results of the nondeterministic operations involved.

A nondeterministic algorithm $\mathcal{A}$ *accepts* an input x if there exists at least one computation sequence that accepts it; otherwise, it *rejects* x. It is usual to interpret a nondeterministic computation by saying that a nondeterministic algorithm has the magical ability to find and follow a correct accepting computation sequence for any input x, if there is any; otherwise, it rejects x. Thus, the computation time of such an algorithm $\mathcal{A}$ on input x is defined to be the computation time of the shortest accepting computation sequence, if x is accepted; otherwise, it is taken to be some constant, say, 0.

The notion of a polynomial-time nondeterministic algorithm and solution of a decision problem by it are similar to the deterministic case. The class of all decision problems which can be solved by polynomial-time nondeterministic algorithm is denoted by NP.

Clearly, P$\subseteq$NP. Whether P=NP is the central open question in theoretical computer science, but the general belief is that P$\neq$NP. A simple characterization of NP which gives an alternative description of polynomial-time nondeterministic computation is:

Proposition 2.1 *A language L is in the class NP if and only if there is a polynomial-time decidable binary predicate $\mathcal{Q}$ and a polynomial p such that, for all x, $x \in L$ if and only if $\exists y$ such that $(|y| \leq p(|x|)) \wedge \mathcal{Q}(x, y)$.*

For a proof see [130,51]. This gives an alternative description of a polynomial-time

nondeterministic computation as mentioned in Section 1.2.

2.1.3 Polynomial-Time Reductions and NP-completeness

A decision problem L_1 is *polynomial-time reducible* to a decision problem L_2, denoted by $L_1 \leq_m^p L_2$, if there is a polynomial-time computable function f such that, for any x, $x \in L_1$ if and only if $f(x) \in L_2$. $\leq_m^p$ is often called a *polynomial-time many-one reduction* or *Karp-reduction*.

Clearly,

(i) If $L_1 \leq_m^p L_2$ and $L_2 \in$ P, then $L_1 \in$ P, and, conversely, if $L_1 \notin$ P, then $L_2 \notin$ P.

(ii) If $L_1 \leq_m^p L_2$ and $L_2 \leq_m^p L_3$, then $L_1 \leq_m^p L_3$.

A decision problem L is *NP-hard* if, for every problem $M \in$ NP, we have $M \leq_m^p L$. An NP-hard problem L is *NP-complete* if L is also in NP.

It follows that if L is NP-complete, then L is solvable in polynomial time if and only if P=NP.

Cook [40] and Levin [110] established that SAT is NP-complete and then Karp [103] established several other problems to be NP-complete including HC. However, PRIME is not known to be NP-complete and is believed to be intermediate between P and NP-complete problems [133].

2.2 NP-hard Optimization Problems and Approximation Algorithms

In this section, we show how the implication of the theory of NP-completeness can be extended to optimization problems by considering the associated decision problems and then introduce the notion of an approximation algorithm, measure of approximation quality and classification of approximation algorithms and corresponding approximation classes.

2.2.1 Optimization and Decision Problems

Following Ausiello et. al. [8], we adopt the following

Definition 2.1 An optimization problem π is characterized by a four-tuple $\pi = (I_\pi, sol_\pi, m_\pi, goal_\pi)$ where

1. I_π is the set of instances of π,
2. sol_π associates, for any $x \in I_\pi$, the set $sol_\pi(x)$ of *feasible solutions* of x,
3. m_π, often called the *objective function* of π, associates, for any $x \in I_\pi$ and $y \in sol_\pi(x)$, a positive integer *measure* or *value* $m_\pi(x, y)$ of the solution y of x, and
4. $goal_\pi \in \{max, min\}$.

We are mainly concerned with *combinatorial* or *discrete* optimization problems for which the set of instances is countable and the set of feasible solutions for each instance is finite.

Solving an optimization problem π requires, for any $x \in I_\pi$, to find a solution $y^* \in sol_\pi(x)$, called an *optimal solution*, such that the value $m_\pi(x, y^*)$ is optimal over all $y \in sol_\pi(x)$; i.e. $m_\pi(x, y^*) \geq m_\pi(x, y)$ (respectively, $m_\pi(x, y^*) \leq m_\pi(x, y)$) for all $y \in sol_\pi(x)$ if π is a maximization (respectively, a minimization) problem. For any $x \in I_\pi$, $opt_\pi(x)$ denotes the set of optimal solutions of x and $m^*_\pi(x)$ denotes the value of an optimal solution y^* of x.

From now on, we make the *convention* of omitting the subscript π to the various components of an optimization problem π except when two or more problems are discussed at the same time. We now give some standard examples of combinatorial optimization problems formulated in the framework of Definition 2.1.

Example 2.2

1. Maximum Independent Set (MAX-IS)

   Instance : Graph $G = (V, E)$

   Solution : Subset $S \subseteq V$ such that no two vertices in S are joined by an edge in E

Cost : $|S|$

Goal : max.

2. Traveling Salesman Problem (MIN-TSP)

   Instance : Complete digraph $G_n = (V, A_n)$ with positive integer weights associated with the arcs

   Solution : A permutation π of V

   Cost : $\sum_{i=1}^{n-1} w(v_{\pi(i)}, v_{\pi(i+1)}) + w(v_{\pi(n)}, v_{\pi(1)})$, where $w(a)$ is the weight of the arc a

   Goal : min.

To each optimization problem π we can naturally associate a decision problem π_d defined as follows:

Definition 2.2 The decision problem π_d associated with the optimization problem π is to decide, for any $x \in I$ and $k \in \mathbb{N}$, whether $m^*(x) \geq k$, if $goal = max$, or whether $m^*(x) \leq k$, if $goal = min$.

The decision problem π_d can then also be specified by the language L_π defined as $L_\pi = \{(x,k) | x \in I, k \in \mathbb{N}, \text{ and } m^*(x) \geq k, \text{ if } goal = max \text{ (or } m^*(x) \leq k, \text{ if } goal = min)\}$.

Thus, the decision problems associated with the optimization problems given in example Example 2.2 are

1. MAX-IS$_d$ is specified as :

   Instance : Graph $G = (V, E)$ and $k \in \mathbb{N}$

   Question : Does there exists an independent set $s \subseteq V$ such that $|S| \geq k$?

2. MIN-TSP$_d$ is specified as :

   Instance : Complete digraph $G_n = (V, A_n)$ with an $n \times n$ arc weight matrix $W = (w_{ij})$ and a $k \in \mathbb{N}$

   Question : Does there exists a permutation π of V such that $\sum_{i=1}^{n-1} w(v_{\pi(i)}, v_{\pi(i+1)}) + w(v_{\pi(n)}, v_{\pi(1)}) \leq k$?

2.2.2 NP-hard Optimization Problems

Analogous to the class NP of decision problems we consider a class of optimization problems which lie in the border line between tractable and intractable optimization problems and whose feasible solutions are short and easy to recognize. Thus with appropriate constraints on a general concept of an optimization problem (see Definition 2.1) we get the concept of an NP-optimization problem. Apparently, Johnson [96] was the first to define such a concept, but the following definition closely follows that in Crescenzi and Panconesi [42] and Ausiello et.al. [9].

Definition 2.3 An optimization problem $\pi = (I, sol, m, goal)$ is an NP-optimization (NPO, for short) problem if

1. I is recognizable in polynomial time,
2. there is a polynomial p such that, for any x and for any $y \in sol(x)$, $|y| \leq p(|x|)$. Moreover, for any x and for any y such that $|y| \leq p(|x|)$, it is decidable in polynomial time whether $y \in sol(x)$ and
3. m is computable in polynomial time.

Definition 2.4 The class NPO is the set of all NPO problems and the class PO is the set of all NPO problems that are solvable in polynomial time. Max-NPO is the set of maximization NPO problems and Min-NPO is the set of minimization NPO problems. An NPO problem π is said to be *polynomially bounded* if a polynomial q exists such that, for any instance x and for any solution y of x, $m(x, y) \leq q(|x|)$. The class NPO-PB is the set of all polynomially bounded NPO problems. Max-PB is the set of all maximization problems in NPO-PB and Min-PB is the set of all minimization problems in NPO-PB.

While in the definition of NPO problem the nondeterminism is implicit, the following easy fact makes it explicit.

Proposition 2.2 [9] *If $\pi \in NPO$, then $L_\pi \in NP$.*

Definition 2.5 An optimization problem π is *NP-hard* if any decision problem $L \in$ NP can be solved in polynomial time by an algorithm which uses an oracle that returns (in unit time) an $y \in opt(x)$ and $m^*(x)$, for any $x \in I$.

The following facts are easy to establish [130,9].

Proposition 2.3

1. *If $\pi \in$ NPO and L_π is NP-complete, then π is NP-hard.*
2. *If P$\neq$NP, then PO$\neq$NPO.*
3. *If $\pi \in$ NPO can be solved in polynomial time, then π_d can be solved in polynomial time.*

From the Definition 2.5 and Proposition 2.3, it follows that if P$\neq$NP, then no NP-hard optimization problem can be solved in polynomial time. For example, MAX-IS and MIN-TSP are NP-hard optimization problems as their decision versions MAX-IS$_d$ and MIN-TSP$_d$ are well known NP-complete problems [103,128]. Often, an NP-hard optimization problem π whose decision version π_d is NP-complete is called an *NP-complete optimization problem.*

2.2.3 Approximation Algorithms and Measures of Quality of Approximation

As a practical approach to solve NP-hard optimization problems, researchers have considered designing efficient algorithms which may not yield optimal solutions but solutions which are 'close' to optimal solutions in some meaningful sense.

Definition 2.6 An *approximation algorithm* for an optimization problem π is an algorithm which, for any instance $x \in I$, yields a feasible solution $y \in sol(x)$ of x.

The quality or performance of an approximation algorithm $\mathcal{A}$ for an optimization problem π depends on how $m^*(x)$ and $m(x, \mathcal{A}(x))$ are related, where $\mathcal{A}(x)$ is the solution returned by $\mathcal{A}$ for the instance x.

There are various measures of quality of approximate solutions considered in the literature [96,42,11,98,9], but we will consider the following which are commonly used.

Definition 2.7 Let π be an NPO problem. Given an instance x of π and a feasible solution $y \in sol(x)$, we define :

1. The *absolute error* of y with respect to x is the difference
   $A(x,y) = |m(x,y) - m^*(x)|$
2. The *relative error* of y with respect to x is the ratio
   $E(x,y) = \frac{|m(x,y)-m^*(x)|}{\max\{m(x,y),m^*(x)\}}$
3. The *performance ratio* of y with respect to x is the ratio
   $R(x,y) = \max\{\frac{m(x,y)}{m^*(x)}, \frac{m^*(x)}{m(x,y)}\}$.

For most of the NP-complete optimization problems, there exists no polynomial-time algorithm for finding a solution with constant absolute error. Therefore, this measure is seldom used. But there are a few NPO problems having approximation algorithms with constant absolute error bound. For example, Minimum Degree Spanning Tree [64] and Minimum Edge Coloring [152] are two such problems.

The relative error and performance ratio are related by the relation $E(x,y) = 1 - \frac{1}{R(x,y)}$. From the definition of $E(x,y)$ and $R(x,y)$, it can be easily seen that $0 \leq E(x,y) \leq 1$ and $R(x,y) \geq 1$. The relative error is closer to 0 as the value of feasible solution is closer to the optimum value. Similarly, the performance ratio is closer to 1 as the value of feasible solution is closer to the optimum value. Thus, it is desired to have a feasible solution whose performance ratio is close enough to 1 (equivalently, relative error is close enough to 0).

2.2.4 Classification of Approximation Algorithms and Approximation Classes

In this section, we will introduce various types of approximation algorithms for NP-optimization problems and corresponding approximation classes in terms of perfor-

mance ratio. However, we will have some occasion to talk of absolute approximation algorithms as well. So, first we give the following definition.

Definition 2.8 An approximation algorithm $\mathcal{A}$ for an optimization problem π is a *c-absolute approximation algorithm* for π, where c is a given positive integer, if $A(x, \mathcal{A}(x)) \leq c$, for all instances x of π. $\mathcal{A}$ is an *absolute approximation algorithm* for π if it is a c-absolute approximation algorithm for π, for some c.

Definition 2.9 Given an approximation algorithm $\mathcal{A}$ for an optimization problem π and a constant $c \geq 1$, we say that $\mathcal{A}$ is a *c-approximation algorithm* for π if, for any instance x of π, the performance ratio of the approximate solution $\mathcal{A}(x)$ is bounded above by c, i.e. $R(x, \mathcal{A}(x)) \leq c$. More generally, for a function $f : \mathbb{N} \to [1, \infty)$, $\mathcal{A}$ is an $f(n)$-approximation algorithm for π, if, for all instances $x \in I$, $R(x, \mathcal{A}(x)) \leq f(|x|)$.

Definition 2.10 APX is the class of all NPO problems π such that, for some constant $c \geq 1$, there exists a polynomial-time c-approximation algorithm for π. More generally, given a class F of functions $f : \mathbb{N} \to [1, \infty)$, an NPO problem π belongs to the class F-APX, if there exists a polynomial-time $f(n)$-approximation algorithm for π, for some $f \in F$.

In particular, APX, log-APX, poly-APX and exp-APX will denote the classes F-APX when F is equal to the sets $O(1)$, $O(\log n)$, $O(n^{O(1)})$ and $O(2^{n^{O(1)}})$, respectively. From this definition, one could think that there is no difference between the classes NPO and exp-APX, since the polynomial bound on the computation time of the objective function implies that any NPO problem is $h2^{n^k}$-approximable for some h and k. This is not true, since NPO problems exist for which it is even hard to find a feasible solution; for example MAX-ONES-SAT is one such problem.

By the definition of a c-approximation algorithm, a polynomial-time 1-approximation algorithm for an NPO problem π finds an optimal solution for π in polynomial time. Hence, unless P=NP, there is no polynomial-time 1-approximation algorithm for .

NP-hard optimization problems. But many NP-hard NPO problems have polynomial-time c-approximation algorithms for some constant $c > 1$. For example, MIN-VC and MAX-W-SUBDAG have 2-approximation algorithms [17,132], and, hence, are in the class APX. But there are polynomial-time algorithms for some NP-hard optimization problems which approximate them within $1 + \epsilon$, for any $\epsilon \geq 0$. Such an algorithm is called a *polynomial-time approximation scheme* and it is formally defined as follows.

Definition 2.11 An approximation algorithm $\mathcal{A}$ for an optimization problem π is a *polynomial-time approximation scheme (ptas)* for π, if on input consisting of an instance x of π and a constant $\epsilon > 0$, it outputs a feasible solution $\mathcal{A}(x, \epsilon)$ in time polynomial in $|x|$ such that $R(x, T(x, \epsilon)) \leq 1 + \epsilon$.

The problem of finding a maximum independent set in a planar graph has a ptas [14]. The time complexity of this ptas is $O(n8^{\frac{1}{\epsilon}}/\epsilon)$, where n is the number of vertices in the graph. Unfortunately, the running time of this ptas increases rapidly with decreasing ϵ as well. A better behaviour of the time complexity would be a polynomial dependence in $\frac{1}{\epsilon}$. A ptas with such a time complexity is called a *fully polynomial-time approximation scheme.*

Definition 2.12 A ptas $\mathcal{A}$ for an optimization problem π is a *fully polynomial-time approximation scheme* (fptas) for π, if the running time of $\mathcal{A}$ is a polynomial in both $|x|$ and $\frac{1}{\epsilon}$, for any input consisting of an instance x of π and constant $\epsilon > 0$.

An example of an NP-optimization problem admitting an fptas is 01-KNAP [94].

Definition 2.13 PTAS is the class of all NPO problems that admit a polynomial-time approximation scheme and FPTAS is the class of all NPO problems that admit a fully polynomial-time approximation scheme.

There are few NPO problems which are known to be in the classes FPTAS and PTAS. For a list of these problems we refer to [98,9]. Clearly, the following inclusions

hold

$$\text{FPTAS} \subseteq \text{PTAS} \subseteq \text{APX} \subseteq \text{log-APX} \subseteq \text{poly-APX} \subseteq \text{exp-APX} \subseteq \text{NPO}$$

and these inclusions are strict if and only if P$\neq$NP [130,28,9].

2.2.5 On Characterization of the Class PTAS

As mentioned in Section 1.4, there are very few results that characterize any of the interesting approximation classes of NPO problems. Here, we shall describe only the characterization of the class PTAS due to Paz and Moran [134] that we shall use in Chapter 4. Before giving this characterization we first give few definitions.

Definition 2.14 For a given NPO problem π and a positive integer k, let $\pi_k = \{x \in I | m^*(x) \leq k\}$. The NPO problem π is said to be *simple* if, for every positive integer k, π_k is decidable in polynomial time.

One example of a simple NPO problem is MAX-IS [8] and in Section 4.5 we will show that MAX-SUBDAG is also simple.

Definition 2.15 An NPO problem π satisfies the *boundedness conditions* [134] if an algorithm $\mathcal{T}_b$ and a positive integer constant E exist such that the following hold:

1. For any instance x of π and for every positive integer c, $\mathcal{T}_b(x, c)$ is a solution y of x such that

$$\begin{aligned} m^*(x) &\leq m(x,y) + cE \quad \text{if } \pi \text{ is a maximization problem} \\ m^*(x) &\geq m(x,y) - cE \quad \text{if } \pi \text{ is a minimization problem.} \end{aligned}$$

2. The time complexity of $\mathcal{T}_b(x, c)$ is a polynomial in $|x|$ whose degree depends only on the value $\frac{m(x,\mathcal{T}_b(x,c))}{c}$.

Now we can state

Theorem 2.1 (Paz-Moran [134]) *An NPO problem π is in PTAS if and only if it is simple and satisfies the boundedness conditions.*

In the same paper [134], Paz and Moran also gave a similar characterization of FPTAS by modifying the two conditions on simplicity and boundedness, called *p-simplicity* and *polynomial boundedness.*

2.3 Reductions and Completeness in Approximation Classes

As already noted in Chapter 1, though the decision versions of most of the NP-hard optimization problems are polynomial-time many-one reducible to each other, the optimization problems do not share the same approximability properties as such reductions may not preserve the measure function or quality of the solutions. For optimization problems, a reduction should not only map instances of a problem π_1 to instances of a problem π_2 but should also map back good solutions of π_2 to good solutions of π_1, i.e. they should preserve approximation quality in some sense. There are various notions of approximation preserving reductions introduced and studied by researchers [128,42, 23,132,10,44,41]. But we shall introduce only those that will be used by us in our work. The reductions among the optimization problems that is regarded as the most appropriate is the AP-reduction formulated by Crescenzi et.al. [41] and is defined below.

Definition 2.16 Let π_1 and π_2 be two NPO problems. π_1 is said to be AP-reducible to π_2, in symbols $\pi_1 \leq_{AP} \pi_2$, if there exist two functions f and g and a constant $\alpha \geq 1$ such that :

1. For any instance $x \in I_{\pi_1}$ and for any rational $r > 1$, $f(x, r) \in I_{\pi_2}$.
2. For any instance $x \in I_{\pi_1}$ and for any rational $r > 1$, if $sol_{\pi_1}(x) \neq \phi$ then $sol_{\pi_2}(f(x, r)) \neq \phi$.
3. For any instance $x \in I_{\pi_1}$, for any rational $r > 1$ and for any $y \in sol_{\pi_2}(f(x, r))$, $g(x, y, r) \in sol_{\pi_1}(x)$.

4. f and g are computable by two algorithms $\mathcal{A}_f$ and $\mathcal{A}_g$, respectively, whose running time is polynomial for any fixed rational r.
5. For any instance $x \in I_{\pi_1}$, for any rational $r > 1$ and for any $y \in sol_{\pi_2}(f(x,r))$. $R_{\pi_2}(f(x,r),y) \leq r \Rightarrow R_{\pi_1}(x, g(x,y,r)) \leq 1 + \alpha(r-1)$.

It is easy to see that the AP-reducibility is transitive i.e., $\pi_1 \leq_{AP} \pi_2$ and $\pi_2 \leq_{AP} \pi_3$ implies that $\pi_1 \leq_{AP} \pi_3$. Also, an AP-reduction is indeed approximation preserving which is justified by

Theorem 2.2 *[9] For any two NPO problems π_1 and π_2, if $\pi_1 \leq_{AP} \pi_2$ and $\pi_2 \in APX$ (respectively, $\pi_2 \in PTAS$), then $\pi_1 \in APX$ (respectively, $\pi_1 \in PTAS$).*

A restricted notion of an AP-reduction is the notion of a *strict-reduction*, in symbol $\leq_{strict}$, which is an AP-reduction for which the condition 5, in the definition of AP-reduction, holds with $\alpha = 1$.

A weaker notion of reducibility is the L-reduction, in symbol $\leq_L$, introduced by Papadimitriou and Yannakakis [132]. This reduction specifies a linear relation between the optimal measures and a linear relation between the absolute errors.

Definition 2.17 Let π_1 and π_2 be two NPO problems. π_1 is said to be L-reducible to π_2, in symbols $\pi_1 \leq_L \pi_2$, if there exists two functions f and g and two positive constants α and β such that:

1. For any $x \in I_{\pi_1}$, $f(x) \in I_{\pi_2}$ is computable in polynomial time.
2. For any $x \in I_{\pi_1}$, if $sol_{\pi_1}(x) \neq \phi$ then $sol_{\pi_2}(f(x)) \neq \phi$.
3. For any $x \in I_{\pi_1}$ and for any $y \in sol_{\pi_2}(f(x))$, $g(x,y) \in sol_{\pi_1}(x)$ is computable in polynomial time.
4. For any $x \in I_{\pi_1}$, $m^*_{\pi_2}(f(x)) \leq \alpha \cdot m^*_{\pi_1}(x)$.
5. For any $x \in I_{\pi_1}$ and for any $y \in sol_{\pi_2}(f(x))$, $|m^*_{\pi_1}(x) - m_{\pi_1}(x, g(x,y))| \leq \beta \cdot |m^*_{\pi_2}(f(x)) - m_{\pi_2}(f(x),y)|$.

A useful fact that will allow us to use L-reduction instead of AP-reduction when we deal with problems in the class APX is

Lemma 2.1 *[41] Let π_1 and π_2 be two NPO problems such that $\pi_1 \leq_L \pi_2$. If $\pi_1 \in$ APX, then $\pi_1 \leq_{AP} \pi_2$.*

We also need the notion of equivalence of two problems with respect to a reduction.

Definition 2.18 Let $\leq_r$ be a reducibility notion among NPO problems. Two NPO problems π_1 and π_2 are equivalent with respect to the $\leq_r$-reducibility, in symbols $\pi_1 \equiv_r \pi_2$, if $\pi_1 \leq_r \pi_2$ and $\pi_2 \leq_r \pi_1$.

Analogous to the notion of NP-completeness of decision problems, we also have the notion of a complete problem in a class $\mathcal{C}$ of NPO problems with respect to an approximation preserving reduction to capture the idea of a problem that is hardest to approximate in the class $\mathcal{C}$. We shall formulate the notion with respect to AP-reducibility as that is the most appropriate reducibility notion for optimization problems.

Definition 2.19 Let $\mathcal{C}$ be a class of NPO problems and π be an NPO problem. π is said to be *$\mathcal{C}$-hard* (with respect to AP-reduction) if for any problem $\pi_1 \in \mathcal{C}$, $\pi_1 \leq_{AP} \pi$. A $\mathcal{C}$-hard problem π is *$\mathcal{C}$-complete* (with respect to AP-reduction) if $\pi \in \mathcal{C}$.

For various classes such as NPO, APX, PTAS, complete problems have been identified. For example, maximum weighted satisfiability, and minimum 0-1 integer programming problem are NPO-complete; maximum 3-satisfiability and max-cut are APX-complete [128,132]. The following result is well known.

Theorem 2.3 [41,42,9] *If an NPO problem π is APX-hard, then $\pi \notin$ PTAS, if P$\neq$NP.*

We also consider another reducibility, called S-reducibility, introduced by Kann [99] which is useful when size of an instance gets amplified by a reduction. This is defined bellow.

Definition 2.20 (Kann [99]) Let π_1 and π_2 be two NPO problems. π_1 is said to be *S-reducible* to π_2, in symbol $\pi_1 \leq_S \pi_2$, with size amplification $\alpha(n)$ if there is a four-tuple (f, g, α, c) such that

1. f and g are polynomial-time computable functions, α is a monotone increasing positive function and c is a positive constant.
2. $f : I_{\pi_1} \to I_{\pi_2}$ and $\forall y \in sol_{\pi_2}(f(x))$, $g(x, y) \in sol_{\pi_1}(x)$.
3. $\forall x \in I_{\pi_1}$ and $\forall y \in sol_{\pi_2}(f(x))$, $R_{\pi_1}(x, g(x, y)) \leq c \cdot R_{\pi_2}(f(x), y)$.
4. $\forall x \in I_{\pi_1}$, $|f(x)| \leq \alpha(|x|)$.

The following theorem follows from the definition of S-reduction.

Theorem 2.4 (Kann [99]) *Given two NPO problems π_1 and π_2, if $\pi_1 \leq_S \pi_2$ with size amplification $\alpha(n)$ and π_2 can be approximated within some monotone increasing positive function $u(n)$ of the size of the input instance, then π_1 can be approximated within $c \cdot u(\alpha(n))$, which is a monotone increasing positive function.*

2.4 Hardness of approximation

When we fail to design an efficient algorithm for an optimization problem π with a specified approximation quality, we would like to understand why we fail to do so. A result concerning hardness of approximation for a problem π provides some qualitative understanding and explanation of such a difficulty by relating the non-existence of the desired kind of approximation algorithm for π with some complexity theoretic hypothesis, such as P$\neq$NP, which is generally believed to be true. Such a result about π generally takes the form: if P$\neq$NP (or similar hypothesis), then there cannot exist a polynomial-time algorithm for π with a specified approximation quality (or $\pi \notin \mathcal{C}$, where $\mathcal{C}$ is a specified approximation class such as APX, PTAS, etc.).

Perhaps the first such hardness result, that holds for a wide class of NP-optimization problems, was obtained by Garey and Johnson [68]. To state their result we need some definitions.

Definition 2.21 (Garey and Johnson [68]) For a problem $\pi \in$ NPO and an instance $x \in I_\pi$, let $max(x)$ denote the value of the largest number occurring in x. Further, for a polynomial p, let $\pi_{max,p}$ denote the problem obtained by restricting π to those instances for which $max(x) \leq p(|x|)$. π is called *strongly NP-hard* if there is a polynomial p such that $\pi_{max,p}$ is NP-hard.

Then their result can be stated as :

Theorem 2.5 (Garey and Johnson [68]) *Let π be a strongly NP-hard optimization problem that admits a polynomial p such that $m^*(x) \leq p(|x|, max(x))$, for every instance x. Then, if P≠NP, $\pi \notin$ FPTAS.*

The theorem provides sufficient conditions, which can often be easily verified for a problem π to show that it cannot be in FPTAS, if P≠NP. For example, it is easy to show that MAX-IS and MAX-CUT satisfy the conditions of the theorem.

Often knowing appropriate characterization of class $\mathcal{C}$, it may be possible to show that a specific problem $\pi \notin \mathcal{C}$, if P≠NP. For example, using Paz and Moran characterization of the class PTAS, Theorem 2.1, we shall show, in Section 4.5, that a restricted version of linear ordering problem cannot be in PTAS, if P≠NP.

A general technique for proving hardness of approximation for a problem π within a specified factor r is to design a polynomial-time reduction from a known NP-hard decision problem π' that produces a gap by a factor g in the optimum values such that π is r-approximable, for $r < g$, implies that π' can be solved in polynomial time. This gap technique [141] was applied to MIN-TSP by Sahni and Gonzalez [141] using a reduction from HC problem to show that MIN-TSP $\notin$ APX, if P≠NP. Such a gap technique was of limited use for a long time till the PCP characterization of NP was established as mentioned in Section 1.4. Using PCP Theorem, many hardness of approximation results were obtained. The first such result was to show that the maximum-3-satisfiability problem is not in PTAS, unless P=NP [5]. Further, the PCP

Theorem also enabled to establish that APX=MAX-SNP and no MAX-SNP-complete problem can be in PTAS, if P≠NP [5].

As already mentioned in Section 1.4., there were also a series of papers concerning hardness of approximation of MAX-CLIQUE, MAX-IS, etc. culminating to the strongest such results due to Håstad [87]. We describe the result of Håstad which will be used by us in Chapter 6.

Before stating the results, we first recall the definitions of the probabilistic complexity class ZPP and its relationship with NP.

Definition 2.22 The class ZPP (probabilistic polynomial-time with zero error) consists of all languages L for which a polynomial-time bounded randomized Turing machine exists which for all inputs $x \in L$ accepts with probability strictly greater than $\frac{1}{2}$ and rejects with probability 0, and for all inputs $x \notin L$ rejects with probability strictly greater than $\frac{1}{2}$ and accepts with probability 0.

It can be shown that ZPP $\subseteq$ NP [28, 130, 117, 50], but it is not known whether ZPP=NP.

Theorem 2.6 (Håstad [87]) *Unless NP=ZPP, for any $\epsilon > 0$, there exists no polynomial-time algorithm to approximate MAX-IS within a factor of $n^{1-\epsilon}$, where n is the number of vertices in an instance.*

Theorem 2.7 (Håstad [87]) *Unless P=NP, for any $\epsilon > 0$, there exists no polynomial-time algorithm to approximate MAX-IS within a factor of $n^{\frac{1}{2}-\epsilon}$, where n is the number of vertices in an instance.*

To apply these results and establish similar results for certain problems in Chapter 6, we shall make use of S-reduction and the following result involving it.

Theorem 2.8 *Let $\pi_1 \leq_S \pi_2$ with size amplification $\alpha(n)$. Let $n = |x| \in I_{\pi_1}$ and $N = |f(x)| \leq an^k$, for $n \geq n_0$, where $a \geq 1$, $k > 0$ and n_0 is a positive integer. If there exists no polynomial-time algorithm to approximate π_1 within a factor of $\frac{1}{p}n^{q-\epsilon}$ for any*

$\epsilon > 0$, for some positive constants p and q, unless P=NP (unless NP=ZPP), then there exists no polynomial-time algorithm to approximate π_2 within a factor of $\frac{1}{pca^{q/k}} N^{\frac{q}{k}-\epsilon}$, for any $\epsilon > 0$, unless P=NP (unless NP=ZPP).

Proof. Proof is quite trivial and depends on the inequality $\frac{1}{pc} n^{q-\epsilon} \geq \frac{1}{pca^{q/k}} N^{\frac{q}{k}-\frac{\epsilon}{k}}$, which follows from $N \leq an^k$ and $a \geq 1$. □

Further, we shall use the following result in Chapter 4.

Theorem 2.9 (Feige [56]) *For any $\epsilon > 0$, there does not exist any polynomial-time approximation algorithm solving MIN-Set-Cover within a factor of $(1-\epsilon)\log n$, unless* NP⊂DTIME($n^{\log\log n}$).

Since, Min-Dom-Set is equivalent to MIN-Set-Cover under L-reduction [10], the same inapproximability result holds also for Min-Dom-set. This is a tight result as it matches with the positive result for these two problems due to Johnson [96].

Chapter 3

Linear Ordering and Related NP-optimization Problems

In this chapter, after introducing some relevant graph theoretic concepts, we introduce the specific NP-optimization problems considered in this thesis. These problems include both the maximization and minimization versions of linear ordering problem, various feedback set problems on graphs and digraphs and their relatives and generalizations, optimal evaluation ordering problem that arises in the context of implementation of eager functional languages, and a special case of quadratic assignment problem. We present applications of the problems, survey known results about their complexity and approximability and make some new observations, specially showing strong NP-completeness of linear ordering problems and establishing reductions and equivalences among the problems.

This chapter is organized as follows. In Section 3.1, we give the relevant graph theoretic concepts and some results about tournaments. In Section 3.2, we present the problems and briefly indicate, for each problem, possible applications in different practical and theoretical contexts. In Section 3.3, we make some observations on complexity of the problems, establish *strict* or AP-equivalences among the problems, and then mention some known results about their approximability. In Section 3.4, we give the formal definitions of the problems that we consider in this thesis.

3.1 Some Graph Theoretic Concepts

We shall denote a graph (i.e. an undirected graph) by $G = (V, E)$ and a digraph (i.e. a directed graph) by $G = (V, A)$, where, for some positive integer n, $V = \{v_1, v_2, \ldots v_n\}$, E is the edge set and A is the arc set. An edge between vertices v_i and v_j is an unordered pair which will be denoted by $\{v_i, v_j\}$, whereas an arc from v_i to v_j will be denoted by the ordered pair (v_i, v_j). If $\{v_i, v_j\}$ is an edge in G then we say that it is incident on v_i and v_j. If (v_i, v_j) is an arc in G then v_i is the initial point and v_j is the terminal point of (v_i, v_j). We shall denote a complete digraph on n vertices by $G_n = (V, A_n)$. In G_n, for any two vertices i and j in V both the arcs (i, j) and (j, i) are in A_n. In an undirected graph $G = (V, E)$, the set $N(i) = \{j | j \in V \text{ and } (i, j) \in E\}$ denotes the neighborhood of the vertex $i \in V$ and the set $N[i] = N(i) \cup \{i\}$ denotes the closed neighborhood of the vertex i in G. In an undirected graph G, *degree* of a vertex v_i is denoted as $d(v_i)$ which is the number of edges incident on v_i in G, and G is called *k-regular* if each vertex in G has degree k. In a digraph G, $d^+(v_i)$ and $d^-(v_i)$ are the number of arcs in G having v_i as the initial vertex and terminal vertex, respectively, and $d(v_i)$, the *total degree* of v_i is defined as $d(v_i) = d^+(v_i) + d^-(v_i)$. A digraph G is *k-total-regular* if for each vertex v_i, $d(v_i) = k$. A *path* $P(v_1, v_t)$ in $G = (V, E)$ (respectively, *dipath* in $G = (V, A)$) is a sequence of distinct vertices $(v_1, v_2, \ldots, v_t)$ such that $\{v_i, v_{i+1}\} \in E$ (respectively, $(v_i, v_{i+1}) \in A$) for $1 \leq i < t$. A path (respectively, dipath) $P(v_1, v_t)$ is called a *cycle* (respectively, *dicycle*) if $v_1 = v_t$.

A *tournament* on V [83] is a digraph (V, T) such that, for every two vertices $u, v \in V$, T contains exactly one arc with end vertices u and v. In this thesis, we shall also call the arc set T a tournament (assuming that the vertex set is given implicitly). A tournament on V is called an *acyclic tournament* on V if it does not contain any dicycle. A tournament (V, T) on V is called *transitive tournament* if (u, v) and $(v, w) \in T$ then $(u, w) \in T$, for any three vertices $u, v, w \in V$. The following theorem tells the equivalence between acyclic tournament and transitive tournament.

Theorem 3.1 (Moon [123]) *A tournament (V,T) on V is acyclic if and only if (V,T) is a transitive tournament.*

Further, a simple but useful fact about acyclic tournaments is that the set of all acyclic tournaments on a set V are in one-to-one correspondence with the set of all permutations of V. More precisely,

Proposition 3.1 *Any acyclic tournament (V,T) on V induces a unique permutation of V that is the topological ordering of (V,T) and, conversely, any permutation π of V induces a unique acyclic tournament (V,T_π) on V such that π is the topological ordering of (V,T_π).*

Proof Let $V = \{v_1, v_2, \ldots v_n\}$ and let (V,T) be any acyclic tournament on V. Since (V,T) is acyclic as well as a tournament, there exists a unique vertex of indegree 0 in (V,T). From this, it follows that there is a unique topological ordering of (V,T) which corresponds to a unique permutation, say $(v_{i_1}, v_{i_2}, \ldots v_{i_n})$ of V such that $(v_{i_j}, v_{i_k}) \in T$ if and only if $i_j < i_k$.

Conversely, let $\pi = (v_{i_1}, v_{i_2}, \ldots v_{i_n})$ be any permutation of V. Define $T_\pi = \{(v_{i_j}, v_{i_k}) : i_j < i_k\}$. Clearly, (V,T_π) is an acyclic tournament on V and π is a topological ordering of (V,T_π). Further, it is easy to see that (V,T_π) is the only acyclic tournament on V for which π is a topological ordering. □

A *feedback arc set* (FAS) in a digraph $G = (V,A)$ is an arc set $B \subseteq A$ such that the subdigraph $(V, A-B)$ is acyclic. Given a digraph $G = (V,A)$, a *minimal FAS* is a FAS $B \subseteq A$ which does not contain another FAS. (V,B) is called a *directed acyclic subgraph* (SUBDAG) of $G = (V,A)$ if $B \subseteq A$ and has no directed cycle. A SUBDAG (V,B) of G is called a maximal SUBDAG if there exists no SUBDAG (V,B') of G such that $B \subset B'$. Given a graph $G = (V,E)$, $C \subseteq V$ is called a *vertex cover* (VC) if for each edge $\{v_i, v_j\} \in E$, C contains either v_i or v_j. A VC C is called a *minimal VC* of G if no proper subset of C is also a VC of G. $S \subseteq V$ is called a *feedback vertex set* (FVS) of G if the subgraph/subdigraph $G[V-S]$, induced by the vertex set $V-S$, is acyclic.

Similarly a *minimal FVS* of G is defined. A dominating set in a graph $G = (V, E)$ is a vertex set S such that $N[i] \cap S \neq \phi$, for all $i \in V$. In the following we shall talk of vertex (arc/edge) weighted graphs (digraphs) in which the weights will be assumed to be nonnegative integers, unless stated otherwise.

3.2 The Problems and Their Applications

Here we will introduce the problems considered in this thesis and give a brief description of their applications.

3.2.1 Linear Ordering Problems

The maximum linear ordering problem (MAX-LOP) is a well studied problem under the name of linear ordering problem [77]. In MAX-LOP, given a complete digraph $G_n = (V, A_n)$ and a nonnegative arc weight function $w : A_n \rightarrow \mathbb{N}$, it is required to find an acyclic tournament (V, T) on V of maximum total arc weight. MAX-LOP has been known by several other names in the literature. For example, it is defined as a triangulation problem by Korte and Oberhofer [105] and as a permutation problem by Young [153]. These names reflect the application from which this optimization problem arises.

In the input-output analysis, the economy of a region is divided into n sectors, and an $n \times n$-input-output matrix X is constructed where the entry x_{ij} denotes the amount of deliveries from sector i to sector j in a certain year. The problem of permuting the rows and columns of X simultaneously such that the sum of entries of the permuted matrix above the main diagonal is as large as possible is called *triangulation problem* [105]. Considering an entry x_{ij} of the matrix X as weight of the arc (i, j) in G_n, one obtains a MAX-LOP problem.

Another application of this problem is *aggregation of individual preferences* and is some times also know as *ranking with paired comparison* [145]. In this application, a

group of persons wants to order n alternatives according to desirability, each person could make up his/her own preference scheme (i.e. he/she prefers some alternative i to alternative j; may be weighted preferences). By choosing c_{ij} to be the sum of weights assigned to "i is preferred to j", it is required to find a preference order that is the most consistent with individual preferences. This problem is also arises in marketing and psychology. Also MAX-LOP is used to break ties in sports tournaments or to rank the players.

Another important application of MAX-LOP is of minimizing total weighted (or average weighted) completion time in one-machine scheduling [24].

One may also be interested in solving the minimization version of MAX-LOP. We shall denote this minimization problem as MIN-LOP. In MIN-LOP, given an arc weighted (nonnegative integral) complete digraph $G_n = (V, A_n)$, it is asked to find an acyclic tournament of minimum total arc weight. MAX-LOP and MIN-LOP are complementary to each other in the sense that, for a given arc weighted complete digraph $(G_n = (V, A_n), w)$, (V, T) is an optimal solution of MAX-LOP for (G_n, w) if and only if $(V, A_n - T)$ is an optimal solution for MIN-LOP for (G_n, w). From this, it follows that MIN-LOP has similar applications as MAX-LOP has. For example, the triangulation problem is same as MIN-LOP, if we focus on minimizing the sum of entries below the main diagonal in permuted matrix, instead of maximizing the sum of entries above main diagonal. However, we are partly motivated to study MIN-LOP while investigating a combinatorial optimization problem called *Optimal Evaluation Ordering Problem* (OEOP) that arises in the context of minimizing the number of register spills in the eager functional language implementation and, as we shall see, this problem is closely related to MIN-LOP. We shall describe OEOP in details in the next section. MIN-LOP is also of theoretical interest as it is closely related to several other interesting known NP-optimization problems that we shall describe in the sequel. We shall write LOP (linear ordering problem) when referring to either MAX-LOP or MIN-LOP. Further, we write LOP(p, q) to denote LOP with weights restricted to two distinct positive integers

p and q.

3.2.2 Optimal Evaluation Ordering Problem

We are concerned with an optimization problem that arises in the context of functional language implementations. One important property of the functional language is the property of *referential transparency* [137]. It means that the value of an expression never changes throughout the process of computation. As a consequence, we can evaluate an expression in any order. Let us consider the expression $f(x, y) + g(x, y)$. If it is an expression in a functional program, the values of x and y remain the same always. So this expression could be evaluated in any order; i.e. we can do any of the following to get the final result. (i) first evaluate the call to f, then to g and add their results, or (ii) first evaluate the call to g, then to f and add their results. So far as the evaluation oder is concerned, functional languages are divided into two classes: *eager functional languages* and *lazy functional languages.* In an eager functional language, the arguments of a function are evaluated before the call to the function is made, whereas, in a lazy functional language, evaluation of an argument of a function is delayed and it is evaluated when it is needed. In the present discussion we will assume that evaluation strategy is eager.

The number of registers in the CPU of a computer usually ranges from 8 to 64. In some modern machines like RISC processors [92], their number may even exceed 200. If two operands are present in registers, operations like addition/subtraction usually take one machine cycle. On the other hand, if the operands are present in memory, they are first brought to registers, and then the addition/subtraction operation is performed. In RISC architectures [92], a memory operation usually takes 5 to 10 machine cycles. In CISC architectures, it may even be more. So it is always preferable that, as much operations as possible should be done in registers. But it is not always possible. Storing a register value and retrieving it later to a register constitute a *register spill* or simply a *spill.* It is desired that a program should be computed with minimum number of

spills.

Here we assume that (i) the language is a first order functional language, (ii) the evaluation strategy is eager and (iii) a function call may destroy the contents of the live registers, which is because we have to deal with recursion. We illustrate our problem through an example.

Example 3.1 Refer to Figure 3.1 which shows a DAG that represents the following expression.

$$F_1(1 + (x * (2 - (y * z)))) + F_2((x * (2 - (y * z))) + ((2 - (y * z)) + ((y * z) - 3)))$$

In the expression above, F_1 and F_2 are two function calls. Their arguments have many sharing computations. The DAG in Figure 3.1 shows them.

Because of the *referential transparency* property of functional languages, we can evaluate function calls F_1 and F_2 in any order. For evaluating the call to F_1, we need to evaluate the the argument of F_1 and hence the expression DAG enclosed within **LP2**. Similarly, before making the call to F_2, we have to evaluate the argument of F_2 which is the expression DAG enclosed within **LP3**. Observe that LP2 and LP3 have sharing between them. Now let us see the relative merits and demerits of the two possible orders of evaluation. Note that we are only considering the number of spills which result due to sharing between the argument DAGs of F_1 and F_2.

- Evaluation of F_1 is followed by F_2: We have to first evaluate LP_2. Then the call to F_1 will be made. After that LP_3 will be evaluated and it will be followed by the call to F_2. During the course of LP_2's evaluation, it will evaluate three shared computations, marked by circled 1, 2 and 3 in the figure. Immediately after the evaluation of LP_2, they are in registers. But following our assumptions, the call to F_2 may destroy them. So when LP_3 will be evaluated they may not in the registers in which they got evaluated. So we need to store them in memory and retrieve them when LP_3 will need them. In conclusion, we need to make 3 spills.

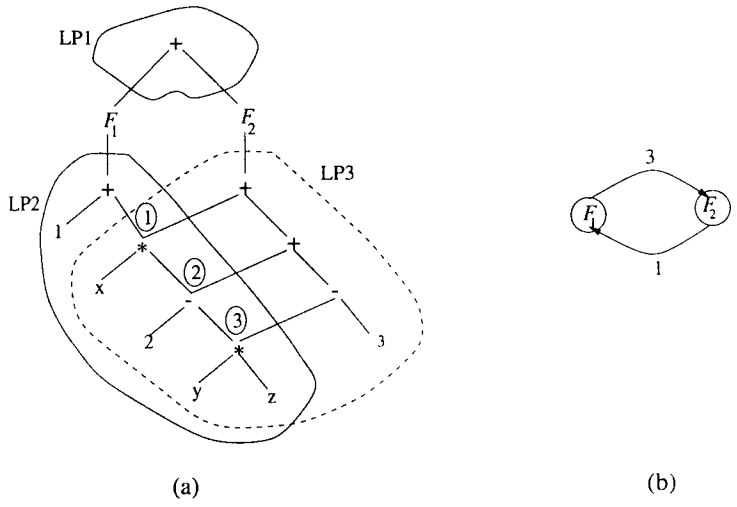


Figure 3.1: Sharing between the arguments of two calls F_1 and F_2

- Evaluation of F_2 is followed by F_1: We have to first evaluate LP_3. Then the call to F_2 will be made. After that LP_2 will be evaluated and it will be followed by the call to F_1. Observe that, during the course of LP_3's evaluation, it will evaluate only one shared computations, marked by circled 1 in the figure, which will be needed by LP_2. So in memory we need to store its value and retrieve it when LP_2 needs it. In conclusion, we will make 1 spill.

We observed that the second approach is more efficient than the first approach since we need to make lesser number of spills. □

Note that we are not just saving two or three spills. Such calls may occur in a recursive environment. In such a case the number of spills that we save by choosing a better evaluation order could be very very large. Let us illustrate when it happens. Let the definitions of functions F, A and B be as:

$$F = \ldots (A() + B()) \ldots$$
$$A = \ldots F(\ldots)$$
$$B = \ldots F(\ldots)$$

Here the expression $A() + B()$ in Figure 3.1 occurs in the body of function F and both

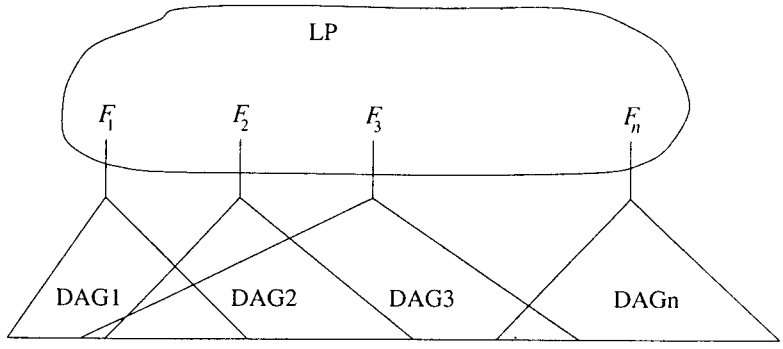


Figure 3.2: Sharing between arguments of n calls $F_1, F_2, \ldots, F_n$

A and B make calls to function F. So if we always evaluate B followed by A in the body of function F, then in each call to F we save 2 spills. And the number of calls to F could be very very large. So, in conclusion, choosing a better evaluation order is important.

We will now illustrate the general case by referring to Figure 3.2. In the figure, function calls $F_1, F_2, \ldots, F_n$ lie at the leaf-level of an expression DAG. To make matters simple, assume that the result of all such calls will be just added. Because of the referential transparency property of functional programs, such calls could be evaluated in any order. For simplicity, assume that all the functions are of one argument. In the figure DAG1,..., DAGn represent the argument DAGs of the functions $F_1, F_2, \ldots, F_n$ respectively, and they are expression DAGs (intermediate nodes of them are machine operators). There can be arbitrary sharing between such DAGs. A particular evaluation order of $F_1, F_2, \ldots, F_n$ will induce, due to the sharing between the argument DAGs, certain number of spills which are as discussed in the previous paragraphs. So now the problem is: how to choose an evaluation order of the calls to $F_1, ..., F_n$ such that the number of spills due to the sharing among argument DAGs is minimum. From now on we will refer to this problem as *optimal evaluation ordering problem (OEOP).*

In order to define the OEOP mathematically, we need to give few definitions. A DAG representing given arithmetic expressions $F_1, F_2, \ldots, F_n$, can be constructed in $O(m^2)$ time [1], where m is the number of arithmetic operators and operands in

$F_1, F_2, \ldots, F_n$. This DAG has n roots $r_1, r_2, \ldots, r_n$ (i.e. nodes with 0 indegree) corresponding to the expressions $F_1, F_2, \ldots, F_n$ where any node in this DAG can be reached from some root r_i. A subdag DAG_i corresponding to F_i is the subdag of the DAG induced by the nodes which are reachable from the root r_i. $SUBDAG(q)$, the subdag rooted at the node q is a common subdag of DAG_i and DAG_j if the node q is reachable from the roots r_i and r_j. A common $SUBDAG(q)$ of DAG_i and DAG_j is a maximal common subdag of DAG_i and DAG_j in DAG_i (respectively in DAG_j) if there exists a parent $p(q)$ of q for which $SUBDAG(p(q))$ is a subdag of DAG_i and not a subdag of DAG_j (respectively of DAG_i). For example in the Example 3.1 the DAG for $(y * z)$ is a maximal common subdag of arguments of F_1 and F_2 contained in the argument of F_2 whereas it is not a maximal common subdag contained in the argument of F_1.

Let SPILL(F_i, F_j) be the set of labels of roots of maximal common SUBDAGs of DAG_i and DAG_j, contained in DAG_j. If we evaluate F_i first and then F_j then we need to store the values of the SUBDAGs of DAG_i rooted at the nodes in the set SPILL(F_1, F_2) for the evaluation of F_2. To compute the set SPILL(F_1, F_2) we need to create an array of n bits for each node in the sharing DAG representing $F_1, F_2, \ldots, F_n$. For a node u in the sharing DAG, the ith bit of this array for u is 1 if and only if u is in the DAG_i otherwise it is 0. This array can be constructed in polynomial time by traversing each subdag DAG_i once. The set SPILL(F_i, F_j) can be computed by traversing the DAG_j as follows: make a DFS traversal starting from the root of the DAG_j, backtrack whenever a node u in the DAG_i is traversed by updating SPILL(F_i, F_j) by SPILL$(F_i, F_j) \cup \{u\}$. Thus SPILL(F_i, F_j) can be computed in polynomial time. Now the problem OEOP is defined as follows:

OEOP (Optimal Evaluation Ordering Problem)

Instance : n arithmetic expressions $F_1, F_2, \ldots, F_n$

Solution : A permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ of $\{1, 2, \ldots, n\}$

Cost : $w(\pi) = |\cup_{i=1}^{n-1} \cup_{j=i+1}^{n} \text{SPILL}(F_{\pi_i}, F_{\pi_j})|$

Goal - Min.

3.2.3 MAX-SUBDAG and Feedback Set Problems

Two problems closely related to MAX-LOP and MIN-LOP, and which have been investigated by several researchers, are maximum weight acyclic subdigraph (MAX-W-SUBDAG) and minimum weight feedback arc set (MIN-W-FAS) problems. Given an arc weighted digraph $(G = (V, A), w)$, in MAX-W-SUBDAG we are asked to find a subset $S \subseteq A$ of maximum total arc weight such that (V, S) is an acyclic subdigraph of G. In MIN-W-FAS, given an arc weighted digraph $(G = (V, A), w)$ we are asked to find $F \subseteq A$ of minimum total arc weight such that $(V, A - F)$ is an acyclic subdigraph of G. We also consider the vertex version of this problem called minimum weight feedback vertex set (MIN-W-FVS) problem. MIN-W-FVS is the problem in which we are given a vertex vertex weighted digraph $(G = (V, A), w)$ and are asked to find a FVS $F \subset V$ of minimum total weight. MIN-W-FVS is also defined on undirected graphs in similar way. The unweighted versions of these two problems, i.e. when all arc weights are 1, are denoted by MAX-SUBDAG, MIN-FAS and MIN-FVS.

Feedback set problems are originally formulated in the area of combinatorial circuit design, where a circuit node may receive new inputs before it stabilizes with old inputs, which is technically known as 'racing condition' [97]. To avoid such condition, a clocked register is placed at each vertex in the circuit. However, the delay in the circuit speed is proportional to the number of registers placed. Therefore, the objective is to minimize the number of clocked registers to be placed so that total delay can be minimized.

Finding a minimum FAS in a digraph has application in deadlock prevention in computer systems [113]. Consider an operating system which schedules different processes with requests on different resources, which need to use exclusively before being released by the process. This resource requirements can be modeled as a digraph in which a vertex corresponds to a process and it has an arc (i, j) if and only if process i requests a resource which is already allocated to process j. Therefore, if there is a dicycle in such a digraph then a deadlock occurs and every process in the dicycle will wait for the requested resource and never release the resource already allocated to it.

To break such dicycles, one can remove some processes form the digraph and put them in a waiting queue. It is clear that the objective is to minimize the number of removed processes.

Recently, Luccio [114] described another application of MIN-FVS problem in synchronous systems. Typically, a synchronous system is modeled by a network, whose vertices are colored either white or black. At each step, a vertex changes its color according to the majority of its neighbors implying that it is receiving the correct information. In monotone synchronous system only white colored vertices can change their color to black. Luccio established that in a toroidal mesh a feedback vertex set of black vertices causes all vertices to black after some steps and that a minimum FVS set is an initial configuration of minimum cardinality that stabilizes the system.

Feedback set problems have applications also in the areas of VLSI circuit testing [109], constraint satisfication problem [48, 17] and Baysian inference [17] and many others. For a list of other applications we refer to the survey article by Festa, Pardalos and Resende [59].

We also consider generalizations of the feedback set problems, called MIN-Subset-FAS and MIN-Subset-FVS. In MIN-Subset-FAS, we are given a digraph $(G = (V, A), w)$ along with a set $\mathcal{C}$ of interesting dicycles, and it is required to find a minimum weight FAS F that breaks all dicycles in $\mathcal{C}$. MIN-Subset-FVS is defined similarly. The motivation for these generalizations are of two fold. First, in some applications one may be interested for breaking some cycles that are interesting. Second, from the theoretical point of view, it is interesting to know whether the quality of the approximation depends on the type of interesting cycles/dicycles. In Section 4.2.2, we prove a stronger negative result for these generalizations than the known negative result via standard feedback set problems.

We consider two generalizations of MIN-LOP based on two properties of solutions of an instance of MIN-LOP. These properties are (i) an acyclic tournament (V, T) on V of G_n is a maximal SUBDAG of G_n and (ii) it is also a minimal FAS of G_n with exactly

$\frac{1}{2}n(n-1)$ arcs. Based on the first propery, one generalization is the minimum weight maximal SUBDAG (MIN-W-MAX-SUBDAG) problem that requires to find a maximal SUBDAG of minimum total arc weight in any given arc weighted digraph (which is not necessarily a complete digraph). The complementary problem corresponding to MIN-W-MAX-SUBDAG is the maximum weight minimal feedback arc set (MAX-W-MIN-FAS) problem which requires to find a minimal FAS of maximum weight in a given arc weighted digraph. In Chapter 6, we show that the unweighted version of these problems are APX-hard. We also consider other related problems such as MAX-W-MIN-FVS for both graphs and digraphs and MAX-MIN-VC. Based on the second property we have the MIN-W-k-FAS problem, which, for a given an arc weighted digraph $G = (V, A)$ and a positive integer k, it is required to find a minimum weight FAS F of G with $|F| = k$. In Chapter 4, we show that MIN-W-k-FAS and its vertex version MIN-W-k-FVS are NPO-complete.

3.2.4 Minimum Quadratic Assignment Problem

We also consider the minimum quadratic assignment problem (MIN-QAP), a special case of which will be shown to be related to MIN-LOP. This problem arises in many practical contexts. For example, suppose an industry wants to set up m plants in m of n ($n \geq m$) possible available locations such that at most one plant can be set up in one location. The industry has to consider various amounts of goods that have to be transported from one plant to another and the costs of transportation from one location to another. Then the problem is to choose m locations for the m plants so as to minimize the total cost of interplant transportation of goods.

To formulate the problem mathematically, let w_{ij} be the cost of transportation of one unit of goods from location i to location j, for all $i, j \in \{1, 2, \ldots n\}$, and let d_{ij} be the amount of goods to be transported from plant i to plant j, for all $i, j \in \{1, 2, \ldots m\}$. Further, let x_{ij} be a binary variable such that $x_{ij} = 1$ if and only if plant j is assigned to location i. Let $W = (w_{ij})$, $D = (d_{ij})$ and $X = (x_{ij})$. We assume that $w_{ii} = 0$,

$1 \leq i \leq n$, and $d_{ii} = 0$, $1 \leq i \leq m$. Then the MIN-QAP is specified as follows: Given an $n \times n$ matrix W, an $m \times m$ matrix D over non-negative integers, to determine an $n \times m$ binary matrix X so as to

$$\text{Minimize} \quad \sum_{\imath=1}^{n}\sum_{\jmath=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{m} w_{\imath\jmath} d_{kl} x_{\imath k} x_{\jmath l}$$

$$\text{Subject to} \quad \sum_{k=1}^{m} x_{\imath k} \leq 1, \quad \text{for } 1 \leq \imath \leq n \tag{3.1}$$

$$\sum_{\imath=1}^{n} x_{\imath k} = 1, \quad \text{for } 1 \leq k \leq m \tag{3.2}$$

$$x_{\imath k} \in \{0,1\}, \text{ for all } 1 \leq \imath \leq n \text{ and } 1 \leq k \leq m. \tag{3.3}$$

Note that an instance of MIN-QAP is specified by the pair (W, D) of non-negative integer matrices of dimensions $n \times n$ and $m \times m$, respectively, and so we shall often say that $(W_{n\times n}, D_{m\times m})$ is an instance of MIN-QAP.

MIN-QAP also arises in several other contexts such as scheduling [71], the backboard wiring problem in electronics [147], parallel and distributed computing [25]. MIN-QAP was shown to be NP-complete by Sahni and Gonzalez [141]. They also showed that MIN-QAP is not polynomial-time approximable within any constant, unless P=NP, by establishing a reduction from HC. Latter Queyranne [139] proved that it cannot be approximated within any constant even if the matrix D satisfies the triangle inequality. For a brief survey on MIN-QAP, regarding theoretical and algorithmic aspects as well as various applications, we refer to [32,129].

We shall consider a special case of MIN-QAP, called MIN-QAP(S), which consists of instances $(W_{n\times n}, D_{m\times m})$ of MIN-QAP satisfying the conditions (1) $m = n$ and (2) $d_{ij} = 1$, if $i < j$, and $d_{ij} = 0$, if $i \geq j$. In Section 3.3, we shall show that MIN-QAP(S) is related to MIN-LOP.

3.3 Preliminaries on the Complexity and Approximability of the Problems

In this section we describe some of the known results and some new ones about the complexity and approximability of the problems described in last section. After presenting some results about NP-completeness of these problems in Section 3.3.1, we establish equivalences among the problems with respect to appropriate approximation preserving reductions in Section 3.3.2. Then in Section 3.3.3, we give a brief survey of the known approximability results for these problems.

3.3.1 On the Complexity of the Problems

All the problems described in Section 3.2 are computationally hard. Like many other such problems mentioned in Chapter 1, these problems are such that, for each instance, the set of feasible solutions is finite but of cardinality that grows exponentially (or worse) with the size of the instances. Thus the naive algorithm that searches the set of feasible solution exhaustively is very inefficient though often much improved algorithm can be designed using appropriate clever algorithm design paradigms. For example, by Proposition 3.1, the set of feasible solutions for any instance of LOP has $n!$ many elements so that the naive algorithm has time complexity $O(n^2 n!)$, but like the Held and Karp's dynamic programming algorithm for MIN-TSP [91], we can design such an algorithm with time complexity $O(n^2 2^n)$ and space complexity $O(n2^n)$. We shall present such an algorithm for LOP in Chapter 4. But no polynomial-time algorithm for LOP is likely to exist, if P$\neq$NP as it is an NP-complete problem as shown bellow.

NP-completeness of LOP and LOP(p, q)

MAX-SUBDAG and MIN-FAS as well as their weighted versions are well known NP-complete optimization problems [103,69,77]. MAX-LOP (and hence MIN-LOP) is also a NP-complete optimization problem [79]. In fact, they are strongly NP-complete.

We present below the proof of these results about MAX-LOP and MIN-LOP as the reductions involved can be extended, as we shall see in the next section, for proving *strict*-equivalence of MIN-LOP with feedback set problems and AP-equivalence of MAX-LOP with MAX-SUBDAG.

First, we present the arguments of Grötschel, Jünger and Reinelt [77,78] to prove :

Proposition 3.2 *Both MAX-LOP and MIN-LOP are NP-complete.*

Proof It is easy to show that the decision version of LOP, viz., LOP$_d$ $\in$ NP. Next, by a reduction from the decision version MAX-W-SUBDAG$_d$ of MAX-W-SUBDAG we complete the argument to show that MAX-LOP$_d$ is NP-complete.

Let $G = (V, A)$ with a weight function w on the arc set A and a bound b be an instance of MAX-W-SUBDAG$_d$. Construct an instance of MAX-LOP$_d$ as follows: Let $G_n = (V, A_n)$ be obtained from G by adding all the arcs (u, v) such that $(u, v) \notin A$ and let $\overline{w}$ be the weight function on A_n defined as $\overline{w}(e) = w(e)$ if $e \in A$ and $\overline{w}(e) = 0$ if $e \notin A$. Also set the bound $\overline{b} = b$.

Now note that (V, T) is an acyclic tournament for G_n with $\overline{w}(T) \leq b$ if and only if $(V, T \cap A)$ is an acyclic subdigraph of G with $w(T \cap A) = \overline{w}(T) \leq b$. Thus, MAX-LOP$_d$ is NP-hard and as MAX-LOP$_d$ is easily seen to be in NP, it is NP-complete.

The NP-completeness proof of MIN-LOP$_d$ follows easily from that of MAX-LOP$_d$ because of their complementarity. □

For proving the strong-NP-completeness of these problems we shall use slightly different reductions. First we establish some simple facts that will be useful in these reductions.

Lemma 3.1 *Let $G = (V, A)$ be any digraph. Then the following holds:*

(i) For $B \subseteq A$, (V, B) is a maximal SUBDAG of G if and only if $F = A - B$ is a minimal FAS of G.

(ii) If (V, B) is any maximal SUBDAG of G, then B is of the form $B = T \cap A$ for some acyclic tournament (V, T) on V.

(iii) For any FAS $F \subseteq A$ of T, there exists a FAS $F' \subseteq F$ such that (V, F') is acyclic. Hence, any minimal FAS of G induces an SUBDAG of G.

Proof (i) If $B \subseteq A$ is such that (V, B) is a maximal SUBDAG of G, then $F = A - B$ is a FAS of G. If possible, let $F' \subset F$ be an FAS of G. Then $B' = A - F' \supset B$ and, by definition, (V, B') is acyclic, which contradicts maximality of B, and hence, F is a maximal FAS of G. The converse is also easy to show.

(ii) Let (V, B) be a maximal SUBDAG of G. Then a topological ordering of (V, B) induces a permutation $\pi = (v_{i_1}, v_{i_2}, \ldots v_{i_n})$ of V. Let $T = B \cup \{(v_{i_j}, v_{i_k}) : i_j < i_k \text{ and } (v_{i_j}, v_{i_k}) \notin B\}$. Then clearly, (V, T) is an acyclic tournament on V and $B = T \cap A$ as (V, B) is a maximal SUBDAG of G.

(iii) If (V, F) is acyclic, we can take $F' = F$. If (V, F) is not acyclic, then, it has at least one dicycle, say, $C = \{e_1, e_2, \ldots e_t\}$. Let $\overline{e}_k = (q_i, p_j)$, if $e_k = (p_j, q_i)$, for $1 \leq k \leq t$. Now, as F is a FAS, $(V, A - F)$ is acyclic and, as in the proof of Proposition 3.1, there is an acyclic tournament (V, T) on V such that $A - F \subseteq T$. Now T contains exactly one arc from $\{e_k, \overline{e}_k\}$, for $1 \leq k \leq t$. If T does not contain any arc from C, T must contain $\overline{C} = \{\overline{e}_1, \overline{e}_2, \ldots \overline{e}_t\}$ which is a dicycle, and so, T must contain at least one arc from C, say, e_i. So, $(A - F) \cup \{e_i\} \subseteq T$, and hence, $(V, (A - F) \cup \{e_i\})$ is acyclic. Thus, $F' = A - \{(A - F) \cup \{e_i\}\} = F - \{e_i\}$ is a FAS properly contained in F. If (V, F') is not acyclic, this argument can be repeated, finitely many times, to arrive at a FAS $F' \subset F$ such that (V, F') is acyclic. From the above argument, it follows that if F is a minimal FAS of G then (V, F) is an acyclic subdigraph of G. □

Next, we show that LOP(1, 2) is NP-complete from which it will follow that (i) LOP(p, q), $p \neq q$, is NP-complete as well as (ii) LOP with maximum weight bounded by a polynomial in the size of the input is also NP-complete and, hence, LOP is strongly NP-complete.

Theorem 3.2

(i) MAX-LOP(1, 2)$_d$ is NP-complete.

(ii) MIN-LOP(1, 2)$_d$ is NP-complete.

Proof (i) Given an instance of MAX-SUBDAG$_d$ consisting of a digraph $G = (V, A)$ and a positive integer bound b, construct an instance of MAX-LOP(1, 2)$_d$ as follows: Let $G_n = (V, A_n)$ with weight function w on A_n defined as: $w(e) = 2$ if $e \in A$ and $w(e) = 1$ if $e \notin A$, and the bound $\bar{b} = \frac{1}{2}n(n-1) + b$.

Now note that, by Lemma 3.1(ii), (V, B) is a maximal SUBDAG of G if and only if there exists a $T \subset A_n$ such that (V, T) is an acyclic tournament on V and $B = T \cap A$. Further, $w(T) = \frac{1}{2}n(n-1) + |B|$. From these it follows that (V, B) is a SUBDAG of G, which may not be maximal, with $|B| \geq b$ if and only if there is an acyclic tournament (V, T) on V with $w(T) \geq \frac{1}{2}n(n-1) + b$. Thus, MAX-LOP(1, 2)$_d$ is NP-hard. As it can be easily shown that MAX-LOP(1, 2)$_d \in NP$, it is NP-complete.

(ii) This follows from (i) by complementarity. However, we give another proof by a reduction from MIN-FAS$_d$ as it will be relevant in the sequel.

Let an instance of MIN-FAS$_d$ be consist of a digraph $G = (V, A)$ and a positive integer bound b. Construct an instance $G_n = (V, A_n, w)$ of MIN-LOP(1, 2)$_d$ with $w(e) = 1$ if $e \notin A$ and $w(e) = 2$ if $e \in A$ and the bound $\bar{b} = \frac{1}{2}n(n-1) + b$.

By Lemma 3.1(i) and (ii), it follows that F is a minimal FAS of G if and only if there exists an acyclic tournament (V, T) on V with $F = T \cap A$. Further, $w(T) = \frac{1}{2}n(n-1) + |F|$. From this it follows that F is a minimal FAS F of G with $|F| \leq b$ if and only if there exists an acyclic tournament (V, T) on V of G_n with $w(T) \leq \frac{1}{2}n(n-1) + b$. From this it follows that MIN-LOP(1, 2)$_d$ is NP-hard and is NP-complete as MIN-LOP(1, 2)$_d \in$ NP. □

From the above Theorem 3.2 it follows that both MAX-LOP(1, 2) and MIN-LOP(1, 2) are strongly NP-complete. Also these two problems do not admit of any fully polynomial-time approximation scheme, unless P=NP, as they are strongly NP-complete and their optimal values are bounded above by $n^2 - n$, which is a polynomial of input length. Thus in general MAX-LOP and MIN-LOP are not in FPTAS, unless P=NP.

NP-completeness of MIN-QAP(S)

Next we show that the special case MIN-QAP(S) of MIN-QAP is also NP-complete. For this it is enough to show:

Theorem 3.3 *MIN-LOP$_d$ $\leq_m^p$ MIN-QAP(S)$_d$.*

Proof Given an instance of MIN-LOP$_d$, consisting of $G_n = (V, A_n)$, an arc weight function w and a bound b, we construct an instance of MIN-QAP(S)$_d$ consisting of matrices W, D and a bound $\bar{b}$ as follows: $w_{ij} = w(i,j)$, for all $(i,j) \in A_n$, and $w_{ii} = 0$, for $1 \leq i \leq n$; $d_{ij} = 1$, for $i < j$, and $d_{ij} = 0$, for $i \geq j$; and $\bar{b} = b$.

It is easy to see that $tr(X'W'XD) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} w_{ij}d_{ij}x_{ik}x_{jl}$. Since $m = n$, any binary matrix $X = (x_{ij})$ satisfying (3.1), (3.2) and (3.3) is a permutation matrix. Also, given a permutation $\pi = (\pi_1, \pi_2, \ldots \pi_n)$ of $\{1, 2, \ldots, n\}$, it is not difficult to see that matrix $X^\pi = (x_{ij}^\pi)$, defined as $x_{ij}^\pi = 1$, if $i = \pi_j$, and $x_{ij}^\pi = 0$, otherwise, satisfies the conditions (3.1), (3.2) and (3.3). Thus, the set of feasible solutions of MIN-QAP(S) is in one-to-one correspondence with the set of permutations of $\{1, 2, \ldots, n\}$.

This shows, via Proposition 3.1, that there is a one-to-one correspondence between the acyclic tournaments of G_n and the feasible solutions of MIN-QAP(S)$_d$, for the above constructed instance. For a permutation π of $\{1, 2, \ldots, n\}$, let (V, T_π) be the acyclic tournament of G_n, as we showed in Proposition 3.1, and let X^π be the feasible solution of MIN-QAP(S) as constructed above.Next, we show that $w(T_\pi) = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} w_{\pi_i\pi_j} = tr(X^{\pi'}W'X^\pi D) = w(W, D, X)$. Since, $tr(X^{\pi'}W'X^\pi D) = \sum_{j=1}^{n} (X^{\pi'}W'X^\pi D)_{jj} = \sum_{j=1}^{n}\sum_{i=1}^{n}(X^{\pi'}W'X^\pi)_{ji}d_{ij}$, we need to show that $(X^{\pi'}W'X^\pi)_{ji} = w_{\pi_i\pi_j}$. Now,

$$\begin{aligned}
(X^{\pi'}W'X^\pi)_{ji} &= \sum_{k=1}^{n}(X^{\pi'}W')_{jk}x_{ki}^\pi \\
&= (X^{\pi'}W')_{j\pi_i} \qquad (\text{as } x_{\pi_i i}^\pi = 1) \\
&= \sum_{k=1}^{n} x_{jk}^{\pi'}w'_{k\pi_i} = \sum_{k=1}^{n} x_{kj}^\pi w_{\pi_i k} \\
&= w_{\pi_i\pi_j} \qquad (\text{as } x_{\pi_j j}^\pi = 1).
\end{aligned}$$

Hence $tr(X^{\pi'}W'X^{\pi}D) = w(T_\pi)$, as $d_{ij} = 1$, for $i < j$. From this it follows that (V, T) is an acyclic tournament of (G_n, w) with $w(T) \leq b$ if and only if MIN-QAP(S)$_d$ has a solution X, for the above instance, with $tr(X'W'XD) \leq \bar{b}$. From this the theorem follows. □

In the next subsection, we examine relationships among some of the problems with respect to approximability and we postpone discussion of complexity and approximability of OEOP till Chapter 5.

3.3.2 Equivalences Among the Problems

In this subsection, we establish some *strict*-reductions or *AP*-reductions and equivalences among several problems discussed in Section 3.2 and take note of their consequences.

Strict Equivalences among MIN-LOP and Other Problems

From the discussions in Section 3.3.1, it is apparent how reductions between MIN-W-FAS and MIN-LOP can be defined. We show that such simple reductions are indeed *strict*-reductions.

Theorem 3.4 *MIN-W-FAS* $\leq_{strict}$ *MIN-LOP.*

Proof. Let $x = (G, w)$ be an instance of MIN-W-FAS where $G = (V, A)$. Let, for any rational $r > 1$, $f(x, r) = (G_n, \overline{w})$ be the instance of MIN-LOP with $\overline{w}(e) = w(e)$ if $e \in A$ and $\overline{w}(e) = 0$ otherwise. Let, for $y \in sol(f(x, r))$, $g(x, y, r) = y \cap A$. Since $\overline{w}(y) = w(y \cap A)$, it follows that $m(f(x), y) = m(x, g(x, y, r))$. Next we show that

$m^*(f(x,r)) = m^*(x)$.

$$
\begin{aligned}
m^*(f(x,r)) &= \min_{y \in sol(f(x,r))} m(f(x,r),y) \\
&= \min_{y \in sol(f(x,r))} m(x, g(x,y,r)) \\
&= \min_{\substack{s \in sol(x), \\ s \text{ acyclic FAS}}} m(x,s) && \text{(Since } g(x, sol(f(x,r)), r) = \text{set of all acyclic FASs of } x\text{)} \\
&= \min_{\substack{s \in sol(x), \\ s \text{ a minimal acyclic FAS}}} m(x,s) && \text{(Since } w(e) \geq 0\text{)} \\
&= \min_{\substack{s \in sol(x), \\ s \text{ a minimal FAS}}} m(x,s) && \text{(Since, by Lemma 3.1(iii), set of minimal acyclic FASs of } G = \text{the set of minimal FASs of } G.\text{)} \\
&= m^*(x)
\end{aligned}
$$

Hence, for any instance x of MIN-W-FAS, for any rational $r > 1$ and for any $y \in sol(f(x,r))$,

$$\frac{m(f(x,r),y)}{m^*(f(x,r))} = \frac{m(x,g(x,y,r)}{m^*(x)}.$$

Hence, MIN-W-FAS $\leq_{strict}$ MIN-LOP. □

Next we show that MIN-LOP$\leq_{strict}$MIN-W-FAS. Before that we prove a lemma.

Lemma 3.2 *A feedback arc set T of G_n is minimal if and only if (V,T) is an acyclic tournament on V.*

Proof. Clearly an acyclic tournament on V is a minimal FAS for G_n

Conversely, let F be any minimal FAS of G_n. Then, by Lemma 3.1, F is an acyclic FAS of G_n. We will show that F is an acyclic tournament on V. For, if not, then there exists $u, v \in V$ such that neither (u,v) nor $(v,u) \in F$. Hence, both $(u,v), (v,u) \in A_n - F$ which shows that $A_n - F$ contains a cycle. But as F is a FAS, $A_n - F$ must be acyclic. □

Note that a FAS of G_n may contain more than one minimal FAS of G_n. Here we define a procedure which, for a given FAS F of G_n, constructs a specific minimal FAS

of G_n contained in F. We call this procedure TOPO-ORD(F, G_n). For an input FAS F of G_n, the procedure finds a unique least indexed node say p_1 in the acyclic digraph $(V, A_n - F)$ with 0 indegree, then it deletes the node p_1 and all the arcs incident to it in $(V, A_n - F)$. It repeats this process on the remaining digraph to have a unique order say $(p_1, p_2, \ldots, p_n)$ of V. Then this algorithm returns the acyclic tournament (V, T) on V, where $T = \{(p_i, p_j) | i > j\}$. $T \subseteq F$ since $(A_n - F) \subset (A_n - T)$.

Theorem 3.5 *MIN-LOP* $\leq_{strict}$ *MIN-W-FAS.*

Proof. Let $x = (G_n, w)$ be an instance of MIN-LOP. We regard x also as an instance of MIN-W-FAS so that $f(x, r) = x$, for any rational $r > 1$. For any given $y \in sol(f(x, r))$, let $g(x, y, r) =$ TOPO-ORD(y, G_n). From the definition of g it follows that (i) $m(f(x, r), y) \geq m(x, g(x, y, r))$. Next we show that (ii) $m^*(f(x, r)) = m^*(x)$.

$$\begin{aligned} m^*(f(x,r)) &= \min_{y \in sol(f(x,r))} m(f(x,r), y) \\ &= \min_{\substack{y \in sol(f(x,r)), \\ y \text{ a minimal FAS}}} m(f(x,r), y) \quad (\text{Since } w(e) \geq 0.) \\ &= \min_{\substack{y \text{ an acyclic} \\ \text{tournament on V}}} m(x, y) \qquad (\text{By Lemma 3.2}) \\ &= m^*(x). \end{aligned}$$

Now , let $\frac{m(f(x,r),y)}{m^*(f(x,r))} \leq r$. Then $m(f(x, r), y) \leq r \cdot m^*(f(x, r))$. By (i) and (ii) we have $m(x, g(x, y, r)) \leq r \cdot m^*(x)$. Hence, $\frac{m(x,g(x,y,r))}{m^*(x)} \leq r$. □

Thus we have

Theorem 3.6 *MIN-LOP* $\equiv_{strict}$ *MIN-W-FAS.* □

Theorem 3.7 *MIN-LOP has an* $O(\log n \log\log n)$*-approximation algorithm.*

Proof. The proof of Theorem 3.5 shows that if MIN-W-FAS has an $f(n)$-approximation algorithm then MIN-LOP has such an algorithm. So it is enough to show that MIN-W-FAS has an $O(\log n \log\log n)$-approximation algorithm. Now in [53] an $O(\log n \log\log n)$-approximation algorithm is obtained for instances of MIN-W-FAS with positive integer

arc weights. However, this algorithm can be easily extended to instances of MIN-W-FAS with nonnegative integer arc weights having same approximation property as follows.

Let $(G = (V, A), w)$ be an instance of MIN-W-FAS with $w(e) \geq 0$. Consider $(G' = (V, A'), w)$ with $A' = \{e \in A | w(e) \geq 1\}$. Let F' be a FAS obtained by Even et.al.'s algorithm [53] for G' and F'_o be an optimal solution for G'. Thus $F = F' \cup (A - A')$ is a FAS for G. Also $F_o = F'_o \cup (A - A')$ is an optimal solution for G. For, if not, there exists a FAS $\overline{F}$ of G with $w(\overline{F}) < w(F'_o \cup (A - A'))$. Now $\overline{B} \cap A'$ is a FAS for G', and $w(\overline{F} \cap A') = w(\overline{F})$. Thus $w(\overline{F} \cap A') < w(F_o)$, implying that F'_o is not optimal FAS for G', which is a contradiction. Now it can be easily shown that $F' \cup (A - A')$ is $O(\log n \log\log n)$-approximate solution of MIN-W-FAS for the instance $(G = (V, A), w)$. □

In Section 3.3.1, we have proved that MIN-QAP(S) is NP-complete and next, we show that MIN-LOP $\equiv_{strict}$ MIN-QAP(S).

Lemma 3.3 *MIN-LOP* $\leq_{strict}$ *MIN-QAP(S).*

Proof Given an instance (G_n, w) of MIN-LOP construct an instance (W, D) of MIN-QAP(S) as constructed in the proof of Theorem 3.3. There we have noted that there is a one-to-one correspondence between acyclic tournaments of G_n and solutions of MIN-QAP(S) for the associated instance (W, D) which are in one-to-one correspondence with permutation of $\{1, 2, \ldots, n\}$. Also we proved that, for a permutation π of $\{1, 2, \ldots, n\}$, corresponding to the acyclic tournament (V, T_π), the feasible solution of MIN-QAP(S) for the instance (W, D) is X^π and $w(T_\pi) = tr(X^{\pi'} W' X^\pi D)$. From this the required *strict*-reduction follows. □

Lemma 3.4 *MIN-QAP(S)* $\leq_{strict}$ *MIN-LOP.*

Proof Given an instance (W, D) of MIN-QAP(S), construct (G_n, w) as an instance of MIN-LOP with $w(i, j) = w_{ij}$, for all $i, j \in \{1, 2, \ldots n\}$ and $i \neq j$. The rest of the arguments for the proof is as in the proof Lemma 3.3. □

From Lemma 3.3 and Lemma 3.4, it follows that:

Theorem 3.8

(i) MIN-QAP(S) $\equiv_{strict}$ *MIN-LOP.*

(ii) MIN-QAP(S) has an $O(\log n \log\log n)$*-approximation algorithm.*

In general, approximability properties of weighted NPO problems do not behave in the same way with their respective unweighted problems. In fact, many unweighted versions of optimization problems have comparatively simpler approximation algorithms as compared to their weighted versions, though they have the same approximation error. For example, Gavril's 2-approximation algorithm for unweighted minimum vertex cover (MIN-VC) problem is based on a simple greedy procedure to find a matching in the graph [68], whereas the 2-approximation algorithm for weighted version of MIN-VC (MIN-W-VC) is based on a linear programming formulation [125]. In some other cases, it is not known whether they attain the same approximation ratio; for example, maximum clique problem is approximable within a factor of $O(n/\log^2 n)$ [26], whereas the best known approximation factor for maximum weighted clique is $O((\log\log n)^2 n/\log^2 n)$ [82]. But it is known that several NP-hard optimization problems are tractable whenever a polynomial bound is imposed on the weights. Recently, Crescenzi et. al. [43] have shown that for any weighted optimization problem satisfying a "niceness" property, the approximation threshold of the unbounded version and that of a polynomially bounded versions are equal. They show that the MIN-VC, minimum satisfiability and some other problems are exactly as hard to approximate as their weighted versions. The question of knowing whether MIN-FVS has such property is raised by Trevisan [150]. Crescenzi et.al. [43] proved that the unweighted and polynomially bounded weighted versions of some NP optimization problems have the same approximability threshold. In fact, they proved that the unweighted versions of minimum vertex cover, minimum SAT, maximum cut, maximum dicut and maximum 2SAT are exactly as hard to approximate as their weighted versions. Here we show similar results about MIN-FVS, MIN-FAS, MIN-Subset-FVS and MIN-Subset-FAS by

establishing several *strict*-reductions. For this, first we define a notation that we shall use. For an NP-optimization problem π on graphs/digraphs and a polynomial p, we denote by π^p the restriction of π to instances of x such that the sum of weights on the arcs/vertices is at most $p(|x|)$.

Theorem 3.9

1 MIN-W-FVS^p $\equiv_{strict}$ *MIN-W-FAS^p* $\equiv_{strict}$ *MIN-FVS* $\equiv_{strict}$ *MIN-FAS.*

2 MIN-W-Subset-FVS^p $\equiv_{strict}$ *MIN-W-Subset-FAS^p* $\equiv_{strict}$ *MIN-Subset-FVS* $\equiv_{strict}$ *MIN-Subset-FAS.*

Proof. To prove this theorem it is enough to show the following
(1) MIN-W-FVS^p $\leq_{strict}$ MIN-W-FAS^p $\leq_{strict}$ MIN-FAS $\leq_{strict}$ MIN-FVS $\leq_{strict}$ MIN-FAS.
(2) MIN-W-Subset-FVS^p $\leq_{strict}$ MIN-W-Subset-FAS^p $\leq_{strict}$ MIN-Subset-FAS $\leq_{strict}$ MIN-Subset-FVS $\leq_{strict}$ MIN-Subset-FAS.
Ali the reductions can be found in [53] except the reductions MIN-W-FAS^p $\leq_{strict}$ MIN-FAS and MIN-W-Subset-FAS^p $\leq_{strict}$ MIN-Subset-FAS. We will establish the first one as the same reduction works for the second one.

Given an instance $(G = (V, A), w)$ of MIN-W-FAS, construct an instance $G' = (V', A')$ of MIN-FAS with $V' = V \cup [\cup_{(v_i,v_j)\in A}\{v_{ij}^1, v_{ij}^2, \ldots v_{ij}^{w_{ij}}\}]$ and $A' = \cup_{(v_i,v_j)\in A}\{(v_i, v_{ij}^1), (v_{ij}^1, v_j), (v_i, v_{ij}^2), (v_{ij}^2, v_j), \ldots (v_i, v_{ij}^{w_{ij}}), (v_{ij}^{w_{ij}}, v_j)\}$. In other words, G' is constructed from (G, w) by replacing each arc $(v_i, v_j) \in A$ with $2w_{ij}$ arcs $(v_i, v_{ij}^1), (v_{ij}^1, v_j), (v_i, v_{ij}^2), (v_{ij}^2, v_j), \ldots (v_i, v_{ij}^{w_{ij}}), (v_{ij}^{w_{ij}}, v_j)$, i.e. each arc $(v_i, v_j) \in A$ is replaced by w_{ij} dipaths (of length 2) $P_{ij}^t = (v_i, v_{ij}^t, v_j)$, for $1 \leq t \leq w_{ij}$ (see Fig. 3.3). We shall denote this set of arcs corresponding to the arc (v_i, v_j) by $A'(v_i, v_j)$. We note that, given $(G = (V, A), w)$, $G' = (V', A')$ can be constructed in polynomial time. From this construction it is easy to observe that $w(A) = \frac{1}{2}|A'|$.

For a minimal FAS F' of G', if $F' \cap A'(v_i, v_j) \neq \phi$ then, there must be at least one dipath $P(v_j, v_i)$ from v_j to v_i in G'. Since the dipaths P_{ij}^t, $1 \leq t \leq w_{ij}$, and $P(v_j, v_i)$

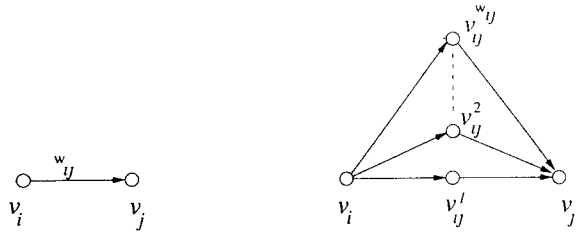


Figure 3.3: An arc $(v_i, v_j) \in A$ and its corresponding subdigraph in G'

form w_{ij} distinct dicycles in G', and the minimal FAS F' of G' contains at least one arc from $A'(v_i, v_j)$, F' must contain exactly one arc from each dipath P_{ij}^t, for $1 \leq t \leq w_{ij}$. Hence, without loss of generality, we assume that if $F' \cap A'(v_i, v_j) \neq \phi$, for a minimal FAS F' of G', then F' contains the arc set $\{(v_i, v_{ij}^t) | 1 \leq t \leq w_{ij}\}$. Now it is clear from the construction that, if F is a minimal FAS of G then, $F' = \cup_{(v_i,v_j)\in F}\{(v_i, v_{ij}^t) | 1 \leq t \leq w_{ij}\}$ is a minimal FAS of G' and conversely. Also $w(F) = |F'|$. From this the required *strict*-reduction follows. □

As a consequence of Theorem 3.6 and Theorem 3.9, we have

Theorem 3.10 *MIN-LOP*^{p} $\equiv_{strict}$ *MIN-FAS* $\equiv_{strict}$ *MIN-FVS.* □

Equivalences among MAX-LOP, MAX-W-SUBDAG and Their Variants

Theorem 3.11

1. *MAX-LOP* $\equiv_{strict}$ *MAX-W-SUBDAG.*
2. *MAX-LOP*^{p} $\equiv_{AP}$ *MAX-W-SUBDAG*^{p}.

Proof Since the proof of this Theorem is similar to that of Theorems 3.4 and 3.6, we shall give a sketch of it.

It is easy to see that MAX-LOP $\equiv_{strict}$ MAX-W-SUBDAG. For MAX-LOP $\leq_{strict}$ MAX-W-SUBDAG, an instance (G_n, w) is also the instance for MAX-W-SUBDAG and to a solution y of the latter associate a specific tournament on V containing y. For MAX-W-SUBDAG $\leq_{strict}$ MAX-LOP, given an instance $(G = (V, A), w)$ of the former

associate the instance $(G_n, \overline{w})$ for the latter with $\overline{w}(e) = w(e)$ for $e \in A$ and $\overline{w}(e) = 0$ otherwise and to a solution y of the latter associate the solution $y \cap A$ for the former.

The proof of the remaining part the Theorem follows similarly. □

Next, we show that MAX-SUBDAG $\equiv_{AP}$ MAX-W-SUBDAG^p. For this equivalence, it is required to show that MAX-SUBDAG $\leq_{AP}$ MAX-W-SUBDAG^p and MAX-W-SUBDAG^p $\leq_{AP}$ MAX-SUBDAG. As the first one is trivial we shall prove the second one.

Lemma 3.5 *MAX-W-SUBDAG^p $\leq_{AP}$MAX-SUBDAG.*

Proof Given an instance $(G = (V, A), w)$ of MAX-W-SUBDAG^p, we construct an instance $G' = (V', A')$ of MAX-SUBDAG as constructed in the proof of the Theorem 3.9.

Let (V, S) be a maximal SUBDAG of G. Then, (V', S') with the arc set $S' = \left[\cup_{(v_i,v_j)\in S}\{P_{ij}^t | 1 \leq t \leq w_{ij}\}\right] \cup \left[\cup_{(v_i,v_j)\in A-S}\{(v_i, v_{ij}^t) | 1 \leq t \leq w_{ij}\}\right]$ is a maximal SUBDAG of G'. This is because, there exists a dipath from v_i to v_j in (V', S') if and only if there exists a dipath from v_i to v_j in (V, S). Also $|S'| = 2w(S) + w(A - S) = w(A) + w(S)$.

Let (V', S') be a maximal SUBDAG of G'. Then, from the construction of G' from G, it is clear that if v_i and v_j are connected by a dipath of length 2 in (V', S') then (V', S') has all the w_{ij} dipaths of length 2 from v_i to v_j. Also, since (V', S') is a maximal SUBDAG of G', if $(v_i, v_j) \in S$ and v_i and v_j are not connected by any dipath of length 2 in (V', S') then (V', S') must contain exactly one arc from each w_{ij} dipaths of length 2 connecting v_i to v_j in G'. From this observation, it follows that given any maximal SUBDAG (V', S') of G' the associated SUBDAG (V, S) of G, where $S = \{(v_i, v_j) | (V', S')$ has a dipath of length 2 from v_i to $v_j\}$, is a maximal SUBDAG of G and $|S'| = w(S) + w(A)$.

Hence, if (V, S_o) is a maximum weight SUBDAG of (G, w), then the associated SUBDAG (V', S_o') (as defined above) is a maximum cardinality SUBDAG of G'. Then

$|S'_o| = w(S_o) + w(A)$ implying that $2w(S_o) \leq |S'_o|$.

In order to show the required AP-reduction, for any given maximal SUBDAG (V', S') of G', we associate a SUBDAG (V, T) of G as follows: From (V', S') construct the arc set S as defined above. Define $T = S$ if $w(S) \geq \frac{1}{2}w(A)$, and $T = A - S$ if $w(S) < \frac{1}{2}w(A)$. Now, (V, T) is a SUBDAG of G as (V, S) is a maximal SUBDAG of Gi. *For, if $T = S$, then (V, T) is a SUBDAG of G, and, if $T = A - S$, then also.* by Lemma 3.1(iii), (V, T) is a SUBDAG of G. From the definition of (V, T) it follows that $3w(T) \geq |S'|$. Hence, for any maximal SUBDAG (V', S') of G', we have $\frac{w(S_o)}{w(T)} \leq \frac{1}{2} \cdot \frac{|S'_o|}{w(T)} \leq \frac{3}{2} \cdot \frac{|S'_o|}{|S'|}$. □

From the above discussions and the Lemma 3.5, we conclude that:

Theorem 3.12 *MAX-LOP^p $\equiv_{AP}$ MAX-SUBDAG.*

3.3.3 On the Approximability of the Problems - A Review

The results of Subsection3.2.3 imply that MIN-LOP (MIN-LOP^p) and various equivalent problems as well as MAX-LOP (or MAX-LOP^p) and its equivalent problems have similar approximability properties with respect to their general formulations. However, some of these problems have restricted versions with possibly different properties. In this subsection, we briefly review some known results about the SUBDAG and feedback set problems and their restricted versions.

MAX-W-SUBDAG is known to be NP-complete even for digraphs with unit arc weight and with total degree of every vertex no more than 3 [67]. Papadimitriou and Yannakakis [132] shown that MAX-W-SUBDAG has a 2-approximation algorithm and is APX-complete. Several attempts have been made to improve on the known approximation bound 2 through various methodologies. One of them is studying polyhedral properties of relaxation of associated binary polytope [77,78,79]. In this approach, a linear relaxation of the associated integral polytope is considered and the optimal solution (may be fractional) of the linear relaxation is compared

with the actual integral optimal solution. The quality of a relaxation is measured by the integrality gap, which is the maximum possible ratio between the linear programming optimum and the true integral optimum. A well known linear program relaxation for MAX-SUBDAG and MAX-LOP is based on the idea that a solution contains at most $|C| - 1$ arcs from every dicycle C in the associated digraph. The corresponding linear constraints are exponential in n, but can be solved in polynomial time via an efficient separation oracle [126]. Another well known relaxation is due to Grötschel et. al. [78]. In this relaxation only two types of constraints are considered, one is for dicycles of length 2 and other is for dicycles of length 3. It is also known as generalized transitive tournament polytope [78,31] and is defined as $P_c^n = \{x \in \mathbb{R}^{A_n} | x_{ij}+x_{ji} = 1, \forall i \neq j; x_{ij}+x_{jk}+x_{ki} \geq 1, \forall \text{ distinct } i,j,k; x_{ij} \geq 0, \forall i \neq j\}$. Newman and Vempala [126] proved that the integrality gap for these two relaxations is at least $2 - \epsilon$, for any $\epsilon > 0$; implying that the linear optimum is no better than the trivial upper bound of total arc weight.

Grötschel et. al. [77,78,79] made attempts to strengthen these standard relaxations by adding constraints such as k-fence constraints and k-Möbius ladder constraints. Though k-fence constraints are NP-complete to separate [124], both k-fence, for fixed k, and k-Möbius constraints can be separable in polynomial time. Addition of these constraints does not yield better approximation bound and recently Newman and Vempala [126] proved that the integrality gap is very close to 2 even with these additional constraints.

As far as combinatorial algorithms are concerned for MAX-W-SUBDAG or MAX-LOP, Korte and Hauseman [107] proved that the greedy algorithm (i.e. construct a solution by repeatedly selecting the arc of maximum weight that does not form a dicycle with the already selected arcs) does not guarantee any fixed error ratio. Even randomized combinatorial algorithms do not give any better approximation bound than 2. Berger and Shor [21] gave a more involved randomized algorithm for MAX-SUBDAG. They note that, without loss of generality, one can assume that G has no

dicycle of length 2, since any bound can be achieved under this assumption can also be achieved without this assumption by a simple modification to the algorithm. Then, they develop an algorithm producing an acyclic subdigraph of at least $(\frac{1}{2}+\Omega(\frac{1}{\sqrt{d_{max}}}))|A|$ arcs where d_{max} is the maximum total degree of a vertex in G. Even when dicycle of length 2 exists, the solution contains at least a factor $(\frac{1}{2} + \Omega(\frac{1}{\sqrt{d_{max}}}))$ of the arcs in an optimal solution. The running time of their algorithm is $O(|A||V|)$. Then, Hassin and Rubinstein [86] achieved the same approximation bound for MAX-SUBDAG in time $O(|A| + d_{max}^3)$, which is better than $O(|A||V|)$ in some cases. All these attempts are failed to get an approximation algorithm better than the approximation bound 2. Towards the lower bound, Newman and Vempala [126] proved that it is not possible to approximate MAX-SUBDAG within a factor of $\frac{66}{65} + \epsilon$, for any $\epsilon > 0$, unless P=NP.

All the feedback set problems defined in the last section are known to be APX-hard for digraphs [98], whereas MIN-FVS for undirected graphs is APX-complete and approximable within a factor of 2, even if the vertices are weighted [13]. The first approximation algorithm for MIN-FAS was by Leighton and Rao [112] with approximation factor $O(\log^2 n)$ where n is the number of vertices in the input graph. This algorithm is based on an $O(\log n)$-approximation algorithm for a directed separator that splits the digraph into two approximately equal sized components. This result was improved by Seymour [142], who gave an $O(\log n \log\log n)$ approximation algorithm based on a linear relaxation of an integer linear program formulation of MIN-FAS. Latter, Even et.al. [53] designed an $O(\log n \,\mathrm{loglog}\, n)$-approximation algorithm for a generalization of MIN-W-FAS, called MIN-Subset-FAS, which is an extension of Seymour's algorithm for MIN-FAS. The same algorithm is also an $O(\log n \,\mathrm{loglog}\, n)$-approximation algorithm for MIN-Subset-FVS for weighted digraphs, whereas for undirected graphs it is approximable within a factor of 2 [12,13,39].

Both MAX-W-SUBDAG and MIN-W-FAS are solvable in polynomial time, when the underlying digraph is planar [115] and are can be solved in $O(n^3)$ time [65]. They are also solvable in polynomial time for more general class of $K_{3,3}$-free digraphs [135]

and classes of reducible flow digraphs [140] and weakly acyclic digraphs [78]. But MIN-FVS is known to be NP-complete for planar digraphs even if total degree of a vertex is at most 3 [67] and has an $\frac{9}{4}$-approximation algorithm [72]. Also MIN-W-FAS has a polynomial time exact algorithm for reducible flow digraphs [140]. For tournaments MIN-W-FVS is known to be APX-complete [146] and has a 2.5-approximation algorithm [33]; whereas it is not known whether MIN-FAS is NP-complete for tournaments [16]. For a detailed survey on various applications, brief description of known approximation algorithms and exact algorithms for specific graphs we refer to the survey article by Festa et. al. [59].

3.4 List of the Problems

The precise formulations of some of the problems we deal with are given below where we have ommited the goal as it can be inferred from the name of the problem.

MIN-LOP (Minimum Linear Ordering Problem)

Instance - A pair $x = (G_n, w)$, where $G_n = (V, A_n)$ is a complete digraph on V and w assigns a nonnegative integer to any $e \in A_n$.

Solution - An acyclic tournament y on V.

Cost - $m(x, y) = \max\{1, \sum_{e \in y} w(e)\}$.

MAX-LOP (Maximum Linear Ordering Problem)

Instance - A pair $x = (G_n, w)$, where $G_n = (V, A_n)$ is a complete digraph on V and w assigns a nonnegative integer to any $e \in A_n$.

Solution - An acyclic tournament y on V.

Cost - $m(x, y) = \max\{1, \sum_{e \in y} w(e)\}$.

MAX-W-SUBDAG (Maximum Weight Acyclic Subdigraph)

Instance - A pair $x = (G, w)$, where $G = (V, A)$ is a digraph and w assigns a nonnegative integer to each $e \in A$.

Solution - A SUBDAG (V, B) of G.

Cost - $m(x, B) = \max\{1, \sum_{e \in B} w(e)\}$.

MIN-W-FAS (Minimum Weight Feedback Arc Set)

Instance - A pair $x = (G, w)$, where $G = (V, A)$ is a digraph and w assigns a nonnegative integer to each $e \in A$.

Solution - A FAS F of G.

Cost - $m(x, B) = \max\{1, \sum_{e \in F} w(e)\}$.

MIN-W-FVS (Minimum Weight Feedback Vertex Set)

Instance - A pair $x = (G, w)$ where G is a graph/digraph and w assigns a nonnegative integer to each $v \in V$.

Solution - A FVS F of G.

Cost - $m(x, F) = \max\{1, \sum_{v \in F} w(v)\}$.

MIN-W-MAX-SUBDAG (Minimum Weight Maximal Acyclic Subdigraph)

Instance - Same as that of MAX-W-SUBDAG.

Solution - A maximal SUBDAG (V, B) of G.

Cost - $m(x, B) = \max\{1, \sum_{e \in B} w(e)\}$.

MAX-W-MIN-FAS (Maximum Weight Minimal Feedback Arc Set)

Instance - Same as that of MAX-W-SUBDAG.

Solution - A minimal FAS F of G.

Cost - $m(x, B) = \max\{1, \sum_{e \in F} w(e)\}$.

MAX-W-MIN-FVS (Maximum Weight Minimal Feedback Vertex Set)

Instance - Same as that of MIN-FVS.

Solution - A minimal FVS F of G.

Cost - $m(x, B) = \max\{1, \sum_{i \in F} w(i)\}$.

MIN-VC (Minimum Vertex Cover)

Instance - A graph $x = G = (V, E)$.

Solution - A VC C of G.

Cost - $m(x, C) = \max\{1, |C|\}$.

MAX-MIN-VC (Maximum Cardinality Minimal Vertex Cover)

Instance - A graph $x = G = (V, E)$.

Solution - A minimal VC C of G.

Cost - $m(x, C) = \max\{1, |C|\}$.

OEOP (Optimal Evaluation Ordering Problem)

Instance : n arithmetic expressions $F_1, F_2, \ldots, F_n$

Solution : A permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ of $\{1, 2, \ldots, n\}$

Cost : $w(\pi) = |\cup_{i=1}^{n-1} \cup_{j=i+1}^{n} \mathrm{SPILL}(F_{\pi_i}, F_{\pi_j})|$

OEOP(S) is the problem of OEOP whose instances satisfy the simple sharing assumption (deined in Section 5.1.1).

MIN-W-k-FAS (Minimum Weight k-FAS)

Instance - $x = (G, w, k)$, where $G = (V, A)$ is a digraph, w assigns a non-negative arc weight to each arc $e \in A$ and k is a positive integer.

Solution - A FAS $F \subseteq A$ of G such that $|F| = k$.

Cost - $m(x, F) = \max\{1, \sum_{e \in F} w(e)\}$

MIN-Subset-FAS (Minimum Weight Subset Feedback Arc Set)

Instance - $x = (G, w, X)$, where $G = (V, A)$ is a digraph, w assigns a non-negative arc weight to each arc in A and $X \subseteq V \cup A$.

Solution - A FAS $F \subseteq A$ such that it intersects with all dicycles those intersects with the set X.

Cost - $m(x, F) = \mathrm{man}\{1, \sum_{e \in F} w(e)\}$

MIN-Subset-FVS (Minimum Weight Subset Feedback Vertex Set)

Instance - $x = (G, w, X)$, where $G = (V, A)$ is a digraph, w assigns a non-negative weight to each vertex in V and $X \subseteq V \cup A$.

Solution - A FVS $F \subseteq V$ such that it intersects with all dicycles those intersects with the set X.

Cost - $m(x, F) = \mathrm{man}\{1, \sum_{i \in F} w(i)\}$

MIN-Ones (Minimum Number of Ones)

Instance - A 3CNF formula ϕ with variable set U

Solution - A subset S of U such that ϕ is satisfied when the variables in S are set to 1 and the variables in $U - S$ are set to 0

Cost - $m(\phi, S) = \max\{1, |S|\}$

MIN-Dom-Set (Minimum Dominating Set)

Instance - A graph $x = G = (V, E)$

Solution - A subset S of V such that $S \cap N[v] = \phi$, for all $v \in V$

Cost - $m(x, S) = \max\{1, |S|\}$

MIN-QAP (Minimum Quadratic Assignment Problem)

Instance - A pair of non-negative integral matrices $(W_{n\times n}, D_{m\times m})$ with $n \geq m$, $w_{ii} = 0$ for $1 \leq i \leq n$, and $d_{jj} = 0$ for $1 \leq j \leq m$.

Solution - A binary $n \times m$ matrix $X = (x_{ij})$ such that

$$\sum_{j=1}^{m} x_{ij} \leq 1, \quad \text{for } 1 \leq i \leq n$$

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

Cost - $m(W, D, X) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} \sum_{l=1}^{m} w_{ij} d_{kl} x_{ik} x_{jl}$

MIN-QAP(S) is the problem of MIN-QAP whose instances (W, D) satisfy the conditions (i) both W and D are $n \times n$ matrices, for some positive integer n, and (ii) $d_{ij} = 1$, if $i < j$, and $d_{ij} = 0$, if $i \geq j$.

MAX-SUBDAG-k is the problem of MAX-SUBDAG on k-total-regular digraphs. Similarly, MIN-FAS and MIN-FAS-k are defined. MIN-FVS is the unweighted version of MIN-W-FVS, i.e. MIN-W-FVS with $w(v) = 1$ for each $v \in V$. MIN-FVS-k is the problem of MIN-FVS on k-regular (respectively, k-total-regular) graphs (respectively, digraphs). MIN-MAX-SUBDAG (respectively, MAX-MIN-FAS) is the unweighted version of MIN-W-MAX-SUBDAG (respectively, MAX-W-MIN-FAS), i.e. MIN-W-MAX-SUBDAG (respectively, MAX-W-MIN-FAS) with $w(e) = 1$ for each $e \in A$. MIN-MAX-SUBDAG$\leq$k (respectively, MAX-MIN-FAS$\leq$k) is the problem of MIN-MAX-SUBDAG

(respectively, MAX-MIN-FAS) on digraphs of total degree at most k. MAX-MIN-FVS-k is the problem of MAX-MIN-FVS on k-regular (respectively, k-total-regular) graphs (respectively, digraphs) and MAX-MIN-FVS$\leq$k is the problem of MAX-MIN-FVS on graphs (respectively, digraphs) of degree (respectively, total-degree) at most k. MAX-MIN-VC$\leq$k is the problem of MAX-MIN-VC on graphs of degree at most k.

Chapter 4

Approximability of LOP

In Section 3.3, we have seen how MAX-LOP and MIN-LOP are related to several other well known NP-hard optimization problems and have similar approximability properties. In particular, we noted that (1) MAX-LOP is APX-complete and has a 2-approximation algorithm, (2) MIN-LOP has an $O(\log n \log\log n)$-approximation algorithm and is APX-hard and (3) LOP $\notin$ FPTAS, if P$\neq$NP. Whether MIN-LOP and other equivalent problems are in APX, if P$\neq$NP, is a major open problem. In this chapter, we primarily deal with LOP and examine (1) how best LOP can be solved exactly, (2) what evidences can we provide for or against the hypothesis that MIN-LOP$\notin$APX, if P$\neq$NP, and (3) how good approximate solutions, if any, can be obtained for LOP, if the weight function is restricted suitably.

The main results and the organization of this chapter are as follows. In Section 4.1, we present a dynamic programming algorithm for LOP with time complexity $O(n^2 2^n)$. In Section 4.2, after noting that LOP does not have any polynomial-time absolute approximation algorithm, if P$\neq$NP, we address the question whether MIN-LOP can be in APX. While all our efforts to show that MIN-LOP$\in$APX have failed, we have some evidences, though not strong, that appear to suggest that MIN-LOP and other equivalent problems may not be in APX, if P$\neq$NP. In particular, in Section 4.2.1, we show that, if non-zero coordinates of the extreme points of the linear ordering polytope are all greater than $\frac{1}{k}$, for some constant $k > 1$, then by rounding an optimal solution of an LP-relaxation of MIN-LOP, we get a k-approximate solution of MIN-LOP. But,

as Nutov and Penn [127] have shown, the non-zero coordinates can be arbitrarily close to 0, and this leads us to believe that we may not be able to get a constant-factor approximation algorithm for MIN-LOP via a LP-relaxation and rounding method. In Section 4.2.2, we show that for MIN-Subset-FAS, a generalization of MIN-FAS, there cannot exist a polynomial-time $(1-\epsilon)\log n$-approximation algorithm, for any $\epsilon > 0$, unless NP$\subset$DTIME($n^{\log\log n}$). We also make some observation regarding a possibility of showing that MIN-LOP and related problems are not in APX. In Section 4.2.3, we show that MIN-W-k-FAS is NPO-complete and, since MIN-W-k-FAS is MIN-W-FAS with restrictions just on cardinalities of the feasible solution, this also seems to suggest that MIN-W-FAS is hard to approximate.

Next, in Section 4.3, we consider LOP with some restrictions on weight function and show that, if the weight function satisfies a kind of triangle inequality, that is stronger than the usual triangle inequality, then a simple heuristic, as suggested in [132] for MAX-SUBDAG, adapted for LOP, called CT-heuristic, yields good approximate solution. In particular, this is a $\frac{4}{3}$-approximation algorithm for $\triangle$-MAX-LOP and a $\frac{3}{2}$-approximation algorithm for $\triangle$-MIN-LOP. In Section 4.3, we extend quite easily the analysis of the performance of CT-heuristic for the case when the arc weights satisfy parametrized triangle inequality (or $\triangle_t$-inequality, for $t \in (0,2]$), and show that the CT heuristic is a $\frac{2+t}{2t}$-approximation algorithm for $\triangle_t$-MIN-LOP and a $\frac{4}{2+t}$-approximation algorithm for $\triangle_t$-MAX-LOP. Further, $\triangle_t$-LOP$\in$PO if and only if $t = 2$. Finally, in Section 4.5, we show that $\triangle_t$-LOP, for $0 < t < 2$, is not in PTAS, unless P=NP.

4.1 Complexity of an Exact Algorithm for LOP

As already noted in Section 3.3.1, a naive algorithm for LOP has time complexity $O(n^2 n!)$. In this section, we present an algorithm based on dynamic programming and show that it has time complexity $O(2^n n^2)$ and space complexity $O(n2^n)$ as mentioned earlier.

A dynamic programming algorithm for LOP can be designed similar to that for

MIN-TSP due to Held and Karp [91]. We will describe the algorithm only for MIN-LOP as the same algorithm, with minor modification, works for MAX-LOP as well.

In order to formulate the dynamic programming algorithm for MIN-LOP, we need to identify suitable subinstances of any instance of MIN-LOP and relate the optimal solution of the instance with the optimal solutions of the sub-instances. Towards this, let $(G_n = (V, A_n), w)$ be an instance of MIN-LOP, where $V = \{1, 2, \ldots, n\}$. the appropriate sub-instances of MIN-LOP corresponding to the instance (G_n, w) are of the form $(G_n[S], w_S)$, where S is any non-empty subset of V, $G_n[S] = (S, A_S)$ is the complete digraph on S and w_S is the weight function w restricted to A_S. We also need to adopt some notations concerning solutions and weights of the solutions of the subinstances of (G_n, w). If $S = \{i_1, i_2, \ldots, i_{j-1}\} \subseteq V$, let *opt-wt*$(S)$ be the weight of a minimum weight acyclic tournament of $(G_n[S], w_S)$, and let *opt-ord*(S) be the ordering of S corresponding to a minimum weight acyclic tournament on S as given by Proposition 3.1. Further, let $ord(S) = (i_1, i_2, \ldots, i_{j-1})$ be any ordering of S and let $wt(ord(S))$ denote the weight of the corresponding acyclic tournament on S, i.e., let $wt(ord(S)) = wt(i_1, i_2, \ldots, i_{j-1}) = \sum_{1 \leq k < l \leq j-1} w_{i_k i_l}$. Also, if $k \in V - S$, let $ord(S/\!/k)_l$ denote the ordering of $S \cup \{k\}$ obtained from the ordering $ord(S)$ by inserting k in the lth slot among the j possible slots with respect to the ordering $ord(S) = (i_1, i_2, \ldots, i_{l-1}, i_l, \ldots, i_{j-1})$ i.e., $ord(S/\!/k)_l = (i_1, i_2, \ldots, i_{l-1}, k, i_l, \ldots, i_{j-1})$. Let

$$wt(S/\!/k) = \min_{1 \leq l \leq j} wt(opt\text{-}ord(S/\!/k)_l) \tag{4.1}$$

and *opt-ord*$(S/\!/k)$ = *opt-ord*$(S/\!/k)_l$, if in (4.1) equality holds for l. By convention, we let, for $|S| = 1$, *opt-wt*$(S) = 0$, and *opt-ord*$(S) = (i)$, if $S = \{i\}$.

Lemma 4.1 *Let $(G_n = (V, A_n), w)$ be an instance of MIN-LOP and $\phi \neq S \subseteq V$. Let $S_k = S - \{k\}$, for $k \in S$. Then, given opt-wt(S_k) and opt-ord(S_k), for all $k \in S$, we have :*

(1) $$opt\text{-}wt(S) = min_{k \in S} wt(S_k/\!/k) \tag{4.2}$$

(2) *opt-ord*$(S) =$ *opt-ord*$(S_l /\!/ l)$, *if in* (4.2) *equality holds for* $k = l$. *Further,*

(3) *opt-wt*(S) *and opt-ord*(S) *can be computed in time* $O(|S|^2)$.

Proof (1) Claerly, *opt-wt*$(S) \leq \min_{k \in S} wt(S_k /\!/ k)$ as the left hand side is the minimum over all $|S|!$ permutations of S but the right hand side corresponds to the minimum over $|S|^2$ permutations of S.

Now, if possible, let *opt-wt*$(S) < \min_{k \in S} wt(S_k /\!/ k)$. Let $S = \{i_1, i_2, \ldots i_j\}$ and, without loss of generality, let *opt-ord*$(S) = (i_1, i_2, \ldots i_j)$. Then, $wt($*opt-ord*$(S)) = wt(i_1, i_2, \ldots i_j) < \min_{k \in S} wt(S_k /\!/ k) \leq wt(S_{i_1} /\!/ i_1) \leq wt($*opt-ord*$(S_{i_1} /\!/ i_1)_1)$. So, $\sum_{k=1}^{j} w_{i_1 i_k} + wt(i_2, i_3, \ldots, i_j) < \sum_{k=1}^{j} w_{i_1 i_k} +$ *opt-wt*(S_{i_1}). So $wt(i_2, i_3, \ldots, i_j) <$ *opt-wt*(S_{i_1}), which is a contradiction. Thus the result follows.

(2) Trivial.

(3) It is enough to show that, for any $k \in S$, $wt(S_k /\!/ k)$ can be computed in $O(|S|)$ time. Let $S_k = \{i_1, i_2, \ldots i_{j-1}\}$ and, without loss of generality, *opt-ord*$(S_k) = (i_1, i_2, \ldots i_{j-1})$. Then $wt($*opt-ord*$(S_k /\!/ k)_1) =$ *opt-wt*$(S_k) + \sum_{l=1}^{j-1} w_{k i_l}$, which can be computed in $O(j)$ time. Further, given $wt($*opt-ord*$(S_k /\!/ k)_l)$, $1 \leq l \leq j-1$, we can compute $wt($*opt-ord*$(S_k /\!/ k)_{l+1})$ in constant time, as $wt($*opt-ord*$(S_k /\!/ k)_{l+1}) = wt($*opt-ord*$(S_k /\!/ k)_l) - w_{k i_l} + w_{i_l k}$. □

The lemma suggest the following

Algorithm 4.1 Dynamic Programming Algorithm for MIN-LOP

Input - $(G_n = (V, A_n), w)$;

Output - *opt-ord*(V) and *opt-wt*(V).

1. **for** $k = 1$ to n **do**

   opt-wt$(\{k\}) = 0$;

   opt-ord$(\{k\}) = (k)$;

   od;

2. **for** $j = 2$ to n **do**

for each $S \subseteq V$ with $|S| = j$ **do**

for each $k \in S$ **do**

$S_k = S - k$;

$wt(S_k /\!\!/ k) = \min_{1 \le l \le j} wt(opt - ord(S_k /\!\!/ k)_l) = wt(opt - ord(S_k /\!\!/ k)_m)$ (say);

$opt\text{-}ord(S_k /\!\!/ k) = opt\text{-}ord(S_k /\!\!/ k)_m$;

od;

$opt\text{-}wt(S) = \min_{k \in S} wt(S_k /\!\!/ k) = wt(S_p /\!\!/ p)$ (say);

$opt\text{-}ord(S) = opt\text{-}ord(S_p /\!\!/ p)$;

od;

od;

4. **return**$(opt\text{-}wt(V), opt\text{-}ord(V))$.

Regarding the complexity of the above algorithm, we have :

Proposition 4.1 *For any instance (G_n, w) of MIN-LOP, Algorithm 4.1 obtains an optimal solution and its value in time $O(n^2 2^n)$ and space $O(n2^n)$.*

Proof From Lemma 4.1, $opt\text{-}wt(S)$ and $opt\text{-}ord(S)$ can be computed in time $O(|S|^2)$, for every $S \subseteq V$. So, the total time is $O(\sum_{j=1}^{n} \binom{n}{j} j^2) = O(n^2 \sum_{j=2}^{n} \binom{n}{j}) = O(n^2 2^n)$. Similarly, the space complexity is $O(n2^n)$, as we need to store $opt\text{-}wt(S)$ and $opt\text{-}ord(S)$, for each $S \subseteq V$, which needs $O(|S|)$ space. □

4.2 Possibly MIN-LOP is not in APX

In this section, we examine whether MIN-LOP and equivalent feedback set problems are in APX. Our efforts towards a positive answer have failed but, based on our expe-rience with various approaches towards a positive answer and some indirect evidences gathered, as mentioned in the introduction of this chapter, we are led to believe and *conjecture* that MIN-LOP and the equivalent feedback set problems cannot be in APX, if P≠NP.

Before we take up the main topics of this section, we may also take note of the following easy negative result about absolute approximation of MIN-LOP.

Theorem 4.1 *If P≠NP, then there exists no polynomial-time absolute approximation algorithm for MIN-LOP.*

Proof To prove this theorem we show that, if MIN-LOP has an absolute approximation algorithm, then MIN-LOP can be solved in polynomial time. Since MIN-LOP is NP-hard, the theorem follows.

Assume that there is a polynomial-time algorithm $\mathcal{A}$ such that $|m^*(x) - m(x, \mathcal{A}(x))| \leq q$, for every instance $x = (G_n = (V, A_n), w)$ of MIN-LOP and a fixed q. Let $x' = (G_n, w')$ be another instance of MIN-LOP with $w'(e) = (q+1)w(e)$, for all $e \in A_n$. Clearly, x and x' have the same set of feasible solutions. Further, $m^*(x') = (q+1) \cdot m^*(x)$, and both x and x' have the same set of optimal solutions. Also since, arc weights are integral, all feasible solutions to x' have value $m^*(x')$ or have value at least $m^*(x') + (q+1)$. Hence, if $|m^*(x) - m(x, \mathcal{A}(x))| \leq q$ then $m^*(x') = m^*(x', y)$. Thus, algorithm $\mathcal{A}$ can be used to obtain an optimal solution for x' and hence for x. Since the length of x' is at most $|x| \log q$, it follows that using the above construction we can obtain a polynomial-time algorithm for MIN-LOP. □

Similar result holds for MAX-LOP also.

4.2.1 On the Approximability of MIN-LOP via Linear Program Relaxation

First we briefly review some concepts and results related to linear ordering polytope. Recall Theorem 3.1 which identifies acyclic tournaments with transitive tournaments. The generalized transitive tournament polytope of G_n [79,31] is defined as the set of all vectors $x \in \mathbb{R}^{A_n}$, $n \geq 3$, satisfying

$$x_{ij} + x_{ji} = 1, \quad i \neq j \tag{4.3}$$

$$x_{ij} + x_{jk} + x_{ki} \geq 1, \quad i,j,k \text{ distinct} \tag{4.4}$$

$$x_{ij} \geq 0, \quad i \neq j. \tag{4.5}$$

Following [79], we shall denote the generalized transitive polytope by P_c^n. It can be seen easily that an integral vector $x \in P_c^n$ is binary. Also a binary vector $x \in P_c^n$ if and only if x is an incidence vector of an acyclic tournament of G_n. It is known that $P_{LO}^n \subseteq P_c^n$ and the integral extreme points of P_c^n coincide with those of P_{LO}^n [79]. A non-integral extreme point of P_c^n, for $n = 6$, was first given by Cruse [45] with values $\{0, \frac{1}{2}, 1\}$. Dridi [49] showed that $P_{LO}^n = P_c^n$, for $n \leq 5$, while $P_{LO}^n \subset P_c^n$, for $n > 5$. Brualdi and Hwang [31] gave a partial characterization of the extreme points of P_c^n with values $\{0, \frac{1}{2}, 1\}$ and latter Borobia [27] gave a complete characterization of such extreme points. Then Nutov and Penn [127] came with a method of finding extreme points of P_c^n with values not in $\{0, \frac{1}{2}, 1\}$, for $n \geq 8$, and showing that the smallest positive value can be arbitrary close to 0 as n increases. But it is not known whether P_c^n, for $n = 6, 7$, has such extreme points. Their main result is the following.

Theorem 4.2 (Nutov-Penn [127]) *P_c^n, for $n \geq 8$, has extreme points with values not in $\{0, \frac{1}{2}, 1\}$. Moreover, P_c^n has extreme points having values $\frac{1}{\lfloor \frac{n}{2} \rfloor - 1}$, for $n \geq 8$.*

We shall use Nutov and Penn's result to suggest that a rounding algorithm via LP-relaxation may not yield a constant approximation algorithm for MIN-LOP. Towards this, we prove some results connecting extreme points of P_c^n and approximate solutions of MIN-LOP obtained via LP-relaxation.

Lemma 4.2 *Let x be an extreme point of P_c^n. If there exists a positive integer k such that either $x_{ij} = 0$ or $x_{ij} \in [\frac{1}{k}, 1]$, for all pairs of vertices $i, j \in V$, then $T_x = \{(i,j) | x_{ij} > 0\}$ contains an acyclic tournament.*

Proof. Since x satisfies (4.3), (4.4) and (4.5), $x_{ij} + x_{ji} = 1$, for every pair of vertices $i, j \in V$, T_x contains at least one arc from (i,j) and (j,i), for each pair of vertices $i, j \in V$. Hence, T_x contains a tournament.

Next we show that T_x contains an acyclic tournament. To show this, it is sufficient to prove that $T_x^c = A_n - T_x = \{(i,j)|x_{ij} = 0\}$ is acyclic. This is because, if T_x^c is acyclic it can be extended to an acyclic tournament, say T; now T^c, the complement of T, is contained in T_x and T^c is acyclic as T is acyclic.

Lemma 4.3 *Let x be an optimal solution of the linear program (LP): Minimize $\sum_{(i,j)\in A_n} w_{ij}x_{ij}$ over $x \in P_c^n$. If k is the smallest positive integer such that either $x_{ij} = 0$ or $x_{ij} \in [\frac{1}{k}, 1]$, for all $i, j \in V$, then any acyclic tournament contained in T_x is a k-approximate solution of MIN-LOP for the instance (G_n, w), where T_x is as defined in Lemma 4.2.*

Proof. Let T_o be a minimum weight acyclic tournament of the instance (G_n, w). Let T be any acyclic tournament contained in T_x. Hence $w(T) \leq w(T_x)$. Now

$$\begin{aligned} w(T) = \sum_{(i,j)\in T} w_{ij} &\leq k \sum_{(i,j)\in T} w_{ij}x_{ij} \quad (\text{as } x_{ij} \geq \tfrac{1}{k} \text{ for } (i,j) \in T) \\ &\leq k \sum_{(i,j)\in T_x} w_{ij}x_{ij} \quad (\text{as } T \subseteq T_x) \\ &\leq kw(T_o) \quad (\text{as } w(T_x) \leq w(T_o)) \end{aligned}$$

Hence T is a k-approximate solution. □

However, the converse of Lemma 4.3 is not known to hold, i.e., we do not know whether it is true that, if any acyclic tournament contained in T_x is k-approximate, then $x_{ij} = 0$ or $x_{ij} \in [\frac{1}{k}, 1]$, where x is defined as in Lemma 4.3. If the converse would hold, then by Theorem 4.2 and Lemma 4.3 we could definitely say that there exists no k-approximation algorithm for MIN-LOP via a rounding algorithm on an extreme point of P_c^n. Yet, Theorem 4.2 and Lemma 4.3 seem to suggest that possibly one may not be able to get a constant approximation algorithm for MIN-LOP via LP relaxation.

4.2.2 Inapproximability of MIN-Subset-FAS

Next we shall consider inapproximability of a generalization of MIN-FAS, called MIN-Subset-FAS. The problems MIN-Subset-FAS and MIN-Subset-FVS are introduced by

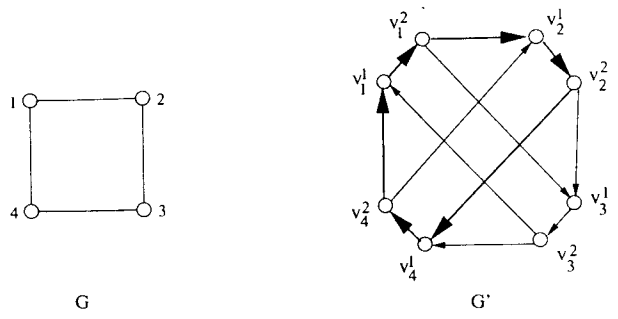


Figure 4.1: Graph G and the corresponding digraph G'

Even et.al. [53] and in these problems only a subset of dicycles are considered which are called *interesting*. A FAS F with respect to the interesting dicycles is an arc set that intersects every interesting dicycle. Even et.al. characterized the set of interesting dicycles by a set $X \subseteq V \cap A$, i.e. given such a set X, the set of interesting dicycles characterized by X is the set of all cycles that intersect X. MIN-Subset-FAS can be approximated within a factor of $O(\log\log n \log n)$ [53] and it is the best known result. Here, we will show that it cannot be approximated within a factor of $(1-\epsilon)\log n$, for any $\epsilon > 0$, unless NP$\subset$DTIME($n^{\log\log n}$), even if $O(n)$ interesting dicycles are given explicitly. Although this result does not imply, yet one may expect that MIN-FAS may not be approximable within a factor of $\log n$. We have:

Theorem 4.3 *For any $\epsilon > 0$, there does not exist any polynomial time approximation algorithm solving MIN-Subset-FAS and MIN-Subset-FVS within a factor of $(1-\epsilon)\log n$, unless* NP$\subset$DTIME($n^{\log\log n}$), *even if the number of interesting dicycles is of $O(n)$ and are given explicitly.*

Proof. It is enough to show that MIN-Dom-Set$\leq_{strict}$MIN-Subset-FAS, as from [58] it is known that, for any $\epsilon > 0$, MIN-Dom-Set cannot be approximated within a factor of $(1-\epsilon)\log n$, unless NP$\subset$DTIME($n^{\log\log n}$).

Let $G = (V, E)$ be an instance of MIN-Dom-Set. We assume that the vertices in V are ordered and let $V = \{1, 2, \ldots, n\}$. We also assume that, for each $i \in V$, the vertices in the closed neighborhood of i, $N[i]$, are also ordered with respect to the order-

ing in V. From G, we construct an instance $(G' = (V', A'), w, \mathcal{C})$ of MIN-Subset-FAS (where $\mathcal{C}$ is the set of interesting dicycles) as follows. Let $V' = \cup_{i \in V}\{v_i^1, v_i^2\}$ and $A' = [\cup_{i \in V}\{(v_i^1, v_i^2)\}] \cup \left[\cup_{i \in V}\left\{\{(v_{i_1}^2, v_{i_2}^1), (v_{i_2}^2, v_{i_3}^1), \ldots, (v_{i_{k-1}}^2, v_{i_k}^1), (v_{i_k}^2, v_{i_1}^1)\} : N[i] = \{i_1, i_2, \ldots, i_k\}\right\}\right]$. For an example see Fig. 4.1. Each arc in the set $\cup_{i \in V}\{(v_i^1, v_i^2)\}$ has weight 1 and all other arcs have weight $2n$ each. From the construction, it is clear that for each set $N[i]$, $i \in V$, G' has the dicycle $(v_{i_1}^1, v_{i_1}^2, v_{i_2}^1, \ldots, v_{i_k}^1, v_{i_k}^2, v_{i_1}^1)$. We shall denote this dicycle by C_i. For example, in Fig. 4.1, the dicycle in G', consisting of thick arrowed arcs, corresponds to the closed neighborhood $N[1]$ of the vertex 1 in G. Finally, let the set of interesting dicycles $\mathcal{C} = \{C_i | i \in V\}$.

Observe that in G', for any $i \in V$, the vertex v_i^1 has only one out going arc viz., (v_i^1, v_i^2) and the vertex v_i^2 has only one incoming arc viz., (v_i^1, v_i^2). From this it follows that, if any FAS F of G' contains an arc of the form (v_p^2, v_q^1), for some $p, q \in V$, then $F' = F - (v_p^2, v_q^1) + (v_q^1, v_q^2)$ is a FAS of G' and $w(F') < w(F)$. This is because, any dicycle containing the arc (v_p^2, v_q^1) must contain the arc (v_q^1, v_q^2) and the arc (v_p^2, v_q^1) has larger weight than (v_q^1, v_q^2). Since MIN-Subset-FAS is a minimization problem, without loss of generality, we can assume that

(*) any FAS of G' contains only arcs of the form (v_i^1, v_i^2), for $i \in V$.

It is not difficult to see that if $S \subseteq V$ is a dominating set of G then the set $F_S = \{(v_i^1, v_i^2) | i \in S\}$ is a FAS with respect to the set of interesting dicycles $\mathcal{C}$ and $|S| = w(F_S)$. Conversely, if F is a FAS of G' with respect to the interesting dicycle set $\mathcal{C}$, then $S_F = \{i | (v_i^1, v_i^2) \in F\}$ is a dominating set of G, because, by (*), if $(v_k^1, v_k^2) \in F \cap C_i$, then $k \in N[i] \cap S_F$. Hence, if F_o is a minimum weight FAS of G' with respect to the set $\mathcal{C}$ then, the associated set S_o is a minimum dominating set of G and $w(F_o) = |S_o|$. *It follows that* $\frac{|S_F|}{|S_o|} = \frac{w(F)}{w(F_o)}$. *From this it follows that* MIN-Dom-Set$\leq_{strict}$MIN-Subset-FAS. □

Remark 4.1 A natural question that arises is: Is MIN-Dom-Set$\leq_{AP}$MIN-W-FAS? The above reduction (without the set $\mathcal{C}$ of interesting dicycles) would yield a positive

answer if $w(F_o) \leq c|S_o|$, where F_o is a minimum weight FAS of (G', w), S_o is a minimum dominating set of G and c is a positive constant. But we have not been able to show this or construct any other AP-reduction. Thus this question remains open. A positive answer to this would establish that MIN-LOP and other related problems are not in APX.

4.2.3 NPO-completeness of MIN-W-k-FAS

In this subsection, we shall consider a generalization of MIN-LOP, called MIN-W-k-FAS, and shall show that it is NPO-complete. In MIN-W-k-FAS, given an instance (G, w, k), where G is a digraph with arc weight function w and a positive integer k, it is required to find a minimum weight FAS F with $|F| = k$. In this problem, FASs of cardinality k are the feasible solutions. MIN-LOP is same as MIN-W-k-FAS when $k = \frac{1}{2}n(n-1)$ and the input graph is a complete digraph. While the unweighted version of MIN-LOP is in the class PO, the same is not true for MIN-W-k-FAS. In fact, just like MIN-FAS, it is NP-complete. To show that MIN-W-k-FAS is NPO-complete, we shall adapt the reduction from 3SAT to MAX-SUBDAG, used by Newman [126] to show that MAX-SUBDAG is not approximable within a factor of $\frac{66}{65} + \epsilon$, for any $\epsilon > 0$, unless P=NP. This is the first lower bound known for MAX-SUBDAG problem, though it can be approximated within a factor of 2 by a simple algorithm [132].

It is known that MIN-Ones is NPO-complete [99, 43]. To show that MIN-Ones is NPO-complete, Kann reduced MIN-IND-DOM-SET (*minimum independent dominating set*) to MIN-Ones (Theorem 4 and Theorem 5 in [99]). In this reduction, the constructed instance ϕ of MIN-Ones has the property that each variable in ϕ appears at least once negated and at least once unnegated. We shall denote the restricted version of MIN-Ones in which each variable in an instance ϕ appears at least once negated and once unnegated, as MIN-Ones+. By Kann's reduction, it follows that MIN-Ones+ is NPO-complete. We next prove, by a reduction from MIN-Ones+, the following:

Theorem 4.4 *MIN-W-k-FAS is* NPO*-complete.*

Proof. Let ϕ be an instance of MIN-Ones+, with m clauses and n variables. From ϕ we shall construct an arc weighted digraph $(G(\phi), w)$ as follows:

1 For any variable $x \in \phi$, let $l(x)$ and $l(\overline{x})$ denote the number of times the literal x and $\overline{x}$ appear in ϕ, respectively. To each variable x in ϕ, construct a variable gadget $G(x)$ with $l(x) + l(\overline{x}) + 2$ vertices and $2(l(x) + l(\overline{x}))$ arcs as follows. The vertex set in $G(x)$ is $\{x_1, x_2\} \cup \{x^1, x^2, \ldots, x^{l(x)}\} \cup \{\overline{x}^1, \overline{x}^2, \ldots \overline{x}^{l(\overline{x})}\}$, and the arc set is $\left[\cup_{i=1}^{l(x)}\{(x_1, x^i), (x^i, x_2)\}\right] \cup \left[\cup_{i=1}^{l(\overline{x})}\{(x_2, \overline{x}^i), (\overline{x}^i, x_1)\}\right]$. For an example, see Fig 4.2(a).

2 For each clause $C \in \phi$, we construct a clause gadget $G(C)$ which consists of a dicycle of length 6 and whose alternate arcs are labelled by distinct literals from C. For example, if $C = (x \vee y \vee \overline{z})$ then $G(C)$ is as shown in Fig. 4.2(b).

3 Each clause gadget $G(C)$ is linked with the variable gadgets corresponding to the literals in C as follows:

- For an arc (i, j) labeled x in the clause gadget, add an arc from vertex j to x_1 and arc from x_2 to i.
- For an arc (i, j) labeled $\overline{x}$ in the clause gadget, add an arc from vertex j to x_2 and an arc from x_1 to i.

$G(\phi)$ has the vertex set V consisting of union of the vertices in the clause gadgets and variable gadgets and the arc set A consists of all the arcs in clause gadgets, vertex gadgets and the arcs linking clause gadgets with vertex gadgets. Fig. 4.2 illustrates the digraph $G(\phi)$ for $\phi = (x \vee y \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee y \vee z)$. Note that $G(\phi)$ has $18m$ arcs as there are $6m$ in all clause gadgets, $6m$ arcs joining clause gadgets with variable gadgets and $6m$ arcs in all variable gadgets.

Now, we define the arc weight function w. The weight of each arc labelled by x in a clause gadget is $\frac{n+1}{l(x)+l(\overline{x})}$ and weight of an arc labelled $\overline{x}$ in a clause gadget is $\frac{1}{l(x)+l(\overline{x})}$. For each variable x, the arcs in the set $\cup_{i=1}^{l(x)}\{(x_1, x^i), (x^i, x_2)\}$ (in the variable gadget)

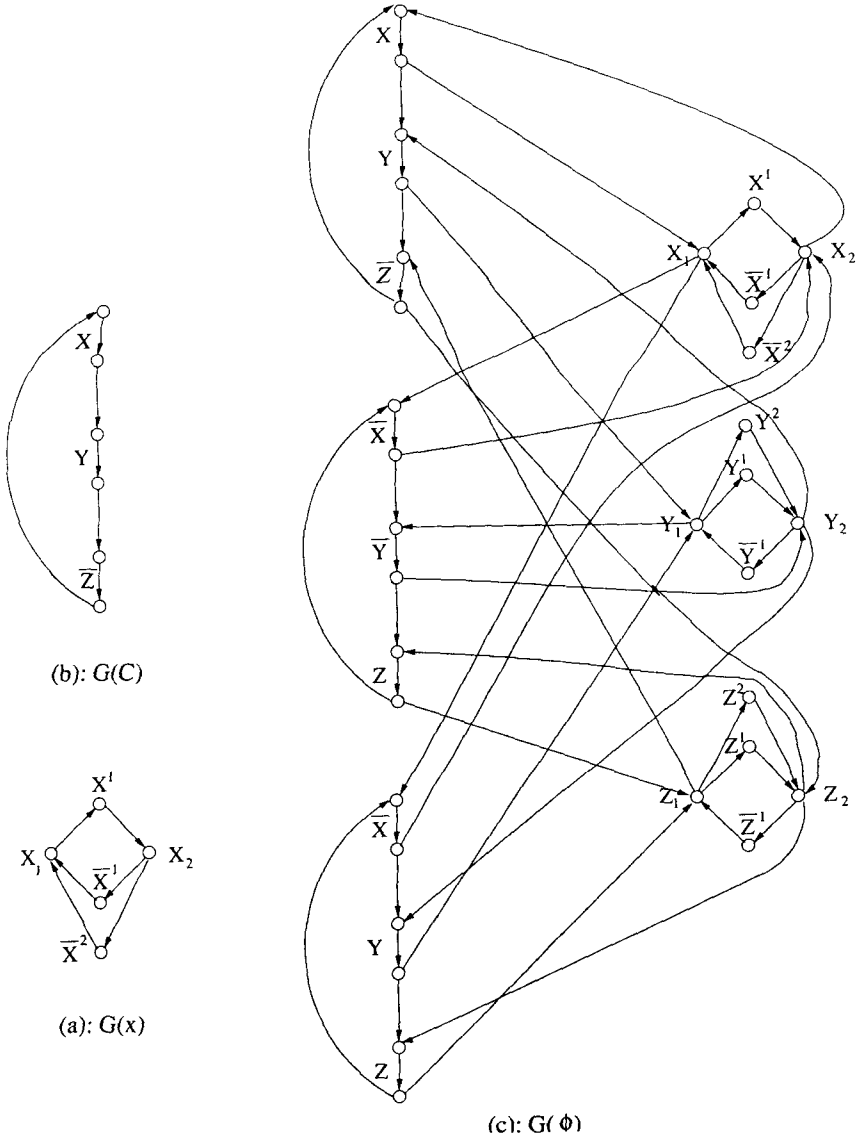


Figure 4.2: (a) Variable gadget for $G(x)$ (b) Clause gadget $G(C)$ for the clause $C = (x \vee y \vee \overline{z})$; (c) The digraph $G(\phi)$ for the formula $\phi = (x \vee y \vee \overline{z}) \wedge (\overline{x} \vee \overline{y} \vee z) \wedge (\overline{x} \vee y \vee z)$;

has weight $\frac{1}{l(x)+l(\overline{x})}$ and each arc in the set $\cup_{i=1}^{l(\overline{x})}\{(x_2,\overline{x}^i),(\overline{x}^i,x_2)\}$ has weight $\frac{n+1}{l(x)+l(\overline{x})}$. All other arcs in $G(\phi)$ has weight $M \geq 18m$. Finally, let $k = 3m$. This completes the construction of $(G(\phi), w, k)$.

Claim 4.1 Let F_o be a minimum cardinality FAS of $G(\phi)$. Then $|F_o| = 3m$

Proof. In [126, Lemma 3], Newman proved that a minimum cardinality FAS F of $G(\phi)$ contains $3m+u$ arcs if and only if the minimum number of unsatisfied clauses in ϕ is u. But, here ϕ being an instance of MIN-Ones+, it is always satisfiable and hence $u = 0$. □

Claim 4.2 Given a solution S of MIN-Ones+ for the instance ϕ, we can construct a FAS F_s of $G(\phi)$ such that $|F_s| = 3m$ and $w(F_s) = n(|S|+1)$.

Proof. From S construct F_s as follows:

- if $x \in S$ then include all the arcs $(x_2, \overline{x}^i)$, for $1 \leq i \leq l(\overline{x})$, in F_s;
- if $x \notin S$ then include all the arcs (x_1, x^i), for $1 \leq i \leq l(x)$, in F_s;
- include all the arcs from the clause gadgets which correspond to a true literal with respect to S.

Now, following the arguments given in the proof of Lemma 3 in [126], it can be proved that F_s is a FAS of $G(\phi)$ and $|F_s| = 3m$. Also

$$\begin{aligned} w(F_s) &= \sum_{x\in S}\left[\frac{n+1}{l(x)+l(\overline{x})}(l(x)+l(\overline{x}))\right] + \sum_{x\notin S}\left[\frac{1}{l(x)+l(\overline{x})}(l(x)+l(\overline{x}))\right] \\ &= \sum_{x\in S}(n+1) + \sum_{x\notin S}1 = n(|S|+1). \square \end{aligned}$$

Claim 4.3 Let F be a minimal FAS of $G(\phi)$ with $|F| = 3m$. Then we can construct a minimal FAS F' of $G(\phi)$ such that F' does not contain any arc of weight M, $|F'| = 3m$ and $w(F') < w(F)$.

Proof. If F contains an arc (i,j) then we can construct another minimal FAS F' of $G(\phi)$ with $w(F') < w(F)$. For this we consider three cases corresponding to the three types of arcs of weight M in $G(\phi)$.

- Let F contains an arc (i,j) of weight M which is an arc in a clause gadget $G(C)$ for some clause $C \in \phi$. From the construction of $G(\phi)$ it follows that $L = F - \{(i,j)\} + \{(h,i)\}$ is a FAS of $G(\phi)$, where (h,i) is the unique arc in $G(C)$ adjacent to (i,j). Now, let F' be any minimal FAS of $G(\phi)$ contained in L and hence $w(F') \leq w(L) < w(F)$.

- Let F contains an arc (i,j) of weight M which is going out from a clause gadget $G(C)$. From the construction of $G(\phi)$, it follows that $L = F - \{(i,j)\} + \{(h,i)\}$ is a FAS of $G(\phi)$, where (h,i) is the unique arc in $G(C)$ adjacent to (i,j). Now, $w(F') \leq w(L) < w(F)$, where $F' \subseteq L$ is any minimal FAS of $G(\phi)$.

- Let F contains an arc (i,j) of weight M which is coming into a clause gadget $G(C)$. Then, from the construction it follows that $L = F - \{(i,j)\} + \{(j,k)\}$ is a FAS, where (j,k) is the unique arc in $G(C)$. Now, $w(F') \leq w(L) < w(F)$, where $F' \subseteq L$ is any minimal FAS of $G(\phi)$.

Since $|L| = |F|$ and $F' \subseteq L$, we have $|F'| \leq |L| = 3m$. But, from Claim 4.1 and as F' is a minimal FAS of $G(\phi)$ with at most $3m$ arcs, it follows that $|F'| = 3m$. □

Claim 4.4 Let F be a minimal FAS of $G(\phi)$ containing no arc of weight M and $|F| = 3m$. Then we can construct a solution S_F satisfying ϕ and $w(F) = n(|S_F| + 1)$.

Proof. Since F is a minimal FAS of $G(\phi)$ and does not contain any arc of weight M, without loss of generality, we assume that F breaks either all the dipaths from x_1 to x_2 and none from x_2 to x_1 or all the dipaths from x_2 to x_1 and none from x_1 to x_2 in the variable gadget $G(x)$, for any $x \in \phi$. This is because, firstly, F must break either all the dipaths from x_1 to x_2 or all dipaths from x_2 to x_1, and secondly, if F breaks all

dipaths from, say, x_1 to x_2, then it cannot break any dipath from x_2 to x_1 as F is a minimal FAS.

Now, we define $S_F = \{x|$ F breaks all the dipaths from x_2 to x_1 in the variable gadget for the variable $x\}$. Note that, for any variable $x \in \phi$, if F breaks all the $l(x)$ dipaths from x_1 to x_2 in the variable gadget $G(x)$, then F must contain all the $l(\overline{x})$ arcs labelled with $\overline{x}$ in the clause gadgets; and, if F breaks all the dipaths from x_2 to x_1i, then F contains all the arcs labelled with x in the clause gadgets. As $\sum_{x\in\phi}[l(x)+l(\overline{x})] = 3m$ and $|F| = 3m$, we can further assume that, if F breaks all the dipaths from x_1 to x_2 (respectively, all the dipaths from x_2 to x_1) in $G(x)$, then F contains the $l(x)$ arcs $(x_1, x^1), \ldots, (x_1, x^{l(x)})$ (respectively, the $l(\overline{x})$ arcs $(x_2, \overline{x}^1), \ldots, (x_2, \overline{x}^{l(\overline{x})})$) from $G(x)$. Then, as in the proof of Claim 4.2, we can show that $w(F) = n(|S_F| + 1)$.

Now, it remains to show that S_F is a solution for the instance ϕ of MIN-Ones+. This can be seen as follows: F contains at least one arc from the clause gadget $G(C)$, for each clause C in ϕ. If that arc is labeled by x, for some variable x, then F must break all the dipaths from x_2 to x_1 in $G(x)$, and hence, $x \in S_F$. If the arc is labeled by $\overline{x}$, then F must break all the dipaths from x_1 to x_2 in $G(x)$, and hence, $x \notin S_F$. Now, for each clause C, the literal l labeling the arc in $G(C)$, that is in F, is true whether the corresponding variable is in S_F or not. Hence, S_F satisfies ϕ. □

From Claims 4.2, 4.3 and 4.4, it follows that if F_o is an optimum solution for the instance $(G(\phi), w, k)$ of MIN-W-k-FAS with $k = 3m$ as constructed above, then the corresponding set S_o, as constructed in the proof of Claim 4.4, is an optimum solution for the instance ϕ of MIN-Ones+. Now, for any FAS F of $G(\phi)$ with $|F| = 3m$, we have $\frac{|S_F|}{|S_o|} \leq 1 + 2\left[\frac{w(F)}{w(F_o)} - 1\right]$, which can be seen by simplification and noting that $|S_F| \geq |S_o| \geq 1$. Hence MIN-Ones+$\leq_{AP}$MIN-W-k-FAS. □

4.3 Approximability of △-LOP

So far we have not made any assumptions about the weight function for linear ordering problems. In this section, we show that, if the weight function satisfies a stronger form of triangle inequality, then a very simple heuristic called, CT-heuristic, produces good approximate solution for linear ordering problems.

△-MIN-LOP (respectively, △-MAX-LOP) denotes MIN-LOP (respectively, MAX-LOP) restricted to instances where the arc weight matrix satisfies the following *triangle inequality* (or △-inequality):

$$\max\{w_{ik}, w_{ki}\} \leq \min\{w_{ij}, w_{ji}\} + \min\{w_{jk}, w_{kj}\} \tag{4.6}$$

for any three nodes $i, j, k \in V$, where $w_{ij} = w(i, j)$. We write △-LOP when referring to either of △-MIN-LOP and △-MAX-LOP. Note that the triangle inequality in (4.6) is stronger than the usual notion of triangle inequality as defined in [4]. Though this may seem to be an artificial restriction on the weight matrix, the point is that with such restrictions on weight matrix we get much better approximate solutions quite easily. However, we do not know whether such approximate solutions can be obtained assuming the standard notion of triangle inequality.

For a set T of arcs in G_n, let $T^c = \{(j, i) | (i, j) \in T\}$. We define a *triangle* $\triangle$ on three distinct nodes $a, b, c \in V$ as a set of three distinct arcs for which any two distinct arcs have exactly one common end point in $\{a, b, c\}$. We first prove a few lemmas before showing △-MIN-LOP (respectively, △-MAX-LOP) admits a $\frac{3}{2}$-approximation (respectively, $\frac{4}{3}$-approximation) algorithm.

Lemma 4.4 *For any triangle* $\triangle$, $w(\triangle) \leq 2w(\triangle^c)$.

Proof. $\triangle$ can be any one of the eight possible triangles over three distinct nodes a, b, c. We shall prove the lemma only for the set $\triangle = \{(a, b), (b, c), (a, c)\}$ and proofs for other sets are similar.

By $\triangle$-inequality, we have

$$w(a,b) \leq w(c,a) + w(c,b)$$
$$w(b,c) \leq w(b,a) + w(c,a)$$
$$w(a,c) \leq w(b,a) + w(c,b).$$

Adding all these inequalities, we have $w(\triangle) \leq 2w(\triangle^c)$. □

Lemma 4.5 *Let (V,T) be any tournament on V. Then $w(T) \leq 2w(T^c)$.*

Proof. Let (V,T) be any arbitrary tournament on V. Let $\triangle$ be any triangle in T. Now consider the sum $\sum w(\triangle)$ over all distinct triangles in T. Since each arc in T appears in exactly $(n-2)$ distinct triangles in T, each arc in T appears exactly $(n-2)$ times in the sum $\sum_{\text{distinct } \triangle \in T} w(\triangle)$. Hence, $\sum_{\text{distinct } \triangle \in T} w(\triangle) = (n-2)w(T)$ and $\sum_{\text{distinct } \triangle^c \in T^c} w(\triangle^c) = (n-2)w(T^c)$. By Lemma 4.4, we have $w(T) \leq 2w(T^c)$. □

We next describe a simple heuristic, called the *complementary tournament heuristic (or CT heuristic)*, for LOP, and then examine its performance.

Algorithm 4.2 CT Heuristic for LOP

Input - An instance (G_n, w) for $\triangle$-LOP

Output - A tournament T_A on V

1. **begin**
2. consider an arbitrary tournament T on V and its complement T^c;
3. choose the one as output T_A which has smaller (respectively, larger) weight for $\triangle$-MIN-LOP (respectively $\triangle$-MAX-LOP);
4. **end.**

The performance of the CT heuristic is given in

Theorem 4.5 *For any instance $x = (G_n, w)$ of $\triangle$-LOP, let $CT(x)$ denote the solution returned by the CT heuristic. Then*

(1) $R(x, CT(x)) \leq \frac{3}{2}$ *for* $\triangle$*-MIN-LOP, and*

(2) $R(x, CT(x)) \leq \frac{4}{3}$ *for* $\triangle$*-MAX-LOP.*

Proof. (1) Let (V, T_A) be the tournament returned by the CT heuristic on the given instance and (V, T_o) be an optimal tournament for the given instance. From our algorithm we have $w(T_A) \leq w(T_A^c)$. So

$$\begin{aligned} 2w(T_A) &\leq w(T_A) + w(T_A^c) \\ &= w(T_o) + w(T_o^c) \\ &\leq w(T_o) + 2w(T_o) \quad \text{(by Lemma 4.5)} \\ &= 3w(T_o). \end{aligned}$$

So, $w(T_A) \leq \frac{3}{2} w(T_o)$, and hence, $\frac{w(T_A)}{w(T_o)} \leq \frac{3}{2}$.

(2) Similar. □

4.4 Approximability of $\triangle_t$-LOP

In this section, we examine how the performance ratio of the solution returned by the CT heuristic for LOP is affected if we relax or strengthen the $\triangle$-inequality for the weight matrix. The weight matrix $w = (w_{ij})$ satisfies the *parametrized triangle inequality* with a positive real parameter t (or $\triangle_t$*-inequality*) if

$$t \cdot \max\{w_{ik}, w_{ki}\} \leq \min\{w_{ij}, w_{ji}\} + \min\{w_{jk}, w_{kj}\}$$

for any 3 nodes i, j, k.

Remark 4.2

1. If the weight function w is not the constant function taking the value 0, then w cannot satisfy $\triangle_t$-inequality for any $t > 2$. So we need to consider $\triangle_t$-inequality only for $t \in (0, 2]$.
2. w satisfies $\triangle_2$-inequality if and only if w is a matrix with all entries equal to a positive constant.

3. If w satisfies $\triangle_t$-inequality, then it also satisfies $\triangle_{t'}$-inequality whenever $0 < t' < t \leq 2$. $\triangle_t$-inequality corresponds to the $\triangle$-inequality for $t = 1$ and is a strengthening (respectively, relaxation) of the $\triangle$-inequality if $t > 1$ (respectively, $t < 1$).

Parametrized triangle inequality is considered for the possible extension of the scope of application of known heuristics for hard combinatorial optimization problems which have better upper bounds on performance ratio when the weight matrix satisfies the standard $\triangle$-inequality [4,15]. In [4], it is observed that the two popular heuristics for MIN-TSP behave differently with respect to $\triangle_t$-inequality, though they have similar properties with respect to the standard $\triangle$-inequality. The performance of T^3 heuristic with respect to $\triangle_t$-inequality for MIN-TSP is consistent with its performance with respect to the standard $\triangle$-inequality, whereas that of Christofides' heuristic is not.

However, we can easily show that the performance of the CT heuristic for LOP with respect to $\triangle_t$-inequality is quite consistent with its performance with respect to the $\triangle$-inequality. By $\triangle_t$-LOP we refer to LOP where the arc weight matrix satisfies the $\triangle_t$-inequality. We have an extension of Theorem 4.5.

Theorem 4.6 *Let $x = (G_n, w)$ be an instance of $\triangle_t$-LOP. If $CT(x)$ is the solution returned by the CT heuristic on input x, then*

(1) for $\triangle_t$-MIN-LOP, $R(x, CT(x)) \leq \frac{2+t}{2t}$, and

(2) for $\triangle_t$-MAX-LOP, $R(x, CT(x)) \leq \frac{4}{2+t}$.

The proof follows from the same sequence of intermediate results and reasoning as for Theorem 4.5, with appropriate modifications of Lemma 4.4 and Lemma 4.5 where the constant 2 appearing in them is replaced by $\frac{2}{t}$. Thus, for instance, the modified form of Lemma 4.5 reads: If $x = (G_n, w)$ is an instance of $\triangle_t$-LOP and T is an acyclic tournament on V, then $w(T) \leq \frac{2}{t} w(T^c)$.

Remark 4.3

1. If $0 < L \leq w_{ij} \leq U$, then w satisfies the $\triangle_t$-inequality with $t = \frac{2L}{U}$, and hence, by Theorem 4.5, $R(x, CT(x)) \leq \frac{U+L}{2L}$ (respectively, $\leq \frac{2U}{U+L}$) for $\triangle_t$-MIN-LOP (respectively, $\triangle_t$-MAX-LOP).
2. Note that the restriction of $\triangle_t$-inequality on the weight matrix becomes stronger and stronger than the $\triangle$-inequality as $t \to 2$ and then $R(x, CT(x)) \to 1$ for $\triangle_t$-LOP, whereas $\triangle_t$-inequality becomes more and more relaxed than the $\triangle$-inequality as $t \to 0$ and then $R(x, CT(x)) \to \infty$ for $\triangle_t$-MIN-LOP and $\triangle_t$-MAX-LOP.

Note that, in view of Remark 4.2, there exists a threshold $t^* \in (0, 2]$ such that $\triangle_t$-LOP is NP-complete for $0 < t < t^*$ and $\triangle_t$-LOP is in the class PO for $t^* \leq t \leq 2$. We next show that $t^* = 2$. That is

Theorem 4.7 *$\triangle_t$-LOP is* NP*-complete for $t \in (0, 2)$.*

Proof. First note that it is enough to prove the statement for any rational $t \in (0, 2)$ as, in view of Remark 4.2 (3), if $\triangle_t$-MIN-LOP is NP-complete, then so is $\triangle_{t'}$-LOP for $0 < t' < t \leq 2$.

Next note that LOP_{pq} is NP-complete for any positive integers p and q. So it is enough to find two positive integers p and q for a given rational $t \in (0, 2)$ such that $p < q$ and $\frac{2p}{q} = t$ so that an instance of LOP_{pq} is an instance of $\triangle_t$-LOP, by Remark 4.3 (1). Now, given any rational $t \in (0, 2)$, let p_1 be any positive integer and let $q_1 = \frac{2p_1}{t}$. As q_1 is rational, $q_1 = \frac{r}{l}$ for some positive integers r and l. Let $p = lp_1$ and $q = r$. As $p_1 < q_1$, we have $p < q$ and $\frac{2p}{q} = \frac{2p_1}{q_1} = t$. □

Corollary 4.1 *If* P≠NP*, then $\triangle_t$-LOP $\in$ PO if and only if $t = 2$.*

4.5 $\triangle_t$-LOP is not in PTAS, if P≠NP

In this section, we shall show that LOP(1, 2) $\notin$ PTAS, unless P=NP, and hence, $\triangle_t$-LOP $\notin$ PTAS, unless P=NP. This would have followed, if we could have prove that

$\triangle_t$-LOP is APX-complete. We have not been able to do that. However, we can prove the result using Theorem 2.1 that gives a characterization of PTAS due to PAZ and Moran [134].

MAX-SUBDAG satisfies the simplicity condition, as defined in Section 2.2.5, because given any digraph $G = (V, A)$ and a positive integer k, in order to decide if the optimum value is at most k, it is sufficient to consider all sets of $k + 1$ arcs and to verify whether at least one of them is acyclic, and this can be done in $O(|A|^{k+1})$ time. We will show that $\triangle_t$-LOP is not in the class PTAS, unless P=NP, by proving that MAX-LOP(1, 2) does not satisfy the bounded condition as defined in the Definition 2.15. Note that, MAX-SUBDAG is APX-complete [132], and hence, is not in PTAS, unless P=NP. But it is simple, and hence, by Theorem 2.1, it does not satisfy the boundedness conditions. We will use this fact to show that MAX-LOP(1, 2) in not in PTAS.

Lemma 4.6 *MAX-LOP(1, 2) does not satisfy the boundedness conditions.*

Proof. It is enough to show that, if MAX-LOP(1, 2) satisfies the boundedness conditions, then so does MAX-SUBDAG. So, suppose MAX-LOP(1, 2) satisfies the boundedness conditions. Then, there exist an algorithm $\mathcal{T}_b$ and a positive integer constant E such that

1. For any instance $f(x)$ of MAX-LOP(1, 2) and for any positive integer c, $T_A = \mathcal{T}_b(f(x), c)$ is a solution of x with $w(T_o) \leq w(T_A) + cE$.
2. The time complexity of $\mathcal{T}_b(f(x), c)$ is a polynomial in $|f(x)|$ whose degree depends only on the value $\frac{m(x, \mathcal{T}_b(f(x),c))}{c}$.

Next, we show that MAX-SUBDAG also satisfies the boundedness conditions by designing an algorithm $\mathcal{T}_b'$ for MAX-SUBDAG which for the same constant E satisfies the boundedness conditions for MAX-SUBDAG.

Let (G_n, w) be any instance of MAX-SUBDAG. Now construct an instance (G_n, w) of MAX-LOP(1, 2) with $w(e) = 2$, if $e \in A$, and $w(e) = 1$ otherwise. For a given

solution (an acyclic tournament) (V,T) of MAX-LOP(1, 2) for (G_n,w), let $\overline{T} = \{e \in T | w(e) = 2\}$. Clearly, $(V,\overline{T})$ is a SUBDAG of G and $w(T) = \frac{1}{2}n(n-1) + |\overline{T}|$. It can be proved that (V,T_o) is an optimal solution of MAX-LOP(1, 2) for (G_n,w) if and only if $(V,\overline{T}_o = \{e \in T_o | w(e) = 2\})$ is an optimal solution of MAX-SUBDAG for G. Also $w(T_o) = \frac{1}{2}n(n-1) + |\overline{T}_o|$.

Now, consider the following algorithm $\mathcal{T}_b'$ for MAX-SUBDAG.

Algorithm 4.3 Algorithm $\mathcal{T}_b'$ for MAX-SUBDAG

Input - $G = (V, A_n)$, an instance of MAX-SUBDAG.

Output - A SUBDAG $(V,\overline{T}_A)$ of G.

1. **begin**
2. Construct an instance (G_n,w) of MAX-LOP(1, 2) from G with $w(e) = 2$ if $e \in A$ and $w(e) = 1$ otherwise;
3. Use the algorithm $\mathcal{T}_b$ and the constant E (which are specified by the boundedness conditions for MAX-LOP(1, 2)) for the instance (G_n,w) of MAX-LOP(1, 2) to get a solution $T_A = \mathcal{T}_b((G_n,w),c)$ of MAX-LOP(1, 2);
4. From T_A construct $\overline{T}_A = \{e \in T_A | w(e) = 2\}$;
5. Return $\overline{T}_A$ as output;
6. **end.**

Since $\mathcal{T}_b$ and E satisfy the boundedness conditions for MAX-LOP(1, 2), we have $w(T_o) \leq w(T_A) + cE$, and hence, $|\overline{T}_o| \leq |\overline{T}_A| + cE$. Hence, the algorithm $\mathcal{T}_b'$ and the constant E satisfy the boundedness conditions for MAX-SUBDAG. □

From the above lemma and the Theorem 2.1, it follows that Max-LOP(1, 2) $\notin$ PTAS, unless P=NP. Next we will show that MAX-LOP(1, 2) $\leq_{AP}$ MIN-LOP(1, 2), from which it will follow that MIN-LOP(1, 2) $\notin$ PTAS, unless P=NP, by Theorem 2.2.

Lemma 4.7 *MAX-LOP(1, 2)* $\leq_{AP}$ *MIN-LOP(1, 2).*

Proof. Let (G_n,w) be an instance of MAX-LOP(1, 2). We also take (G_n,w) as the corresponding instance of MIN-LOP(1, 2). For a solution (V,T) of MIN-LOP(1, 2) for

(G_n, w) we associate (V, T^c) as a solution of MAX-LOP(1, 2) for (G_n, w). (V, T_o^c) is an optimal solution of MIN-LOP(1, 2) if and only if (V, T_o) is an optimal solution of MAX-LOP(1,2) for (G_n, w). It is easy to see that MAX-LOP(1,2)$\leq_{AP}$MIN-LOP(1,2). □

Theorem 4.8 *$\triangle_t$-LOP $\notin$ PTAS for $0 < t < 2$, unless* P=NP.

Proof. As LOP(1, 2) $\notin$ PTAS, it follows that LOP(p, q) $\notin$ PTAS for any positive integer p and q, unless P=NP. Following the reasoning in the proof of Theorem 4.7, we conclude that $\triangle_t$-LOP $\notin$ PTAS for $0 < t < 2$ unless P=NP. □

Remark 4.4 Since MIN-LOP $\equiv_{strict}$ MIN-QAP(S) (Lemma 3.3 and 3.4), the approximability and inapproximability results of $\triangle$-MIN-LOP and $\triangle_t$-MIN-LOP hold for MIN-QAP(S), when the matrix W, in an instance (W, D) of MIN-QAP(S), satisfies the $\triangle$-inequality and $\triangle_t$-inequality.

Chapter 5

On the Complexity and Approximability of OEOP and Some Heuristics

In Section 3.2.2, we described the formulation of the *Optimal Evaluation Ordering Problem*, OEOP, and the context in which it arises. In this chapter, we first prove, in Section 5.1, that OEOP is NP-complete by showing that a restricted version, called OEOP(S), of OEOP is NP-complete. Then, in Section 5.2, we establish *strict*-reductions from OEOP(S) to MIN-LOP and from OEOP to MIN-LOP(set) which is a generalization of MIN-LOP. From these, we make some observations about approximability of OEOP(S) and OEOP. Finally, in Section 5.3, we propose some heuristics for OEOP.

5.1 NP-completeness of OEOP

In this section, we define a restricted version OEOP(S) of OEOP by making some simplifying assumption about the instances of OEOP and show that OEOP(S) is NP-complete. From this, NP-completeness of OEOP follows.

5.1.1 Assumption of Simple Sharing

We will make the following assumption about instances of OEOP.

- *Assumption of simple-sharing*: A shared expression is shared between at most two expression DAGs. Further, if a subexpression is shared by two expressions

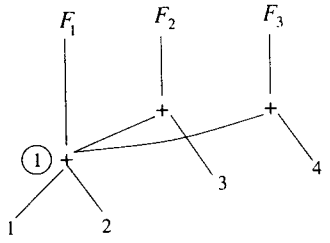


Figure 5.1: The sharing DAG of F_1, F_2 and F_3 does not satisfy the assumption of simple sharing

say F_i and F_j then no subexpression of it is shared by any other expression F_k $(k \neq i, j)$.

In Figure 5.1, the node marked 1 is shared by the three expressions F_1, F_2 and F_3. Let F_2 be evaluated before F_3. Now, whether F_2 will spill node 1 for F_3 depends on whether F_1 has been evaluated already or not (i.e. whether F_1 has spilled the node 1 or not). In other words, the spilling decisions between F_2 and F_3 depends on the context in which they are evaluated. To make such spilling decision between any two function calls, independent of the context in which they are evaluated, we have introduced the assumption of simple sharing. In general, $\text{SPILL}(F_i, F_j) \cap \text{SPILL}(F_j, F_i) \neq \phi$ if one of them is nonempty for any $i \neq j$. But, when OEOP satisfies the assumptions of simple sharing we have the restriction as follows:

$$\text{SPILL}(F_i, F_j) \cap \text{SPILL}(F_k, F_l) = \phi$$

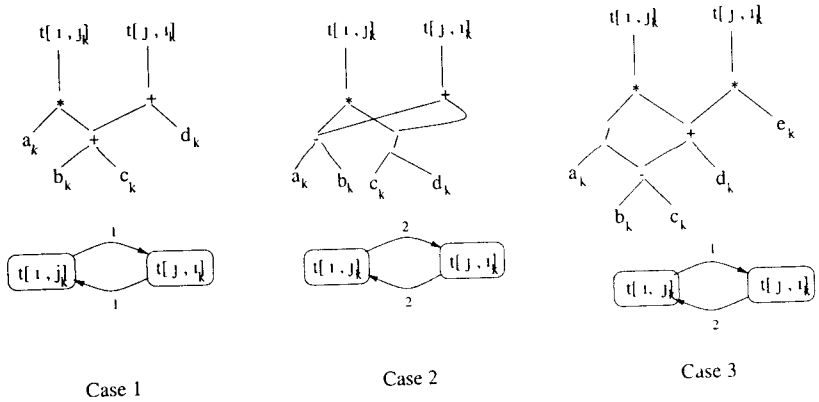
$$\text{SPILL}(F_i, F_j) \cap \text{SPILL}(F_l, F_k) = \phi$$

for all i, j, k, l such that $i \neq j$, $k \neq l$ and $\{i, j\} \neq \{k, l\}$.

The *restricted version* OEOP(S) of OEOP consists of all instances of OEOP that satisfy the assumption of simple sharing.

5.1.2 NP-completeness of OEOP(S)

Clearly, the decision version OEOP(S)_d of OEOP(S) is in the class NP. Therefore, to show that OEOP(S)_d is NP-complete, by Theorem 3.2(ii), it is enough to show that



Case 1 Case 2 Case 3

Figure 5.2: Sharing DAGs of the subexpressions $t[p,1]_k$ and $t[p,2]_k$ and their corresponding arc weighted symmetric digraphs

Theorem 5.1 *MIN-LOP(1, 2)$_d$* $\leq_m^p$ *OEOP(S)$_d$.*

Proof - Let $G_n = (V, A_n)$, together with an arc weight function w and a positive integer bound b, be an instance of MIN-LOP(1, 2)$_d$. From this instance, we will construct an instance of OEOP(S)$_d$ that consists of n arithmetic expressions $F_1, F_2, \ldots, F_n$ satisfying the simple sharing assumption and a positive integer bound $\bar{b}$. We set $\bar{b} = b$. Initially, we set each F_i to be an empty expression. Then, for every pair of nodes i, j, we examine the type of the subdigraph of G_n induced by the nodes i and j, and add two appropriate subexpressions to F_i and F_j. The subdigraph of G_n induced by the nodes i and j will be isomorphic to one of the three different digraphs depending on the arc weights $w(i,j)$ and $w(j,i)$. We associate a pair of subexpressions for each of these induced subdigraphs. The possible three cases along with the corresponding pairs of subexpressions are given bellow.

Case 1 $w(i,j) = w(j,i) = 1$

$t[1,1]_k = a_k * (b_k + c_k)$

$t[2,1]_k = (b_k + c_k) + d_k$

Case 2 $w(i,j) = w(j,i) = 2$

$t[2,1]_k = (a_k - b_k) * (c_k \div d_k)$

$t[2,2]_k = (a_k - b_k) + (c_k \div d_k)$

Case 3 $w(i,j) = 1$ and $w(j,i) = 2$

$$t[3,1]_k = (a_k \div (b_k - c_k)) * ((b_k - c_k) + d_k)$$

$$t[3,2]_k = ((b_k - c_k) + d_k) * e_k$$

It can be easily seen that if we evaluate $t[1,1]_k$ and $t[1,2]_k$ in any order only one spill is required (see Figure 5.2). Suppose $w(i,j) = w(j,i) = 1$. Thus, if we can construct $F_1, F_2, \ldots, F_n$ out of $t[p,q]_k$s such that F_i and F_j share only at the sharing nodes of $t[p,1]_k$ and $t[p,2]_k$i, then the number of spills required for the evaluation of F_j while evaluating F_i will be same as that of $w(i,j)$. The pairs of subexpressions for Case 2 and Case 3 are chosen accordingly. In order to ensure that the assumption of simple sharing holds, we will choose a different suffix k appearing in the subexpressions for different pair of nodes in V.

Formally, we describe this algorithm for constructing n arithmetic expressions from an instance of MIN-LOP(1, 2) as follows:

Algorithm 5.1

Input - An instanc (G_n, w) of MIN-LOP(1, 2).

Output - n arithmetic expressions $F_1, F_2, \ldots, F_n$.

1. **begin**
2. **for** $i = 1$ to n set F_i as an empty expression and set $k = 1$;
3. **for** $i = 1$ to $n - 1$ **do**
4. **for** $j = i + 1$ to n **do**
5. **if** $w(i,j) = w(j,i) = 1$ **then** $F_i = F_i + (t[1,1]_k)$ and $F_j = F_j + (t[1,2]_k)$
6. **if** $w(i,j) = w(j,i) = 2$ **then** $F_i = F_i + (t[2,1]_k)$ and $F_i = F_i + (t[2,2]_k)$
7. **if** $w(i,j) = 1$ and $w(j,i) = 2$ **then** $F_i = F_i + (t[3,1]_k)$ and $F_i = F_i + (t[3,2]_k)$
8. **if** $w(i,j) = 2$ and $w(j,i) = 1$ **then** $F_i = F_i + (t[3,2]_k)$ and $F_i = F_i + (t[3,1]_k)$
9. **od**
10. $k = k + 1$
11. **od.**
12. **end.**

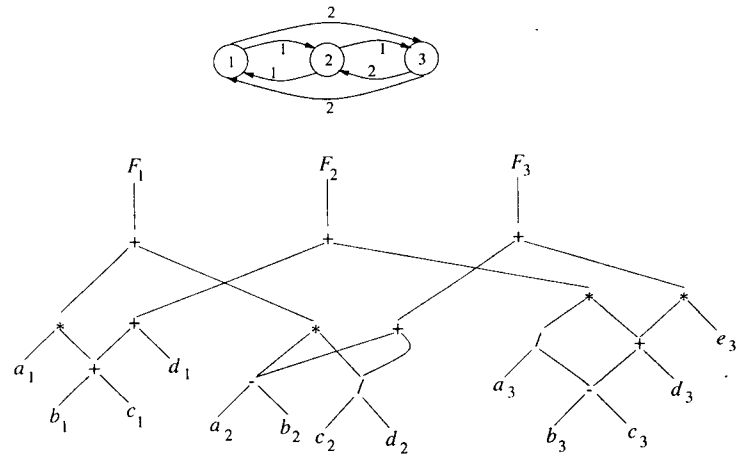


Figure 5.3: A instance of MIN-LOP$_{12}$ and the corresponding sharing DAG

Clearly, the time complexity of the above algorithm is of $O(n^2)$. It can also be seen that each F_i is a summation of $(n-1)$ distinct subexpressions of the form $t[p,q]_k$,and hence, the height of the DAG for each expression F_i is of order $O(n)$.

Each arithmetic expression F_i constructed by this algorithm is of the form

$$F_i = (t[p_{i_1}, q_{i_1}]_{i_1}) + (t[p_{i_2}, q_{i_2}]_{i_2}) + \ldots + (t[p_{i_n}, q_{i_n}]_{i_n})$$

where $p_{i_m} \in \{1,2,3\}$ and $q_{i_m} \in \{1,2\}$ for $1 \leq m \leq n-1$. Two expressions F_i and F_j will share each other if and only if either (i) $t[p,1]_k$ is present in F_i and $t[p,2]_k$ is in F_j or, (ii) $t[p,2]_k$ is in F_i and $t[p,1]_k$ is in F_j, for some p and k. Since we have used different k for different pair of nodes in V no common subexpression of F_i and F_j will be shared by a third expression F_k. This shows that $F_1, F_2, \ldots, F_n$ satisfy the simple sharing assumptions. Let F_i be evaluated before F_j, then $|SPILL(F_i, F_j)|$ is the number of shared subexpressions of F_i to be needed subsequently for the evaluation of F_j. From the construction of $F_1, F_2, \ldots F_n$ if follows that $|SPILL(F_i, F_j)| = w(i,j)$.

Let $\pi = (\pi_1, \pi_2, \ldots \pi_n)$ be a permutation of $\{1, 2, \ldots n\}$. We define $\overline{w}(F_{\pi_1}, F_{\pi_2}, \ldots F_{\pi_n})$ $= |\cup_{i=1}^{n-1} \cup_{j=i+1}^{n} SPILL(F_{\pi_i}, F_{\pi_j})|$. Hence, the $\overline{w}(F_{\pi_1}, F_{\pi_2}, \ldots F_{\pi_n}) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w(\pi_i, \pi_j)$ $= w(T_\pi)$, where $T_\pi = \{(\pi_i, \pi_j) | i < j\}$. Thus, the cost of an order of evaluation of $F_1, F_2, \ldots F_n$ (i.e. a permutation π) is the same as the weight of the acyclic tourna-

ment of G_n associated with the permutation π. This shows that $\overline{w}(F_1, F_2, \ldots F_n) \leq \bar{b}$ if and only if $w(T_\pi) \leq b$. Hence, MIN-LOP(1, 2)$_d$ $\leq_m^p$ OEOP(S)$_d$. □

Example 5.1 For an example, consider the instance of MIN-LOP(1, 2) as in the Figure 5.3. If we apply the algorithm given in the proof of the above theorem for constructing arithmetic expressions satisfying simple sharing assumption, then we will get the sharing DAG for three expressions as shown in Figure 5.3. □

5.2 On the approximability of OEOP(S) and OEOP

In this section, we obtain some relations about approximability of OEOP(S) and MIN-LOP and that of OEOP and a generalization of MIN-LOP, called MIN-LOP(set), via *strict*-reductions. First we show

Theorem 5.2 *OEOP(S)* $\leq_{strict}$ *MIN-LOP.*

Proof Given n arithmetic expressions $F_1, F_2, \ldots F_n$, an instance of OEOP(S), we construct an instance $(G_n = (V, A_n), w)$ of MIN-LOP as follows: The vertex set $V = \{1, 2, \ldots n\}$, where vertex i corresponds to the expression F_i. For any $i, j \in V$, (i, j) is an arc of weight $w(i, j) = |SPILL(F_i, F_j)|$ if the expressions F_i and F_j have a common subexpression, otherwise (i, j) is an arc of weight 0.

Since an order of evaluation of $F_1, F_2, \ldots F_n$ is a permutation $\pi = (\pi_1, \pi_2, \ldots \pi_n)$ of $\{1, 2, \ldots n\}$ and there is a one-to-one correspondence between the permutations of $\{1, 2, \ldots n\}$ and the acyclic tournaments on V of G_n (Proposition 3.1), to each evaluation order $F_{\pi_1}, F_{\pi_2}, \ldots F_{\pi_n}$ of $F_1, F_2, \ldots F_n$ according to π, we associate the corresponding acyclic tournament (V, T_π) on V of G_n. Also, denoting by $\overline{w}(\pi)$ the cost of evaluation of the expressions according to the order π, as defined in Section 3.2.2, and by $w(T_\pi)$ the weight of the corresponding acyclic tournament T_π, we have :

$$\overline{w}(\pi) \quad = \quad |\cup_{i=1}^{n-1} \cup_{j=i+1}^{n} SPILL(F_{\pi_i}, F_{\pi_j})|$$

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} |SPILL(F_{\pi_i}, F_{\pi_j})|$$

(as $F_1, F_2, \ldots F_n$ satisfy simple sharing assumption)

$$= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w(\pi_i, \pi_j) = w(T_\pi).$$

From this, it follows that OEOP(S) $\leq_{strict}$ MIN-LOP. □

Conversely, however, we only have:

Theorem 5.3 *MIN-LOP(1, 2)* $\leq_{strict}$ *OEOP(S).*

Proof Given an instance $(G_n = (V, A_n), w)$ of MIN-LOP(1, 2), we construct an instance $(F_1, F_2, \ldots F_n)$ of OEOP(S), where $F_1, F_2, \ldots F_n$ are as constructed in the proof of Theorem 5.1. There, we have already proved that, for any permutation π of $\{1, 2, \ldots n\}$, $w(T_\pi) = \overline{w}(F_{\pi_1}, F_{\pi_2}, \ldots F_{\pi_n})$. From this the required *strict*-reduction follows. □

We do not know any relationship between OEOP and MIN-LOP with respect to an approximation preserving reduction. However, we can easily establish a *strict*-reduction from OEOP to a generalization of MIN-LOP called MIN-LOP(set). In MIN-LOP(set), given a complete digraph $G_n = (V, A_n)$ and a function $f: A_n \rightarrow 2^S$, where S is a finite nonempty set, we are asked to find an acyclic tournament (V, T) on V of G_n that minimizes the objective function value $w(T) = |\cup_{e \in T} f(e)|$.

We have:

Theorem 5.4 *OEOP* $\leq_{strict}$ *MIN-LOP(set).*

Proof Given an instance $(F_1, F_2, \ldots F_n)$ of OEOP, we construct an instance (G_n, f) of MIN-LOP(set) with $f(i, j) = SPILL(F_i, F_j)$ similar to that in the proof of Theorem 5.2. As there is a one-to-one correspondence between the orders of evaluation of $F_1, F_2, \ldots F_n$ and the acyclic tournaments on V of G_n, and the fact that, for any permutation π of $\{1, 2, \ldots n\}$, $w(F_{\pi_1}, F_{\pi_2}, \ldots F_{\pi_n}) = |\cup_{i=1}^{n-1} \cup_{j=i+1}^{n} SPILL(F_{\pi_i}, F_{\pi_j})| = w(T_\pi)$, the theorem follows. □

On the basis of the above results, we can say a few things about approximability of OEOP(S) and OEOP. In Section 4.5, we have shown that MIN-LOP(1, 2)$\notin$PTAS, unless P=NP. Hence, from Theorem 5.3, it follows that OEOP(S)$\notin$PTAS, unless P=NP. But, by Theorem 5.2, we can say that OEOP(S) has an $O(\log n \log\log n)$-approximation algorithm as MIN-LOP has such an algorithm as shown in Theorem 3.7. Further, in-view of Remark 4.3(1) and Theorem 5.2, if the instances of OEOP(S) are such that any pair of expressions have at least one common subexpression and the number of common subexpressions in any pair is bounded above by a constant, U, then, with such resrictions, OEOP(S) has a $\frac{U+1}{2}$-approximation algorithm.

Intuitively, we feel that approximating OEOP is harder than MIN-LOP but has at least similar approximability properties as that of MIN-LOP(set), for which we do not have any result regarding approximability. On the other hand, MIN-LOP is believed not to be in the class APX as discussed in Section 4.2 and, hence, OEOP is not expected to be in APX, if P$\neq$NP. Because of these negative remarks, we propose some heuristics for OEOP, in the next section.

5.3 Some Heuristics for OEOP

In this section, we propose some heuristics for solving OEOP which are modifications of known heuristics for MAX-LOP. There are various type of heuristics proposed for MAX-LOP [86,21,34]. Of these, the heuristic due to Chanas and Kobilanski [34] takes exponential time in worst case, as it may encounter all solutions of an instance of MAX-LOP. Here, we describe a generic heuristic for OEOP, which is based on the heuristic for MAX-LOP by Hassin and Rubinstein [86].

Before describing the heuristic we need to fix some notations. Let $F_1, F_2, \ldots F_n$ be an instance of OEOP. For a set $S \subset \{1, 2, ...n\}$ and a vertex $i \notin S$, we define SPILL$^{in}(i, S)$ as the set of shared nodes required for the computation of F_i due to the evaluation of expressions F_j, for $j \in S$; and SPILL$^{out}(i, S)$ is the set of shared nodes to be spilled while evaluating the expression F_i for the subsequent evaluation of the

expressions F_j, for $j \in S$. Both SPILL$^{in}(i, S)$ and SPILL$^{out}(i, S)$ can be computed in polynomial time by traversing the DAG for F_i only once. Note that OEOP may not satisfy the simple sharing assumption.

Algorithm 5.2 A Generic Heuristic for OEOP

Input - n arithmetic expressions $F_1, F_2, \ldots F_n$.

Output - A permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ of $\{1, 2, \ldots, n\}$

1. $S = \{1, 2, \ldots, n\}$; $l = 1$; $u = n$.
2. Choose a $i \in S$ and set $S = S - \{i\}$.
3. Compute SPILL$^{in}(i, S)$ and SPILL$^{out}(i, S)$.
4. If $|SPILL^{in}(i, S)| \geq |SPILL^{out}(i, S)|$ then set $\pi(l) = i$ and $l = l + 1$;
   otherwise, set $\pi(u) = i$ and $u = u - 1$.
5. If $u \geq l$ go to Step 2, otherwise, stop.

Depending on the choice of index i in S, various heuristics can be obtained for OEOP. To describe some of the specific choices of $i \in S$ in Step 2 of Algorithm 5.2, we define $w_i(S) = \min\{|SPILL^{in}(i, S)|, |SPILL^{out}(i, S)|\}$ and $W_i(S) = \max\{|SPILL^{in}(i, S)|, |SPILL^{out}(i, S)|\}$. With respect to these notations, we state the choice of $i \in S$ in Step 2 of Algorithm 5.2 as follows:

Algorithm 5.3 In Step 2 of Algorithm 5.2, choose $i \in S$ with smallest $w_i(S - \{i\})$.

Algorithm 5.4 In Step 2 of Algorithm 5.2, choose $i \in S$ with largest $W_i(S - \{i\})$.

For details about other various choices of the index i in Step 2, we refer to the paper of Hassin and Rubinstein [86].

These heuristics work well in practice where much restricted instances need to be solved. To see the kind of instances of OEOP that may have to be solved, note that functional programs written by programmers consists of a set of function definitions and a main program. The amount of sharing between the arguments of functions in user programs is usually not high. However, such programs, for optimization purposes,

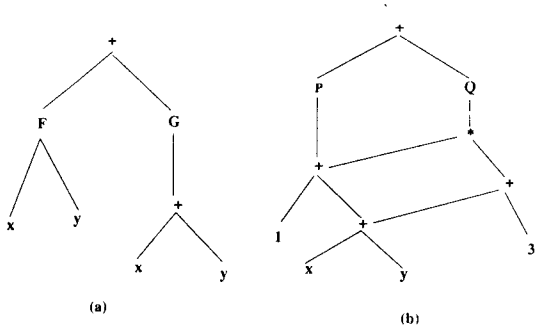


Figure 5.4: Function unfolding introducing sharing

are transformed into equivalent programs. Function unfolding is one such important transformation technique, and this technique introduces sharing [47]. We illustrate this through the following example. Let us consider the expression $F(x,y) + G(x+y)$, where F and G are defined as follows:

$$\begin{aligned} F(a,b) &= P(1+a+b) \\ G(c) &= Q((1+c)*(c+3) \end{aligned}$$

Figure 5.4(a) shows the DAG in relation to the expression $F(x,y) + G(x+y)$, and figure 5.4(b) shows the DAG when functions F and G are unfolded. Observe how sharing gets introduced in the process.

Further, our observation is that the number of function call nodes at the leaf-level of an expression DAG (after unfolding) does not go beyond 7, and further, the number of shared nodes between the argument DAGs of two functions also hardly goes beyond 8. So, the graph that we will construct will not have nodes more than 7 and further the size of a set associated with an arc will be within 0 to 8. With such restrictions the above heuristics give satisfactory results.

Chapter 6

Hardness of Approximating Some NPO Problems *Related* to MIN-LOP

In this chapter, we deal with hardness of approximating several minimum-maximal (or simply, minimaximal) and maximum-minimal (or simply, maximinimal) NP-complete optimization problems on graphs as well as related maximum/minimum problems. In general, for any given instance x of such a problem, it is required to find a minimum (respectively, maximum) weight (or, cardinality) maximal (respectively, minimal) feasible solution with respect to a partial order on the set of feasible solutions of x. The terminology of minimaximal and maximinimal is apparently first used by Peters et.al. [136], though the concept has received attention of many others, specially in connection with many graph problems. We may cite for example, minimum chromatic number and its maximum version, the achromatic number [84,85,58,36]; maximum independent set and minimaximal independent set (minimum independent dominating set) [87, 99,95,81]; minimum vertex cover and maximinimal vertex cover [132,118]; minimum dominating set and maximinimal dominating set [116,37]; minimum vertex (edge) connectivity and maximinimal vertex (edge) connectivity [136,85] and a recent systematic study of minimaximal and maximinimal optimization problems by Manlove [118].

We are led to investigation of several such problems while considering a generalization of MIN-LOP, viz. MIN-W-MAX-SUBDAG and its unweighted version MIN-MAX-SUBDAG as discussed in Section 3.2.3. As MIN-LOP is APX-hard, MIN-W-MAX-SUBDAG is so. For MIN-W-MAX-SUBDAG, unlike MIN-LOP, we cannot talk

about arc weights satisfying triangle (or parameterized triangle) inequality, and so there is no immediate answer to the question whether it is in APX with suitable restrictions on arc weights. But it appears to be unlikely as we show that MIN-MAX-SUBDAG is APX-hard even though MIN-LOP with constant arc weight is solvable in polynomial time. The complementary problem corresponding to MIN-MAX-SUBDAG is the MAX-MIN-FAS problem and its vertex version is MAX-MIN-FVS. We also consider the problem MAX-MIN-VC that is related to MAX-MIN-FVS and the problem MIN-MAX-IS. MIN-MAX-IS and MAX-MIN-VC have the same complexity over every graph class [118, Theorem 4.2.11] though they may not have similar approximation properties and are NP-complete even for bipartite graphs [95,81] and dually chordal graphs [29] though polynomial-time algorithms are known for many special classes of graphs such as chordal graphs [61], interval and circular-arc graphs [35], permutation graphs [62, 30] and many other graph classes (see [118]).

We establish several results about hardness of approximating such graph problems on general graphs as well as degree bounded graphs and digraphs, using appropriate reductions.

We present the results of this chapter organized as follows. In Section 6.1, we first prove APX-hardness of MIN-MAX-SUBDAG and MAX-MIN-FAS for arbitrary digraph by establishing L-reductions from MAX-SUBDAG. Then, using the results of Håstad concerning hardness of approximating MAX-IS (Theorem 2.6 and Theorem 2.7) and the technique of Theorem 2.8 regarding S-reduction, we prove similar results about MAX-MIN-VC for arbitrary graphs and about MAX-MIN-FVS for arbitrary graphs and digraphs. In Section 6.2, we show that, MIN-FVS is APX-complete for 6-regular graphs and MAX-MIN-FVS is APX-hard for all graphs of maximum degree 9. We also prove that MAX-MIN-VC is k-approximable for all graphs without any isolated vertex and having maximum degree k, $k \geq 1$, and is APX-complete for all graphs of maximum degree 5. In Section 6.3, we prove APX-hardness of MIN-FAS and MAX-SUBDAG for k-total-regular digraphs, for all $k \geq 4$. Then we show that MIN-MAX-SUBDAG is

APX-hard for digraphs of maximum total degree 12 and MAX-MIN-FVS is APX-hard for all digraphs of maximum total degree 6.

6.1 Hardness Results for Arbitrary Graphs/Digraphs

In this section, we prove hardness results for several minimaximal and maximinimal problems for general graphs and digraphs. First we consider MIN-MAX-SUBDAG problem. As already noted, MIN-W-MAX-SUBDAG is APX-hard as it is a generalization of MIN-LOP which is APX-hard [121]. However, even though MIN-LOP can be solved in polynomial time when the arc weights are constant, the MIN-MAX-SUBDAG, the unweighted version of MIN-W-MAX-SUBDAG, is APX-hard.

Theorem 6.1 *MIN-MAX-SUBDAG is APX-hard.*

Proof It is enough to show that MAX-SUBDAG$\leq_L$MIN-MAX-SUBDAG as MAX-SUBDAG is APX-complete [132].

For each instance $x = (G = (V, A))$ of MAX-SUBDAG, we construct in polynomial time an instance $f(x) = (G' = (V', A'))$ of MIN-MAX-SUBDAG and with each feasible solution (V', S') of $f(x)$, we associate a feasible solution $g(S') = S = S' \cap A$ of x such that f and g satisfy the conditions of L-reduction with $\alpha = 5$ and $\beta = 1$.

Let $K = \{(v_i, v_j) | (v_i, v_j) \in A \text{ and } (v_j, v_i) \notin A\}$. For each arc $(v_i, v_j) \in K$, we introduce a new vertex v_{ij} for the construction of G'. Construct $G' = (V', A')$ as follows: $V' = V \cup \{v_{ij} | (v_i, v_j) \in K\}$ and $A' = A \cup \{(v_j, v_i), (v_j, v_{ij}), (v_{ij}, v_i) | (v_i, v_j) \in K\}$. For an example, see Figure 6.1. Let $k = |K|$ and p be the number of pairs of vertices $v_i, v_j \in V$ such that both $(v_i, v_j), (v_j, v_i) \in A$. Hence, $p = \frac{|A-K|}{2}$.

Next we establish a few claims.

Claim 6.1 Let (V', S') be a maximal SUBDAG of G' and $S = S' \cap A$. Then (V, S) is a SUBDAG of G and $|S'| = 3k + 2p - |S|$.

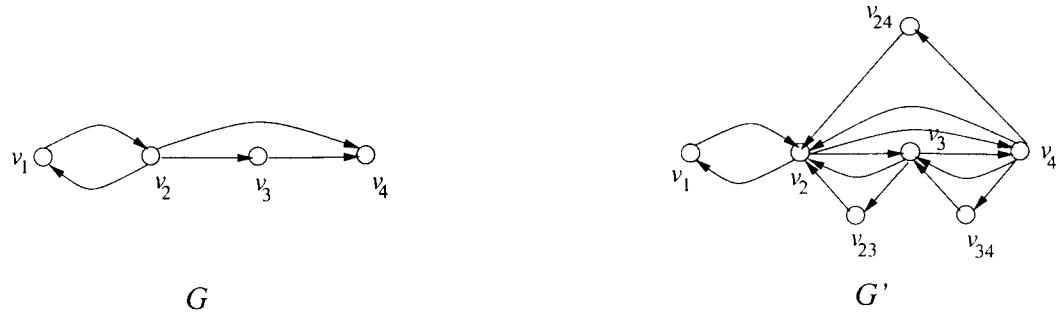


Figure 6.1: A digraph G and the corresponding digraph G'

Proof Since (V, S) is just a subdigraph of (V', S'), it is acyclic and so (V, S) is a SUBDAG of G.

Now note that an arc in $S' - A$ must come from the set $\{(v_j, v_i), (v_j, v_{ij}), (v_{ij}, v_i)\}$ for any arc $(v_i, v_j) \in K$. As (V', S') is a maximal SUBDAG of G', if $(v_i, v_j) \in K \cap S'$, then S' must contain exactly one of (v_j, v_{ij}) and (v_{ij}, v_i), and if $(v_i, v_j) \in K - S'$, then S' must contain all the three arcs $(v_i, v_j), (v_j, v_{ij})$ and (v_{ij}, v_i). So, $|S'| = |S| + k + 2|K - S|$. We now show that $|K - S| = k + p - |S|$. Note that $|K - S| = |K| - |K \cap S|$ and $|K \cap S| = |S| - |S - K|$. Also $S - K = (A \cap S') - K = (A - K) \cap S'$. Now S' contains exactly one of the two (v_i, v_j) and (v_j, v_i) for every pair of arcs (v_i, v_j) and (v_j, v_i) in $A - K$ as (V', S') is a maximal SUBDAG of G'. It follows that $|(A-K) \cap S'| = \frac{1}{2}|A - K| = p$ and $|S - K| = p$ so $|K \cap S| = |S| - p$. Thus $|S'| = |S| + k + 2(k + p - |S|) = 3k + 2p - |S|$. □

Claim 6.2 If (V', S'_o) is a minimum maximal SUBDAG of G', then $(V, S_o = S'_o \cap A)$ is a maximum SUBDAG of G. Also $|A| \leq 2|S_o|$.

Proof If (V, S_o) is not a maximum SUBDAG of G, then let $(V, \overline{S})$ be a maximum SUBDAG of G. Let $\overline{S}' = \overline{S} \cup \{(v_{ij}, v_i) | (v_i, v_j) \in K \cap \overline{S}\} \cup \{(v_j, v_i), (v_j, v_{ij}), (v_{ij}, v_i) | (v_i, v_j) \in K - \overline{S}\}$. Now clearly $\overline{S} = \overline{S}' \cap A$. Also $(V, \overline{S}')$ is a maximal SUBDAG of G'. This is because

(a) as $(V, \overline{S})$ is a maximum SUBDAG of G and $\overline{S}'$ contains $\overline{S}$, no arc from $A - \overline{S}$ can be added to $\overline{S}'$ without creating a cycle, and

(b) no arc from $A' - (A \cup \overline{S}')$ can be added to $\overline{S}'$ without creating a cycle. To see this, note that $A' - (A \cup \overline{S}') = \{(v_j, v_i), (v_j, v_{ij}) | (v_i, v_j) \in K \cap \overline{S}\}$. As both (v_i, v_j) and $(v_j, v_i) \in \overline{S}'$, we cannot add any arc from $A' - (A \cup \overline{S}')$ to $\overline{S}'$ without creating a cycle.

Also by Claim 6.1, $|\overline{S}'| = 3k + 2p - |\overline{S}|$. Since $|\overline{S}| > |S_o|$, $3k + 2p - |S_o| = |S'_o| > 3k + 2p - |\overline{S}|$, which contradicts the assumption that (V', S'_o) is a minimum cardinality maximal SUBDAG of G'.

To show $|A| \leq 2|S_o|$, it is enough to observe that as (V, S_o) is a maximum SUBDAG of G, $(V, A - S_o)$ is a SUBDAG of G. Hence $|S_o| \geq |A - S_o| = |A| - |S_o|$, i.e., $2|S_o| \geq |A|$. □

Now returning to the proof of Theorem 6.1, note that $|S'_o| = 3k + 2p - |S_o| \leq 3(k + p) - |S_o| \leq 6|S_o| - |S_o| = 5|S_o|$. Also for any maximal SUBDAG (V', S') of G', $|S_o| - |S| = |S'| - |S'_o|$. □

We also have

Theorem 6.2 *MAX-MIN-FAS is APX-hard.*

Proof As MAX-SUBDAG is APX-complete, it is enough to show that MAX-SUBDAG $\leq_L$ MAX-MIN-FAS. Towards this, given an instance $G = (V, A)$ of MAX-SUBDAG, we construct an instance $G' = (V', A')$ of MAX-MIN-FAS as defined in the proof of Theorem 6.1. We will use the notations used in the proof of Theorem 6.1 in proving this theorem.

First we prove two claims.

Claim 6.3 Let $S' \subseteq A'$ be a minimal FAS of G' and $S = A \cap (A' - S')$. Then (V, S) is a SUBDAG of G, and $|S'| = k + |S|$.

Proof Since S' is a minimal FAS of G', $(V', A' - S')$ is a maximal SUBDAG of G'. Then by Claim 6.1, (V, S) is a SUBDAG of G and

$$|A' - S'| = 3k + 2p - |S|$$

$$\begin{aligned} i.e. \quad |A'| - |S'| &= 3k + 2p - |S| \\ i.e. \quad |S'| &= |A'| - 3k - 2p + |S| \\ &= k + |S| \quad (\text{as } |A'| = 4k + 2p). \square \end{aligned}$$

Claim 6.4 If S'_o is a maximum minimal FAS of G', then $(V', A' - S'_o)$ is a minimum maximal SUBDAG of G' and $(V, S_o = A \cap [A' - S'_o])$ is a maximum SUBDAG of G. Also $|A| \leq 2|S_o|$.

Proof Proof follows directly from Claim 6.2. □

Now returning to the proof of theorem, by Claims 6.3 and 6.4, we have : $|S'_o| = k + |S_o| \leq k + p + |S_o| \leq 2|S_o| + |S_o| = 3|S_o|$. Also, for any maximal SUBDAG (V', S') of G', $|S'_o| - |S'| = k + |S_o| - k - |S| = |S_o| - |S|$. This establishes the required L-reduction with $\alpha = 3$ and $\beta = 1$. □

Next we prove results about hardness of approximating MAX-MIN-VC and MAX-MIN-FVS, using S-reducibility arguments and the results of Håstad [87] concerning MAX-IS stated in Section 2.4, Theorem 2.6 and Theorem 2.7.

Theorem 6.3 *Unless NP = ZPP, for any $\epsilon > 0$ there exists no polynomial-time algorithm to approximate MAX-MIN-VC within a factor of $\frac{1}{2\sqrt{2}} n^{\frac{1}{2}-\epsilon}$, where n is the number of vertices in an instance.*

Proof Given an instance $G = (V, E)$ of MAX-IS, we construct an instance $G' = (V', E')$ of MAX-MIN-VC, where $V' = V \cup [\cup_{v \in V} \{v^1, v^2, \ldots v^{n+1}\}]$ and $E' = E \cup \{\{v, v^1\}, \{v, v^2\}, \ldots \{v, v^{n+1}\} | v \in V\}$. In other words, G' is obtained from G by introducing for each vertex $v \in V$, $n + 1$ additional vertices $v^1, v^2, \ldots, v^{n+1}$ and $(n + 1)$ additional edges $\{v, v^1\}, \{v, v^2\}, \ldots \{v, v^{n+1}\}$ to the graph G (for example see Figure 6.2).

We first establish a few claims.

Claim 6.5 A vertex cover $S' \subseteq V'$ of G' is a minimal VC if and only if

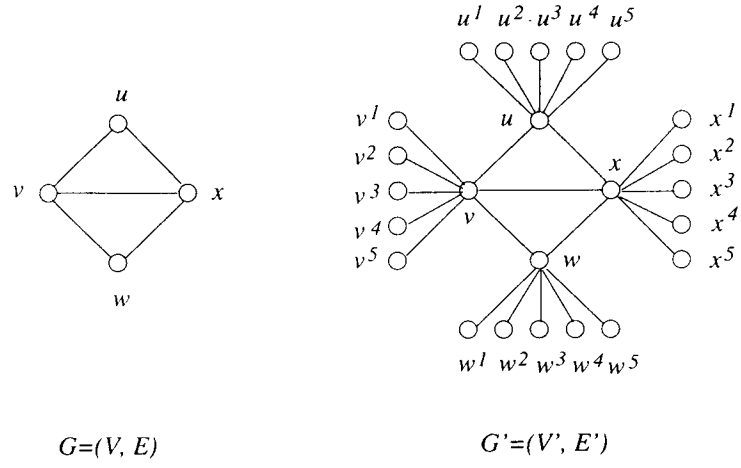


Figure 6.2: An instance G of MAX-IS and the corresponding instance G' of MAX-MIN-VC

(a) for $v \in S' \cap V$, $v^i \notin S'$, for any $1 \leq i \leq n+1$, and

(b) for $v \in V - S'$, $\{v^1, v^2, \ldots v^{n+1}\} \subseteq S'$.

Proof Let S' be a minimal VC of G'. If $v \in S' \cap V$, then for $1 \leq i \leq n+1$, no v^i is in S' as S' is a minimal VC and v covers the edge $\{v, v^i\}$. If $v \in V - S'$, then to cover the edges $\{v, v^i\}$, $1 \leq i \leq n+1$, S' must include v^i.

Conversely, suppose S' satisfies (a) and (b) of Claim 6.5. We will show that we cannot drop any vertex from S' to get a proper subset $S'' \subset S'$ which is also a VC of G'. If $v \in S' \cap V$ then, as for $1 \leq i \leq n+1$, none of v^i is in S' and none of the edges $\{v, v^i\}$ can be covered by a vertex in S' other than v, hence v cannot be dropped from S'. If $v \in V - S'$ then any vertex $v^i \in S'$, $1 \leq i \leq n+1$, cannot be dropped from S', as no vertex in $S' - \{v^i\}$ will cover the edge $\{v, v^i\}$. □

From the Claim 6.5, it follows that for any minimal VC $S' \subseteq V'$ of G', there exists a set $S \subseteq V$ such that $S' = (V - S) \cup [\cup_{v \in S}\{v^1, v^2, \ldots, v^{n+1}\}]$.

Claim 6.6 Let S be a maximal IS of G. Then $S' = (V - S) \cup [\cup_{v \in S}\{v^1, v^2, \ldots, v^{n+1}\}]$ is a minimal VC of G'.

Claim 6.7 Let S' be a minimal VC in G'. If $V - S'$ is not a maximal IS of G, then there exists a minimal VC S'' of G' such that $V - S''$ is a maximal IS of G and moreover,

(a) $|S''| > |S'|$

(b) $|V - S''| > |V - S'|$ and

(c) $|S''| = n(|V - S''| + 1)$.

Proof Note that, for any VC S' of G', $V \cap S'$ is a VC of G. Hence $V - S' = V - (V \cap S')$ is an independent set of G. Let S' be a minimal VC of G' for which $V - S'$ is not a maximal independent set of G. Then we can always extend $(V - S')$ to a unique maximal IS S of G (in polynomial-time) by introducing vertices of G one by one in the order $v_1, v_2, \ldots, v_n$ while mentaining the independence property. Hence $S \supset (V - S')$. By Claim 6.6, $S'' = (V - S) \cup [\cup_{v \in S}\{v^1, v^2, \ldots, v^{n+1}\}]$ is a minimal VC of G' and $|S''| = n(|S| + 1)$. Now we show that $S = V - S''$. For this first note that $S \subseteq V$ as S is a maximal independent set of G. Next, let $u \in S$, then from the definition of S'' it follows that $u \notin S''$, so $u \in V - S''$. Hence $S \subseteq V - S''$. Also, if $u \in V - S''$, then $u \notin S''$, i.e. $u \notin V - S$, so $u \in S$. Hence $S \supseteq V - S''$. Thus $S = V - S''$. From this it follows that $V - S''$ is a maximal independent set of G.

From Claim 6.5, we have $|S'| = |V \cap S'| + (n+1)|V - (V \cap S')| = n(n+1) - n|V \cap S'| = n + n|V - S'| = n(|V - S'| + 1)$. Since $|S| > |V - S''|$, it follows that $|S''| > |S'|$. Also (b) and (c) follow from the fact that $S = V - S''$. □

Claim 6.8 $S \subseteq V$ is a maximum IS of G if and only if $S' = (V - S) \cup [\cup_{v \in S}\{v^1, v^2, \ldots, v^{n+1}\}]$ is a maximum cardinality minimal VC of G'.

Proof Let S be a maximum IS of G. By Claim 6.6, S' is a minimal VC of G'. If S' is not a maximum cardinality minimal VC of G', then using Claim 6.7 there exists a minimal VC S'' of G' such that $|S''| > |S'|$, $S = V - S''$ is a maximal IS of G and $|S''| = n(|S| + 1)$. As $|S'| < |S''|$, $|S'| = n(|S| + 1)$ and $|S''| = n(|\overline{S}| + 1)$, it follows

that $|S| < |\overline{S}|$, which is a contradiction. Hence S' is a maximum cardinality minimal VC in G.

Let S' be a maximum cardinality minimal VC of G'. Then by Claim 6.7, $S = V - S'$ is a maximal IS of G and $|S'| = n(|S| + 1)$. We claim that S is a maximum IS in G. Suppose there exists a maximal IS $S^* \subseteq V$ of G with $|S^*| > |S|$. By Claim 6.6, $\hat{S} = (V - S^*) \cup [\cup_{v \in S^*} \{v^1, v^2, \ldots, v^{n+1}\}]$ is a minimal VC in G' and $|\hat{S}| = n(|S^*| + 1)$. Since $|S^*| > |S|$, it follows that $|\hat{S}| > |S'|$, which is a contradiction. Hence, S is a maximum IS of G. □

We now complete the proof Theorem 6.3. Let $\alpha(G)$ denote the independence number and $\beta(G)$ denote the size of a maximum cardinality minimal VC in G. Hence, from Claim 6.8, we have $\beta(G') = n(\alpha(G) + 1)$. Now let S' be any minimal VC of G'. If $V - S'$ is a maximal IS of G then to S' we associate $S = V - S'$ as the feasible solution of MAX-IS for G. If $V - S'$ is not a maximal IS of G then let S'' be the minimal VC of G' corresponding to S' as in Claim 6.7, so that $S = V - S''$ is a maximal IS of G and $|S'| < |S''| = n(|S| + 1)$. To this minimal VC S' of G' we associate S as the feasible solution of MAX-IS for G. Hence for any minimal VC S' of G' we have

$$\begin{aligned}
\frac{\alpha(G)}{|S|} &= \frac{n\alpha(G)}{n|S|} = \frac{\beta(G') - n}{|S''| - n} = \frac{\beta(G')}{|S''| - n} - \frac{n}{|S''| - n} \\
&= \frac{\beta(G')}{|S''|} \cdot \frac{|S''|}{|S''| - n} - \frac{1}{|S|} = \frac{\beta(G')}{|S''|} \cdot \frac{n(|S| + n)}{n|S|} - \frac{1}{|S|} \\
&= \frac{\beta(G')}{|S''|} + \frac{1}{|S|}\left(\frac{\beta(G')}{|S''|} - 1\right) \\
&\leq \frac{\beta(G')}{|S''|} + \frac{\beta(G')}{|S''|} - 1 \quad \left(\text{since } \frac{\beta(G')}{|S''|} \geq 1 \text{ and } |S| \geq 1\right) \\
&< 2\frac{\beta(G')}{|S''|} \leq 2\frac{\beta(G')}{|S'|}
\end{aligned}$$

Let N be the number of vertices in G'. Since $N = n^2 + 2n$ and $N \leq 2n^2$, for $n \geq 2$. Now, for any $\epsilon > 0$, $n^{1-\epsilon} \geq \frac{N^{\frac{1}{2}(1-\epsilon)}}{2^{\frac{1}{2}(1-\epsilon)}} \geq \frac{N^{\frac{1}{2}(1-\epsilon)}}{\sqrt{2}} \cdot 2^{\frac{\epsilon}{2}} \geq \frac{1}{\sqrt{2}} N^{\frac{1}{2}(1-\epsilon)}$. From Theorem 2.6, it now follows that, unless NP=ZPP, for any $\epsilon > 0$, there exists no polynomial-time algorithm to approximate $\beta(G')$ within a factor of $\frac{1}{2\sqrt{2}} N^{\frac{1}{2}(1-\epsilon)}$, where N is the number of vertices in G'. Hence the theorem follows. □

We also have:

Theorem 6.4 *Unless P=NP, for any $\epsilon > 0$, there is no polynomial-time algorithm to approximate MAX-MIN-VC within a factor of $\frac{1}{2\sqrt[4]{2}} n^{\frac{1}{4}-\epsilon}$, where n is the number of vertices in an instance.*

Proof Same as that of Theorem 6.3, and the fact that $\frac{1}{\sqrt[4]{2}} N^{\frac{1}{4}(1-2\epsilon)} \leq n^{\frac{1}{2}-\epsilon}$. □

Regarding MIN-MAX-FVS, we have similar results. First we shall consider this problem for digraphs.

Theorem 6.5 *Unless NP=ZPP, for any $\epsilon > 0$, there exists no polynomial-time algorithm to approximate MAX-MIN-FVS within a factor of $\frac{1}{4} n^{\frac{1}{2}-\epsilon}$, where n is the number of vertices in an instance.*

Proof We prove this by an S-reduction from *MAX-MIN-VC to MAX-MIN-FVS* as follows.

Let $G = (V, E)$ be a graph (an instance of MAX-MIN-VC). Construct an instance $G' = (V', A')$ of MAX-MIN-FVS from G with $V' = \cup_{v_i \in V} \{v_i^1, v_i^2\}$ and $A' = [\cup_{v_i \in V} \{(v_i^1, v_i^2)\}] \cup [\cup_{\{v_i, v_j\} \in E} \{(v_i^2, v_j^1), (v_j^2, v_i^1)\}]$. In other words, for each $v_i \in V$, G' has 2 vertices v_i^1, v_i^2 and an arcs (v_i^1, v_i^2). Also for each $\{v_i, v_j\} \in E$ G' has (v_i^2, v_j^1) and (v_j^2, v_i^1). Hence, G' has $2n$ vertices and $n + 2m$ arcs. See Figure 6.3 for an example.

First we establish two claims.

Claim 6.9 For any $C \subseteq V$,

(1) C is a VC of G if and only if $F = \{v_i^1 | v_i \in C\}$ is a FVS of G'.

(2) C is a minimal VC of G if and only if F is a minimal FVS of G'.

Proof (1) Let C be a VC of G. We will show that F is a FVS of G'. Every dicycle in G' contains the sequence of four vertices $(v_i^1, v_i^2, v_j^1, v_j^2)$ for some $\{v_i, v_j\} \in E$. Since C is a VC, it follows that at least one of v_i^1 and v_j^1 is in F. Hence, F is a FVS of G'.

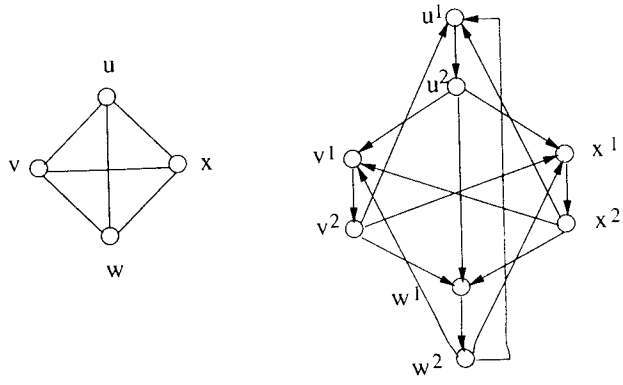


Figure 6.3: An instance G of MAX-MIN-VC and the corresponding instance G' of MAX-MIN-FVS

Conversely, let F be a FVS of G. Suppose C is not a VC of G. Then there exists an edge $\{v_i, v_j\} \in E$ none of v_i and v_j is in C. Hence, the cycle $(v_i^1, v_i^2, v_j^1, v_j^2, v_i^1)$ is in $G' - F$, and hence, F is not a FVS of G' which is a contradiction.

(2) Let C be a minimal VC of G. Thus, for any $v_i \in C$, there exists a $v_j \in V - C$ such that $\{v_i, v_j\} \in E$. Hence for any $v_i^1 \in F$, there is some $v_j \in V - C$ such that the cycle $(v_i^1, v_i^2, v_j^1, v_j^2)$ contains one and only one vertex namely v_i^1 in F. Hence F is a minimal FVS of G'.

Conversely, let F be a minimal FVS of G' and suppose C is not a minimal VC of G. Let $C' \subset C$ be a minimal VC of G. By (1), F' is a FVS of G' and $F' \subset F$, which is a contradiction. So C must be a minimal VC of G. □

Claim 6.10 Let F be any minimal FVS of G'. Then

(1) for any $v_i \in V$, $F \cap \{v_i^1, v_i^2\}$ is either empty or singleton.

(2) for any $v_i \in V$ such that $F \cap \{v_i^1, v_i^2\} \neq \phi$, $F' = F - \{v_i^1, v_i^2\} + v_i^1$ is also a minimal FVS of G'.

(3) There is a minimal FVS F' of G' such that $|F'| = |F|$ and $F' = \{v_i^1 | v_i \in C\}$ for some minimal VC C of G such that $|C| = |F'|$.

Proof (1) This follows because F is a minimal FVS and every dicycle of G' contains either both v_i^1 and v_i^2 or none at all.
(2) If F contains v_i^1 then $F = F'$. If F contains v_i^2, then $F' = F - v_i^2 + v_i^1$ and, as every cycle in G' containing V_i^2 must contain v_i^1 and vice versa, F' must be a minimal FVS of G'.
(3) By repeated application of (2), we get a minimal FVS F' of G' such that $|F'| = |F|$ and any vertex $v \in F'$ is v_i^1 for some $v_i \in V$. Let $C = \{v_i | v_i^1 \in F'\}$. Then $F' = \{v_i^1 | v_i \in C\}$. Now by Claim 6.9, as F' is a minimal FVS of G', C is a minimal VC of G. □

Now coming back to the proof of Theorem 6.5, let F_o be a maximum minimal FVS of G' and F be any minimal FVS of G'. By Claim 6.10, without loss of generality we can assume that every vertex in F_o (respectively, in F) is v_i^1 for some $v_i \in V$. Also by Claim 6.10, $C_o = \{v_i | v_i^1 \in F_o\}$, (respectively, $C = \{v_i | v_i^1 \in F\}$) is a maximum minimal VC (respectively, minimal VC) of G, and $|C_o| = |F_o|$ (respectively, $|C| = |F|$). Hence $\frac{|C_o|}{|C|} = \frac{|F_o|}{|F|}$.

Let $N = |V'|$. Then $N = 2n$. Now for any $\epsilon > 0$, $\frac{1}{2\sqrt{2}} n^{\frac{1}{2}-\epsilon} = \frac{1}{2\sqrt{2}} \frac{(2n)^{\frac{1}{2}-\epsilon}}{2^{\frac{1}{2}-\epsilon}} = \frac{1}{4} N^{\frac{1}{2}-\epsilon} \cdot 2^{\epsilon} \geq \frac{1}{4} N^{\frac{1}{2}-\epsilon}$. Hence by, Theorem 6.3, the result follows. □

We also have:

Theorem 6.6 *Unless P=NP, for any $\epsilon > 0$, there exists no polynomial-time algorithm to approximate MAX-MIN-FVS within a factor of $\frac{1}{2\sqrt{2}} n^{\frac{1}{4}-\epsilon}$, where n is the number of vertices in an instance.*

Towards MAX-MIN-FVS for general graphs we have

Theorem 6.7 *Unless NP=ZPP, for any $\epsilon > 0$, there exists no polynomial-time algorithm to approximate MAX-MIN-FVS, for general graphs, within a factor of $\frac{1}{4} n^{\frac{1}{2}-\epsilon}$, where n is the number of vertices in an instance.*

Proof We shall prove this by establishing an S-reduction from MAX-MIN-VC. For any given instance of MAX-MIN-VC, i.e. a graph $G = (V, E)$, construct an instance

$G' = (V', E')$ from G by introducing a new vertex t and connecting each vertex of G to t by an edge. Hence in G', $V' = V \cup \{t\}$ and $E' = E \cup \{(v_i, t) | v_i \in V\}$. Clearly, $|V'| = n + 1$.

Obviously, if $C \subseteq V$ is a minimal VC of G then C is also a minimal FVS of G'. It can be easily observed that, for any graph G, G' has a maximum cardinality minimal FVS without containing the vertex t. Also, if $F \subseteq V'$ is a minimal FVS of G' with $t \notin F$, then F is a minimal VC of G. From this, it follows that, if C_o is a maximum cardinality minimal VC of G then C_o is also a maximum cardinality minimal FVS of G'. It is important to observe that if F is a minimal FVS of G' with $t \in F$ then in polynomial time one can construct a FVS F' of G' with $|F| \leq |F'|$ and $t \notin F'$. Since an optimal solution of MAX-MIN-FVS for the instance G' does not contain the special vertex t, without loss of generality, we can assume that a minimal FVS of G' does not contain the vertex t. Hence, this is an S-reduction with the parameter $c = 1$.

Now, let $N = |V'|$. Hence $N = n+1$. As $N \geq 2$, for any $\epsilon > 0$, we have$(N-1)^{\frac{1}{2}-\epsilon} \geq \frac{1}{\sqrt{2}} N^{\frac{1}{2}-\epsilon}$. Thus $\frac{1}{2\sqrt{2}} n^{\frac{1}{2}-\epsilon} \geq \frac{1}{4} N^{\frac{1}{2}-\epsilon}$. Hence, by Theorem 6.3, the result follows. □

Similarly, we can prove

Theorem 6.8 *Unless P=NP, for any $\epsilon > 0$, there exists no polynomial-time algorithm to approximate MAX-MIN-FVS, for general graphs, within a factor of $\frac{1}{2\sqrt{2}} n^{\frac{1}{4}-\epsilon}$, where n is the number of vertices in an instance.*

6.2 Hardness Results for Bounded Degree Graphs

In this section, we establish APX-hardness of MIN-FVS and MAX-MIN-FVS for certain restricted class of undirected graphs. Regarding MIN-FVS, it is known that it can be solved in polynomial time for all graphs of maximum degree 3 [151], but it is not known whether MIN-FVS is NP-complete for graphs of maximum degree 4 or 5. However, as suggested by Fujito [63], it is easy to show that:

Proposition 6.1 *MIN-W-FVS-≤ 4 is NP-complete and also APX-complete.*

Proof From a 3-regular graph (an instance of MIN-VC-3) construct a vertex weighted graph $(G' = (V', E'), w)$ of degree at most 4 as follows. $V' = V \cup \{v^i | 1 \leq i \leq \frac{n}{2}\}$, $E' = E \cup [\cup_{1 \leq i \leq \frac{n}{2}} \{\{v_{2i-1}, v^i\}, \{v_{2i}, v^i\}\}] \cup [\{\{v^i, v^{i+1}\} | 1 \leq i < \frac{n}{2}\}]$, and $w(v) = 1$ for $v \in V$ and $w(v) = M$ for $v \in V' - V$, where $M > n$ is a positive large integer.

It can be shown that, any FVS F of G' with $w(F) < n$ is also a VC of G and any VC C of G is also a FVS of G'. From this, NP-completeness of MIN-W-FVS-$\leq$4 follows. Further, as MIN-W-FVS $\in$ APX [13], MIN-W-FVS-$\leq$4 $\in$ APX. Now, if C_o is a minimum VC of G, then C_o is also a minimum FVS of G' and they are of same cost. To a FVS F of G' with $w(F) = \sum_{v \in F} w(v) \geq n$ we assign V as a vertex cover of G. Hence, from any FVS F of G' we can construct a VC C of G such that $|C| \leq w(F)$. So $\frac{|C|}{|C_o|} \leq \frac{w(F)}{w(F_o)}$. From this it follows that, MIN-VC-3 $\leq_{AP}$ MIN-W-FVS-$\leq$4 with $\alpha = 1$, and so, MIN-W-FVS-$\leq$4 is APX-complete as MIN-VC-3 is so [3]. □

Next we show that MIN-FVS-6 is APX-complete.

Theorem 6.9 *MIN-FVS-6 is APX-complete.*

Proof As MIN-FVS is in class APX [13], it is enough to show that MIN-FVS-6 is APX-hard. Towards this we will show that MIN-VC-3 $\leq_L$ MIN-FVS-6.

Let $G = (V, E)$ be a 3-regular graph. From G construct a 6-regular graph $G' = (V', E')$ as follows: For every edge $\{v_i, v_j\} \in E$, let $V_{ij} = \{v_{ij}^1, v_{ij}^2, v_{ij}^3, v_{ij}^4, v_{ij}^5, v_{ij}^6, v_{ij}^7\}$ be the set of seven new vertices and $H_{ij} = (V_{ij}, E_{ij})$ be the graph obtained from the complete graph on V_{ij} by removing the edge $\{v_{ij}^1, v_{ij}^7\}$. Now $V' = V \cup [\cup_{\{v_i, v_j\} \in E} V_{ij}]$ and $E' = E \cup [\cup_{\{v_i, v_j\} \in E} [E_{ij} \cup \{\{v_i, v_{ij}^1\}, \{v_{ij}^7, v_j\}\}]]$, see Figure 6.4. Clearly G' is 6-regular.

Next we establish a few claims. But first note that if F is a FVS of G', then F contains at least 4 vertices from V_{ij}.

Claim 6.11 Let F be any FVS of G' containing exactly 4 vertices from V_{ij} for some $\{v_i, v_j\} \in E$. Then F must contain either v_i or v_j.

Proof If $v_{ij}^1 \in F$, then then the three vertices $v_{ij}^p, v_{ij}^q, v_{ij}^r$ not in F form a cycle in G'. Hence, F cannot be a FVS of G'.So $v_{ij}^1 \notin F$. By similar argument, one can show that

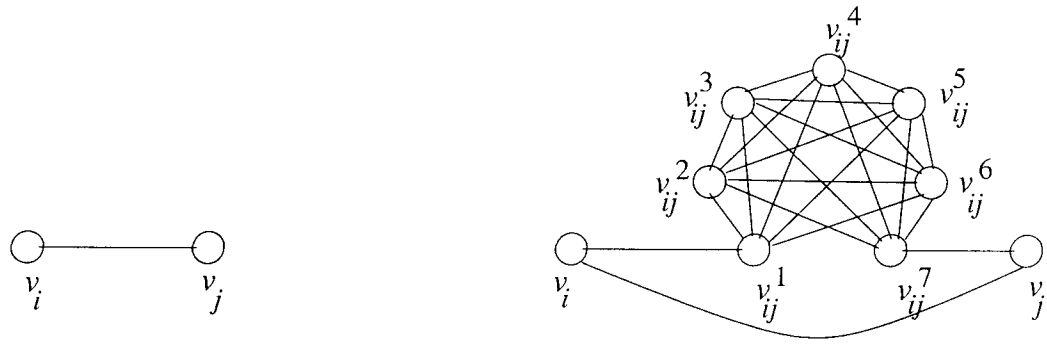


Figure 6.4: An edge $\{v_i, v_j\} \in E$ and corresponding subgraph in G'.

$v_{ij}^7 \notin F$. Hence, F contains 4 vertices from $v_{ij}^2, v_{ij}^3, v_{ij}^4, v_{ij}^5, v_{ij}^6$. Let $v_{ij}^l \notin F$ for some $2 \leq l \leq 6$. Since F is a FVS of G and the vertices $v_i, v_{ij}^1, v_{ij}^l, v_{ij}^7, v_j$ form a cycle in G', F must contain either v_i or v_j. □

To a FVS F of G', we associate the set C of vertices in G defined as $C = (F \cap V) \cup \{v_i \mid |F \cap V_{ij}| \geq 5 \text{ and } i < j\}$.

Claim 6.12 C is a VC of G and $|F| \geq |C| + 4|E| = |C| + 6n$.

Proof If C is not a VC of G, then there exists $\{v_i, v_j\} \in E$ such that $C \cap \{v_i, v_j\} = \phi$. By the definition of C, it follows that $|F \cap V_{ij}| \leq 4$ and $F \cap \{v_i, v_j\} = \phi$. If $|F \cap V_{ij}| < 4$, then F is not a FVS of G', so $|F \cap V_{ij}| = 4$. By Claim 6.11, F must contain either v_i or v_j. Otherwise F cannot be a FVS of G'. This contradicts that $F \cap \{v_i, v_j\} = \phi$. Hence, C is a VC of G.

Now $|F| = 4|E| + |F \cap V| + |\{v_i \mid |F \cap V_{ij}| \geq 5, i < j\}|$, as F contains at least 4 vertices from V_{ij} for each $\{v_i, v_j\} \in E$, and for the edges $\{v_i, v_j\} \in E$ such that $|F \cap V_{ij}| \geq 5$, F contains at least one more vertex from V_{ij} in addition to 4 vertices already considered. Hence, $|F| \geq |C| + 4|E| = |C| + 6n$ as G is a 3-regular and $|E| = \frac{3}{2}n$. □

Claim 6.13 For any VC C in G, the set $F = C \cup \{v_{ij}^2, v_{ij}^3, v_{ij}^4, v_{ij}^5 | \{v_i, v_j\} \in E\}$ is a FVS of G' such that $C = F \cap V$ and $|F| = |C| + 6n$.

Proof Take any cycle K in G'. If K is a cycle in G, then K has a vertex in $C \subset F$. Hence, F breaks all cycles in G. If K contains a vertex in G' and if K contains none of $v_{ij}^2, v_{ij}^3, v_{ij}^4, v_{ij}^5$ for any $\{v_i, v_j\} \in E$, then it must contain both v_i and v_j, and so K has a common vertex with $C \subset F$. Thus every cycle K in G' intersects F. So F is a FVS of G'.

Clearly, $C = F \cap V$ and so $|F| = |C| + 6n$. □

Claim 6.14 If F_o is a minimum FVS of G', then the associated set C_o is a minimum VC of G and $|F_o| = |C_o| + n$.

Proof Let C_o be the VC of G associated with F_o, and F be the FVS of G' associated with C_o as constructed in proof of Claim 6.13. Then $|F_o| \geq |C_o| + 6n = |F|$. But as F_o is a minimum FVS of G', equality holds, i.e., $|F_o| = |C_o| + n$. From this it follows that C_o is a minimum VC of G. □

Now to complete the proof of Theorem 6.9, first note that, any VC C in a 3-regular graph G contains at least $\frac{n}{2}$ vertices. To see this, if possible let $|C| < \frac{n}{2}$. Then note that $V - C$ is an IS of G and the number of edges between $V - C$ and C is $3|V - C| = 3n - 3|C| > \frac{3}{2}n$, the number of edges in G, which is not possible.

Now note that $|F_o| = |C_o| + 6n \leq |C_o| + 12|C_o| = 13|C_o|$. Hence, the first inequality of L-reduction holds with $\alpha = 13$. Next, for any FVS F of G', $|F| - |F_o| \geq |C| + 6n - |C_o| - 6n = |C| - |C_o|$. So the second inequality of L-reduction holds with $\beta = 1$. □

Next we shall consider MAX-MIN-FVS. Before that we note the following:

Lemma 6.1 *For any FVS F of a 6-regular graph $G = (V, E)$, $|F| > \frac{2}{5}n$.*

Proof Observe that, G has $3n$ edges and the subgraph $G[V - F]$ induced by $V - F$, is a forest. Since $G[V - F]$ has at most $n - |F| - 1$ edges and the number of edges removed by removing F is at most $6|F|$, we have $3n - 6|F| \leq n - |F| - 1$, i.e. $|F| > \frac{2}{5}n$. □

Theorem 6.10 *MAX-MIN-FVS-≤ 9 is APX-hard.*

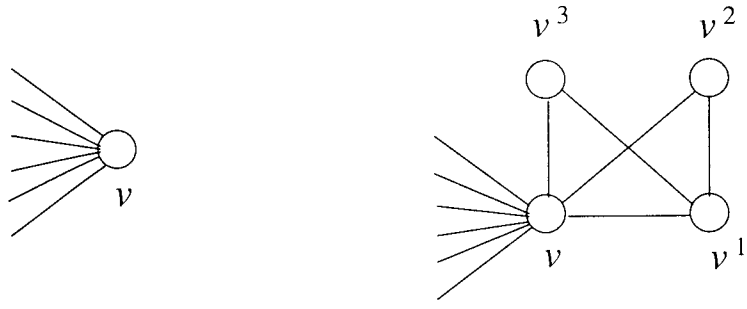


Figure 6.5: A vertex v in G and its corresponding neighbors in G'

Proof Let $G = (V, E)$ be a 6-regular graph. Construct a graph $G' = (V', E')$ of degree at most 9 as follows: $V' = V \cup \{v^1, v^2, v^3 | v \in V\}$ and $E' = E \cup \{(v, v^1), (v, v^2), (v, v^3), (v^1, v^2), (v^1, v^3)\ |v \in V\}$ (see Figure 6.5).

Let F be any minimal FVS of G'. Note that, for any $v \in V - F$, F contains either v^1 or both v^2 and v^3. Further, if $v \in F \cap V$, then $F \cap \{v^1, v^2, v^3\} = \phi$. To F we associate $C = F \cap V$, which is clearly a FVS of G. Note that $|F| \leq |C| + 2|V - F| = |C| + 2|V - C| = 2n - |C|$.

If C is a FVS of G, then we show that $F = C \cup \{v^2, v^3 | v \in V - C\}$ is a minimal FVS of G' and $|F| = 2n - |C|$. Clearly F is a FVS of G'. F is a minimal FVS of G' as (a) any $v \in C$ cannot be dropped from F as $\{v^1, v^2, v^3\} \cap F = \phi$ and $G'[\{v, v^1, v^2, v^3\}]$ contains cycles; and (b) for any $v \in V - C$, none of v^2 and v^3 can be dropped from F as $G'[\{v, v^1, v^2\}]$ and $G'[\{v, v^1, v^3\}]$ contain cycles. Obviously $|F| = 2n - |C|$.

Let F_o be a maximum minimal FVS of G' and $C_o = F_o \cap V$. We claim that $|F_o| = 2n - |C_o|$. For, if $|F_o| < 2n - |C_o|$, then $F = C_o \cup \{v^2, v^3 | v \in V - C_o\}$ is a minimal FVS of G' with $|F| = 2n - |C_o| > |F_o|$ contradicting the assumption that F_o is a maximum minimal FVS of G'. Next we claim that C_o is a minimum FVS of G. If possible, let C be a FVS of G with $|C| < |C_o|$. Then $F = C \cup \{v^2, v^3 | v \in V - C\}$ is a minimal FVS of G' with $|F| = 2n - |C|$. But as $|C| < |C_o|$, $|F| = 2n - |C| > 2n - |C_o| = |F_o|$, which is a contradiction to our assumption that F_o is a maximum minimal FVS of G'.

Now, $|F_o| = 2n - |C_o| < 5|C_o| - |C_o| = 4|C_o|$, (by Lemma 6.1). So, the first inequality of L-reduction holds with $\alpha = 4$. Next, for any minimal FVS F of G', $|F_o| - |F| \geq 2n - |C_o| - 2n + |C| = |C| - |C_o|$. So the second inequality of L-reduction holds with $\beta = 1$. □

Next we shall consider MAX-MIN-VC. First we have the following two simple lemmas.

Lemma 6.2 *For any 3-regular graph $G = (V, E)$ and any maximal IS I in G, $|I| \geq \frac{1}{4}|V|$.*

Proof Let I be any maximal IS of G. Then the subgraph $G[V - I]$ of G induced by $V - I$ is of degree at most 2. Thus $G[V - I]$ contains at most $|V - I|$ edges. Thus the number of edges in G is bounded above by $3|I| + |V - I|$.Hence $\frac{3}{2}|V| \leq |V| + 2|I|$, i.e., $\frac{|V|}{4} \leq |I|$. □

Lemma 6.3 *MAX-MIN-VC is k-approximable for graphs of maximum degree k, $k \geq 1$, and having no isolated vertex.*

Proof Let $G = (V, E)$ be any graph of maximum degree k, $k \geq 1$, and having no isolated vertex. We show that any minimal VC C of G is a k-approximable solution for MAX-MIN-VC. A minimal VC of G can be obtained by the simple algorithm:

Algorithm 6.1

1. **begin**
2. $C = V$;
3. **while** there exists $v_i \in C$ such that $N(v_i) \cap (V - C) = \phi$ *do*
4. $C = C - v_i$;
5. **od**
6. **end.**

where $N(v_i) = \{v_j | \{v_i, v_j\} \in E\}$.

Claim 6.15 For any minimal VC C of G, $|C| \leq \frac{kn}{k+1}$, where $n = |V|$.

Proof As C is a minimal VC of G, $V - C$ is a maximal IS of G. Also as $V - C$ is a maximal IS of G and G has no isolated vertices, for any $v_i \in V - C$, v_i is not adjacent to any $w \in V - C$, but is adjacent to some $v_j \in C$. Let $S_{v_i} = \{v_j \in C | \{v_i, v_j\} \in E\}$, for $v_i \in V - C$. Then $C = \cup_{v_i \in V-C} S_{v_i}$. Clearly, $\cup_{v_i \in V-C} S_{v_i} \subseteq C$. Conversely, let $v_j \in C$, then there exists $v_i \in V - C$ such that $\{v_i, v_j\} \in E$ and so $v_j \in S_{v_i}$. Hence, $C \subseteq \cup_{v_i \in V-C} S_{v_i}$. Therefore, $|C| \leq \sum_{v_i \in V-C} |S_{v_j}| \leq k|V - C| \leq nk - k|C|$. So $|C| \leq \frac{kn}{k+1}$. □

Claim 6.16 For any VC C of G, $|C| \geq \frac{n}{k+1}$.

Proof As $V - C$ is an IS of G and G has no isolated vertices, for any $v_i \in V - C$, v_i is not adjacent to any $v_k \in V - C$ but must be adjacent to some $v_j \in C$. Let $S_{v_j} = \{v_i \in V - C | \{v_i, v_j\} \in E\}$, for $v_j \in C$. Then $V - C = \cup_{v_j \in C} S_{v_j}$. Hence, $k|C| \geq |V - C|$, i.e. $|C| \geq \frac{n}{k+1}$. □

Let C_o be a maximum cardinality minimal VC of G. By Claim 6.15, $|C_o| \leq \frac{kn}{k+1}$, and so by Claim 6.16, $\frac{|C_o|}{|C|} \leq k$. □

Now we have

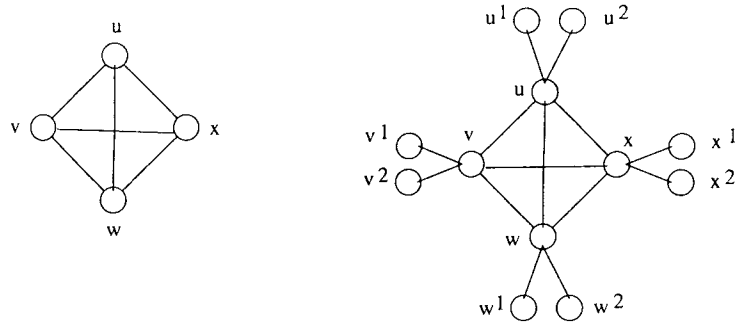
Theorem 6.11 *MAX-MIN-VC-≤5 is APX-complete.*

Proof Since MAX-MIN-VC is in class APX for bounded degree graphs (Lemma 6.3) and MAX-IS-3 is APX-complete [3], it is enough to show that MAX-IS-3 $\leq_L$ MAX-MIN-VC-≤5.

Let $G = (V, E)$ be a 3-regular graph. From G we construct $G' = (V', E')$ of degree at most 5 as follows: $V' = V \cup [\cup_{v \in V} \{v^1, v^2\}]$ and $E' = E \cup [\cup_{v \in V} \{\{v, v^1\}, \{v, v^2\}\}]$ (see Figure 6.6 for an example).

As in the proof of Theorem 6.3, we can prove the following:

(1) Any VC C of G' is a minimal VC of G' if and only if



G G'

Figure 6.6:

(a) $v \in C \cap V \Rightarrow v^1v^2 \notin C$ and
(b) $v \in (V - C) \Rightarrow \{v^1, v^2\} \subseteq C$.

Moreover, any minimal VC C of G' is of the form $C = (V - I) \cup [\cup_{v \in I}\{v^1, v^2\}]$, for some IS I of G where $I = V - (C \cap V)$. The IS I of G defined above is the IS of G associated with the minimal VC C of G'.

(2) C is a minimal VC of G' if and only if the associated I is a maximal IS of G, with $|C| = |I| + n$.

(3) C_o is a maximum cardinality minimal VC of G' if and only if the associated I_o is a maximum IS of G, with $|C_o| = |I_o| + n$.

Now, $|C_o| = |I_o| + n \leq |I_o| + 4|I_o| = 5|I_o|$ (by Lemma 6.2), so that, the first inequality of L-reduction holds with $\alpha = 5$. Next, for any minimal VC C of G', $|C_o| - |C| = |I_o| + n - |I| - n = |I_o| - |I|$, so that, the second inequality of L-reduction holds with $\beta = 1$. □

6.3 Hardness Results for Bounded Degree Digraphs

We know that MIN-FAS is APX-hard [98] and MAX-SUBDAG is APX-complete [132] for general digraphs. In this section, we show that these problems remain APX-hard even for k-total-regular digraphs for all $k \geq 4$. We also show that MIN-MAX-SUBDAG

is APX-hard for digraphs of maximum total degree 12 and MAX-MIN-VC is APX-hard for graphs of maximum degree 5. Regarding MIN-FAS, we first prove the following.

Lemma 6.4 *MIN-FAS-k* $\leq_L$ *MIN-FAS-*$(k+1)$*, for all* $k \geq 1$.

Proof We construct in polynomial time, from a k-total-regular digraph $G = (V, A)$, a $(k+1)$-total-regular digraph $G' = (V', A')$ where $V' = \{v^1, v^2 | v \in V\}$ and $A' = \{(u^1, v^1), (u^2, v^2) | (u, v) \in A\} \cup \{(v^1, v^2) | v \in V\}$. We shall denote $V^i = \{v^i | v \in V\}$ and $A^i = \{(u^i, v^i) | (u, v) \in A\}$, for $i = 1, 2$. From a minimal FAS S' of G', we construct a minimal FAS S of G as follows: $S = \{(u, v) | (u^1, v^1) \in S^1\}$ where, without loss of generality, we assume that $S' = S^1 \cup S^2$ with S^1 and S^2 are minimal FASs of $G^1 = (V^1, A^1)$ and $G^2 = (V^2, A^2)$, respectively, and $|S^1| \leq |S^2|$. It is easy to see that, if S'_o is a minimum FAS of G', then the corresponding S_o is a minimum FAS of G and $|S'_o| = 2|S_o|$. Further, for any minimal FAS $S' = S^1 \cup S^2$ of G', with $|S^1| \leq |S^2|$, $|S'| - |S'_o| = |S'| + |S_o| - 2|S_o| \geq 2(|S'| - |S_o|)$ so that $|S| - |S_o| \leq \frac{1}{2}(|S'| - |S'_o|)$. Thus, the two inequalities of L-reduction hold with $\alpha = 1$ and $\beta = \frac{1}{2}$. □

We now have the following.

Theorem 6.12 *MIN-FAS-k is APX-hard for all* $k \geq 4$.

Proof By Lemma 6.4, it is enough to show that MIN-FAS-4 is APX-hard. For this we show that MIN-VC-3 $\leq_L$ MIN-FAS-4.

We construct in polynomial time, from any 3-regular graph $G = (V, E)$ a 4-total-regular digraph $G' = (V', A')$ where $V' = \{v^1, v^2 | v \in V\}$ and $A' = \{(v^1, v^2) | v \in V\} \cup \{(u^2, v^1), (v^2, u^1) | \{u, v\} \in E\}$. Clearly, G' is a 4-total-regular digraph. For an example see Figure 6.3. For each FAS F of G', we associate a VC C of G defined as $C = \{v |$ either $(u^2, v^1) \in F$ or $(v^1, v^2) \in F\}$. Further, C is a VC of G with $|C| \leq |F|$. For every edge $\{u, v\} \in E$, as $(u^1, u^2, v^1, v^2, u^1)$ is a dicycle in G', F must contain at least one arc from this dicycle, and so, C must contain either u or v. Hence, C is a VC of G, and by the construction of C from F, $|C| \leq |F|$.

Next we show that, if F_o is a minimum FAS of G', then the associated VC C_o of G is a minimum VC of G and $|F_o| = |C_o|$. As F_o is a minimum FAS of G, without loss of generality, we may assume that F_o contains arcs only of the form (v^1, v^2) for some $v \in V$. For, if F_o contains arcs of the form (u^2, v^1), then F'_o obtained from F by replacing any such arc (u^2, v^1) by (v^1, v^2), we have F'_o a FAS of G' with $|F'_o| \leq |F_o|$. Hence, $C_o = \{v|(v^1, v^2) \in F_o\}$. If C_o is not a minimum VC of G, let C be a minimum VC of G. Now $F = \{(v^1, v^2)|v \in C\}$ is a FAS of G'. If not, let K be a cycle in G' without containing any arc of F. Let (u^2, v^1) be an arc in K. Then both (v^1, v^2) and (u^1, u^2) are in K. Hence, $u, v \notin C$, but $\{u, v\} \in E$ as $(u^2, v^1) \in K \subseteq A'$. This implies that C is not a VC of G, which contradicts our assumption. Hence F must be a FAS of G'. But then $|F| = |C| < |C_o| = |F_o|$, which contradicts that F_o is a minimum FAS of G'. Hence, C_o is a minimum VC of G.

Also, $|F_o| = |C_o|$ and for any FAS F of G', $|C| - |C_o| \leq |F| - |F_o|$. So the transformation from G to G' is an L-reduction with $\alpha = 1$ and $\beta = 1$. □

Similarly, for MAX-SUBDAG, we first prove the following.

Lemma 6.5 *MAX-SUBDAG-k* $\leq_L$ *MAX-SUBDAG-(k+1).*

Proof From a k-total-regular digraph $G = (V, A)$ construct a $(k+1)$-total-regular digraph $G' = (V', A')$ as constructed in the proof of Lemma 6.4. We shall use the notations used in the proof of Lemma 6.4 and let $B = \{(v^1, v^2)|v \in V\}$. Let $S = \{(u, v)|(u^1, v^1) \in S^1\}$ where, without loss of generality, we assume that $S' = S^1 \cup S^2 \cup B$ and $|S^1| \geq |S^2|$ and (V^i, S^i) is a maximal SUBDAG of G^i for $i = 1, 2$. Clearly, (V, S) is a maximal SUBDAG of G. Now, if S'_o is a maximum SUBDAG of G', $S'_o = S^1_o \cup S^2_o \cup B$, then note that $|S^1_o| = |S^2_o|$ and $S_o = \{(u, v)|(u^1, v^1) \in S^1_o\}$ is a maximum SUBDAG of G. Also, $|S'_o| = 2|S_o| + |B| = 2|S_o| + n \leq 2|S_o| + |A| \leq 4|S_o|$, since $|A| \leq 2|S_o|$. So, the first inequality of L-reduction holds with $\alpha = 4$. Next for any maximal SUBDAG (V', S') of G', $|S'_o| - |S'| = |S^1_o| + |S^2_o| + |B| - |S^1| - |S^2| - |B| = |S^1_o| + |S^2_o| - |S^1| - |S^2| \geq |S_o| - |S|$. So, the second inequality of L-reduction holds with $\beta = 1$. □

We now prove the following.

Theorem 6.13 *MAX-SUBDAG-k is APX-complete for any* $k \geq 4$.

Proof By Lemma 6.5, it is enough to show that MAX-SUBDAG-4 is APX-hard. For this we show that MIN-VC-3 $\leq_L$ MAX-SUBDAG-4.

Let $G = (V, E)$ be any 3-regular graph. From G we construct a 4-total-regular digraph $G' = (V', A')$ as constructed in Theorem 6.12. To each SUBDAG (V', S') of G', we associate a VC of G, C defined as $C = \{v |$ either $(u^2, v^1) \in A' - S'$ or $(v^1, v^2) \in A' - S'\}$. C is indeed a VC of G. If C is not then there exists $\{u, v\} \in E$ for which both u and v are not in C. For this edge $\{u, v\}$, G' contains the cycle $\{(u^2, v^1), (v^1, v^2), (v^2, u^1), (u^1, u^2)\}$. As none of u and v is in C, $A' - S'$ does not contain any of the arcs in this cycle. So, $A' - S'$ is not a FAS of G', which is a contradiction. Further, note that $|A'| = 2|E| + n$, $|C| < |A' - S'| = 2|E| + n - |S'|$ which implies that $|S'| \leq 2|E| + n - |C|$.

Now, if (V', S'_o) is a maximum SUBDAG of G', then the VC C_o of G associated with it is a minimum VC of G and $|S'_o| = 2|E| + n - |C_o|$. To see this, note that $A' - S'_o$ is a minimum FAS of G'. So by the proof of Theorem 6.12, C_o is a minimum VC of G and $|C_o| = |A - S'_o| = 2|E| + n - |S'_o|$.

Next note that, any VC C in a 3-regular graph G contains at least $\frac{n}{2}$ vertices (see proof of Theorem 6.9). Now, $|S'_o| = 2|E| + n - |C_o| = 3n + n - |C_o| = 4n - |C_o| \leq 8|C_o| - |C_o| = 7|C_o|$. So the first inequality of L-reduction from G to G' holds with $\alpha = 7$.

Next, for any SUBDAG (V', S') of G', $|S'_o| - |S'| = |A'| - |S'| - |A'| + |S'_o| = |A' - S'| - |A' - S'_o| \geq |C| - |C_o|$. Hence the second inequality of L-reduction holds with $\beta = 1$. □

Regarding MIN-MAX-SUBDAG, we have the following easy theorem.

Theorem 6.14 *MIN-MAX-SUBDAG-≤12 is APX-hard.*

Proof In the proof of Theorem 6.1, we constructed an instance G' of MIN-MAX-SUBDAG from an instance G of MAX-SUBDAG in such a way that if G is 4-total-regular then, every vertex in G' is of total degree at most 12. Since MAX-SUBDAG-4 is APX-complete, the result follows. □

Finally, we have,

Theorem 6.15 *MAX-MIN-FVS-$\leq$6, for digraphs, is APX-hard.*

Proof In the proof of Theorem 6.5, we constructed an instance G' of MAX-MIN-FVS, for digraphs, from an instance of G of MAX-MIN-VC, in such a way that if G is of degree at most 5, then G' is of total-degree at most 6. Since MAX-MIN-VC-$\leq$5 is APX-complete, it follows that MAX-MIN-FVS-$\leq$6 is APX-hard. □

Chapter 7

Concluding Remarks and Some Open Problems

In this thesis, we have investigated various issues regarding approximability of several optimization problems of practical interest which are related to linear ordering problems. We have obtained a few positive results about approximate solutions of some of these problems or their variants, and many negative results expressing hardness of approximation of several of these problems and their variants. Because of their practical interest, it is important to obtain as complete understanding as possible regarding their approximability. While we have made a modest contribution towards this goal, there are many questions, arising out of our work, which remain unresolved. Answer to these questions will make father progress towards achieving this goal.

We formulate these questions immediately after reviewing the contexts in which they arise and these are organized as follows. In Section 7.1, we first recall the main results of Chapter 3 and Chapter 4 and then describe some problems that are yet to be solved. We do the same regarding Chapter 5 and Chapter 6 in Section 7.2 and Section 7.3, respectively.

7.1 Problems Related to Chapters 3 and 4

In Chapter 3, our main contributions consists of (1) proving strong NP-completeness of LOP, and (2) establishing *strict*-equivalence of MIN-LOP (or MIN-LOP^p) with MIN-QAP(S) and MIN-W-FAS (respectively, with MIN-W-FAS^p, MIN-W-FVS^p and

their unweighted versions), *strict*-equivalence of MAX-LOP and MAX-W-SUBDAG, and AP-equivalences of MAX-LOP^p, MAX-W-SUBDAG^p and MAX-SUBDAG. Such equivalences establish that they have similar approximability properties. In Chapter 4, our main contributions consists of (1) proving some negative results that seem to provide some evidences towards our conjecture that MIN-LOP$\notin$APX, if P$\neq$NP, and (2) showing that, while $\triangle_t$-LOP, $0 < t < 2$, have good approximate solutions and $\triangle_t$-LOP$\in$APX, $\triangle_t$-LOP$\notin$PTAS, unless P=NP.

Regarding the few problems, arising in this context, that need to be solved, first note that for LOP, the structure of input digraph is fixed, viz., a complete digraph on n vertices, and so, any variant of LOP can arise only by putting some restrictions on (or, by some generalization of) the weight function. However, the graph problems which are equivalent, in the general form, with MIN-LOP (or MAX-LOP) admit of variants obtained by putting restrictions also on the structure of the input graph/digraph. So our first problem is:

Problem 7.1 Is it possible to establish *strict* or AP-equivalence among other interesting variants of the graph problems related to LOP obtained by putting restrictions on the structure of the input graph/digraph?

Another problem that arise naturally is:

Problem 7.2 Can we obtain an approximation algorithm, that is better than an $O(\log n \log\log n)$-approximation algorithm, for MIN-LOP and its equivalent problems? If not, can we prove that no better approximation algorithm can be obtained for these problems?

Of course, the major open problem is:

Problem 7.3 Can we resolve the conjecture that MIN-LOP$\notin$APX, unless P=NP?

In view of our Remark 4.1, our next problem is:

Problem 7.4 Is MIN-Dom-Set$\leq_{AP}$MIN-W-FAS?

Using Paz-Moran characterization of PTAS, we have shown that $\triangle_t$-LOP$\notin$PATS, unless P=NP, but we have not able to resolve the following:

Problem 7.5 Is $\triangle_t$-LOP APX-complete, for $0 < t < 2$?

As remarked earlier, we have used a notion of triangle inequality that is stronger than the usual notion of triangle inequality [4]. So, it remains to solve the following:

Problem 7.6 Using the parametrized version of standard notion of triangle inequality, can we show that $\triangle_t$-MIN-LOP$\in$APX?

Of course, a general problem for further investigation is:

Problem 7.7 Can we get good approximate solutions for other interesting variants of some of the problems we have considered?

7.2 Problems related Chapter 5

In Chapter 5, apart from proving NP-completeness of OEOP and describing some heuristics for OEOP, we have shown that OEOP(S)$\leq_{strict}$MIN-LOP and OEOP$\leq_{strict}$ MIN-LOP(Set). Some problems, related to these, that may be worth investigating are as follows:

Problem 7.8 Can we establish *strict* or AP-equivalence of OEOP(S) (respectively, OEOP) with some suitable variant of MIN-LOP (respectively, MIN-LOP(Set))? It is also of interest to find possible other applications of MIN-LOP(Set).

Problem 7.9 Can we get efficient $f(n)$-approximation algorithms for OEOP(S) (and OEOP) for suitably slowly growing function $f(n)$ other than what is suggested by the above mentioned reductions? Else can we establish lower bound on approximation factor for such problems?

Problem 7.10 How well the heuristics proposed for OEOP perform on an average in practice or in theory? While theoretical average-case analysis may be difficult, it is desirable to do at least an empirical average-case performance analysis for these heuristics.

7.3 Problems Related to Chapter 6

In Chapter 6, we have established hardness results for several NP-hard optimization problems related to MIN-LOP. These problems are variations or generalizations of well known NP-optimization problems on graphs/digraphs. While for MAX-MIN-VC and MAX-MIN-FVS we have established strong results like Håstad [87] concerning MAX-IS, for others we have just shown them to be APX-hard (or APX-complete).

So the first problem that is worth investing is:

Problem 7.11 Can we prove strong results about hardness of approximation, like those of Håstad for MAX-IS, for the problems which have been shown to be APX-hard?

In particular,

Problem 7.12 Can we obtain a hardness result for MAX-MIN-FAS similar to those (Theorems 6.7-6.8) for MAX-MIN-FVS?

Despite such negative results, as presented in this chapter, it is important to investigate the following.

Problem 7.13 Can we obtain $f(n)$-approximation algorithms for the problems considered with suitable slowly growing function $f(n)$?

Bibliography

[1] A.V. Aho, R. Sethi AND J. D. Ullman,*Compilers : Principles, Techniques and Tools*, Addison Wesley, 1986.

[2] R. K. Ahuja, T. L. Magnanti AND J. B. Orlin, *Network Flows, Theory, Algorithms and Applications*, Prentice Hall, Englewood Cliffs, 1993.

[3] P. Alimonti AND V. Kann, Hardness of approximating problems on cubic graphs, in *Proc. 3rd Italian Conf. on Algorithms and Complexity*, LNCS-1203, Springer-Verlag, 1997, 288-298.

[4] T. Andreae AND H.-J. Bandelt, Performance guarantees for the approximation algorithms depending on parametrized triangle inequality, *SIAM J. Discrete Math.*, vol. 8, 1995, 1-16.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan AND M. Szegedy, On the intractability of approximation problems, *Proc. of 33rd. IEEE FOCS*, 1992.

[6] S. Arora, C. Lund, R. Motwani, M. Sudan AND M. Szegedy, Proof verifications and hardness of approximation problems, *Proc. 33rd IEEE Symposium on Foundations of Computer Science*, 1992, 14-23.

[7] S. Arora AND S. Safra, Probabilistic checking of proofs: A new characterization of NP, In *Proc. 36th IEEE Symp. on Foundations of Computer Science*,1992, 2-13.

[8] G. Ausiello, P. Crescenzi AND M. Protasi, Fundamental Study: Approximate solution of NP optimization problems, *Theoretical Computer Science*, vol. 150, 1995, 1-55.

[9] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela AND M. Protasi, *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*, Springer-Verlag, Berlin Heidelberg, 1999.

[10] G. Ausiello, A. D'Atri AND M. Protasi, Structure preserving reductions among convex optimization problems, *Journal of Computer and System Sciences*, vol. 21, 1980, 136-153.

[11] G. Ausiello, A. Marchetti-Spaccamela AND M. Protasi, Towards a unified approach for the classification of NP-complete optimization problems, *Theoret. Comput. Sci.*, vol. 12, 1980, 83-96.

[12] A. Backer AND D. Geiger, Approximation algorithm for the loop cutset problem, in *Proc. of the 10th Conf. on Uncertainty in artificial Intelligence*, Morgan Kaufman, 1994, 167-188.

[13] V. Bafna, P. Berman AND T. Fujito, Constant ratio approximations of feedback vertex sets in weighted undirected graphs, in *6th Annual International Symposium on Algorithms and Computation*, 1995.

[14] B. S. Baker, Approximation algorithms for NP-complete on planar graphs. In *Proc. of 24th Annual IEEE Sympos. on Foundations of Computer Science*,1983, 265-273.

[15] H.-J. Bandelt, Y Crama AND F. R. C. Spieksma, Approximation algorithms for multidimensional assignment problems with decomposable costs, *Discrete Appl. Math.*, vol. 49, 1994, 25-50.

[16] J. Bang-Jensen AND G. Gutin, *Digraphs: Theory, Algorithms and Applications*, Springer 2001.

[17] R. Bar-Yehuda AND S. Even, A local-ratio theorem for approximating the weighted vertex cover problem, vol. 25 of *Annals of Discrete Mathematics*, pages 27-46, Elsevier science publishing company, Amsterdam, 1985.

[18] R. Bar-Yehuda, D. Geiger, J. Noar AND R. M. Roth, Approximation algorithms for the feedback vertex set problems with application to constraint satisfaction and Bayesian inferences, *SIAM Jr. on Computing*, vol. 27, 1998, 942-959.

[19] M. Bellare, S. Goldwasser, C. Lund AND A. Russell, Efficient probabilistically checkable proofs and applications to approximation, In *Proc. 25th ACM Symposium on Theory of Computing*, 1993, 294-304.

[20] M. Bellare AND M. Sudan, Improved non-approximability results, In *Proc. 25th ACM Symposium on Theory of Computing*, 1994, 184-193.

[21] B. Berger AND Shor P. W., Approximation algorithms for the maximum acyclic subgraph problem, *1st Annual ACM-SIAM Symposium on Discrete Algorithms*, ACM-SIAM, 1990, 236-243.

[22] L. Berman AND J. Hartmanis, Isomorphism and density of NP and other complete sets, *SIAM J. Computing*, vol. 6, 1977, 305-322.

[23] P. Berman AND G. Schnitger, On the complexity of approximating the independent set problem, *Inform. and Comput.*, vol. 96, 1992, 77-94.

[24] K. Boenchendorf, Reihenfolgenprobleme/Mean-Flow-Time Sequencing, *Mathematical Systems in Economics*, vol. 74, 1982.

[25] S. H. Bokhari, Assignment problems in parallel and distributed computing, Kluwer Academic Publishers, Boston, 1987.

[26] R. Boppana AND M. M. Halldórsson, Approximating maximum independent set by excluding subgraphs, *Bit*, vol. 32 1992, 180-196.

[27] A. Borobia, $(0, \frac{1}{2}, 1)$ matrices which are extreme points of the generalized transitive tournament polytope, *Linear Algebra Appl.*, vol. 220, 1995, 97-110.

[28] D. P. Bovet AND P. Crescenzi, *Introduction to the Theory of Complexity*, Prentice Hall 1994.

[29] A. Brandstädt, V. D. Chepoi AND F. F. Dragan, The algorithmic use of hypertree structure and maximum neighborhood orderings, *Discrete Appl. Math.*, vol. 82, 1998, 43-77.

[30] A. Brandstädt AND D. Kratsch, On domination problems for permutation and other graphs, *Theoretical Computer Science*, vol. 54, 1987, 181-198.

[31] R. A. Bruladi AND G. -S. Hwang, Generalized transitive tournaments and double stochastic matrices, *Linear Algebra Appl.*, vol. 172, 1992, 151-168.

[32] R. E. Burkard, Locations with spatial interactions: the quadratic assignment problem, in *Discrete Location Theory*, P. B. Mirchandani and R. L. Francis, Eds., Wiley Intersection Series in Discrete Mathematics and Optimization, 1990, 387-437.

[33] M. Cai, X. Deng AND W. Zang, A TDI system and its application to approximation algorithm, *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, 1998.

[34] S. Chanas AND P. Kobylański, A new heuristic algorithm solving the linear ordering problem, *Computational Optimization and Applications*, vol. 6, 1996, 191-205.

[35] M. S. Chang, Efficient algorithms for the domination problems on interval and circular-arc graphs, *SIAM Jr. on Computing*, vol. 27, 1998, 1671-1694.

[36] A. Chaudhary AND S. Vishwanathan, Approximation algorithms for achromatic number, *Proc. 8th Ann. ACM-SIAM Symp. on Discrete Algorithms*, ACM-SIAM, 1997, 558-563.

[37] G. A. Cheston, G. Fricke, S. T. Hedetniemi AND D. P. Jacobs, On the computational complexity of upper fractional domination. *Discrete Appl. Math.*, vol. 27, 1990, 195-207.

[38] N. Christofides, Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburg, 1976.

[39] F. A. Chudak, M. X. Goemans, D. S. Hochbaum AND D. P. Williamson, A primal-dual interpretation of two 2-approximation algorithms for the feedback vertex set problem in undirected graphs, *Operations Research Letters*, vol. 22, 1998, 111-118.

[40] S. Cook, The complexity of theorem proving procedures, in *Proceedings of the 3rd ACM Symposium of Theory of Computing*, 1971, 151-158.

[41] P. Crescenzi, V. Kann, R. Silvestri AND L. Trevisan, Structures in approximation classes, in *1st. Annu. Int. Conf. on Computing and Combinatorics*, LNCS-959, Springer-Verlag, 1995, 539-548.

[42] P. Crescenzi AND A. Panconesi, Completeness in approximation classes. *Information and Computation*, vol. 93, 1991, 241-262.

[43] P. Crescenzi, R. Silvestri AND L. Trevisan, On weighted vs unweighted versions of combinatorial optimization problems, *4th Isral Symp. on Theory of Comp. and Systems*, 1996.

[44] P. Crescenzi AND L. Trevisan, On approximation scheme preserving reducibility and its applications, *Proc. of 14th Conf. on Foundations of Software Technology and Theoret. Comp. Sci.*, LNCS, 880, 330-341, 1994.

[45] A. Cruse, On removing a vertex from the assignment polytope, *Linear Algebra Appl.*, vol. 26, 1979, 45-57.

[46] D. G. Corneil AND Y. Perl, Clustering and domination in Perfect graphs, *Discrete Appl. Math.*, ivol. 9, 1984, 27-39.

[47] J. W. Davidson AND A. M. Holler, Subprogram Inlining: A study of its effect on program execution time, IEEE Tr. on *Software Engineering*, vol. 18(2), 1992, 89-102.

[48] R. Dechter, Enhancement schemes for constraint processing: backjumping, learning and cutset decomposition, *Arti. Intell.* vol. 41, 1990, 273-312.

[49] T. Dridi, Sur les distributions binaries associees a des distributions ordinales. *Math. et Sci. Humaines*, vol. 69, 1980, 15-31.

[50] D. Du AND K. Ko, *Theory of Computational Complexity*, John Wiley & Sons, 2000.

[51] J. Edmonds, Paths, trees and flowers, *Canad. J. Math.*, vol. 17, 1965, 449-467.

[52] J. Edmonds AND R. Karp, Theoretical improvements in algorithmic efficiency for network flow problem, *Journal of the ACM*, vol. 19, 1972, 248-264.

[53] G. Even, J. Noar, B. Schieber AND M. Sudan, Approximating minimum feedback sets and multicuts in directed graphs, in "Proc. 4th. MPS Conference on Integer Prog. and Combinatorial Optimization (IPCO)", pp. 14-24, University of Copenhagen, Copenhagen, 1995.

[54] S. Even AND O. Kariv, An $O(n^{2.5})$ algorithm for maximum matching in general graphs, *Proc. of the 16th Sympo. on FOCS*, 1975, 100-112.

[55] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, pp. 43-73 in *Complexity of Computation*, edited by R. M. Karp, SIAM-AMS Proceedings, vol. 7, 1974.

[56] U. Feige, A threshold of ln n for approximating set cover, *Journal of the ACM*, vol. 45(4), 1998, 634–652.

[57] U. Feige, S. Goldwasser, L. Lovász, S. Safra AND M. Szegedy, Interactive proofs and the hardness of approximating cliques, *Journal of the ACM*, vol. 42, 1996, 268-292.

[58] U. Feige AND J. Kilian, Zero knowledge and the chromatic number, *Proc. Comp. Complexity* 1996.

[59] P. Festa, P. M. Pardalos AND M. G. C. Resende, Feedback set problems, AT AND T Labs Reasearch Technical Report: 99.2.2, 1999.

[60] L. R. Ford AND D. R. Fluckerson, Maximal flow through a network, Canadian Jr. of Math., vol. 8, 1956, 399-404.

[61] M. Fraber, Independent domination in chordal graphs, *Oper. Res. Letters*, vol. 1, 1982, 134-138.

[62] M. Fraber AND J. M. Keil, Domination in permutation graphs, *Jr. of Algorithms*, vol. 6, 1985, 309-321.

[63] T. Fujito, Personal communication (1999).

[64] M. Fürer AND Raghavachari B., Approximating the minimum-degree Steiner tree to within one of optimal, J. Algorithms, vol 17, 1994, 409-423.

[65] H. N. Gabow, A representation of crossing set families with application to submodular flow problems, *Proc. 4th Annual ACM-SIAM Sympo. on Disc. Algorithm*, 202-211, 1993.

[66] M.R. Garey, R. L. Graham AND J. D. Ullman, Worst case analysis of memory allocation algorithms, *Proc. 4th. ACM Symposium on Theory of Computing*, ACM, 1972, 143-150.

[67] M.R. Garey AND D.S. Johnson, The complexity of near-optimal graph coloring, *Journal of the ACM*, vol. 23, 1976, 499-508.

[68] M.R. Garey AND D.S. Johnson, Strong NP-completeness results: motivations, examples, and applications, *J. Assoc. Comput. Mach.*, vol. 25, 1978, 499-508.

[69] M.R. Garey AND D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman and Company, New York 1979.

[70] F. Gavril, Algorithm for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph, *SIAM J. Computing*, vol. 1, 1972, 180-187.

[71] A. M. Geoffrion AND G. W. Graves, Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/LP approach, *Operation Research*, vol. 24, 1976, 595-610.

[72] M. X. Goemans AND D. P. Willamson, Primal-Dual approximation algorithms for feedback problems in planar graphs, *Combinatorica* 18, 37-59. 1998. Preliminary version appeared in IPCO '96, 147-161.

[73] M. X. Goemans AND D. P. Willamson, 0.878-approximation algorithm for MAX-CUT and MAX-2SAT, *Proc. 26th Annual ACM Symposium on Theory of Computing*, 1994, 422-431.

[74] M. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[75] R. L. Graham, Bounds for multiprocessing timing anomalies, *SIAM J. Applied Mathematics*, vol. 17, 1969, 263-269.

[76] R. Greenlaw, The parallel complexity of approximate algorithms for the maximum acyclic subgraph problem, *Math. Systems Theory*, vol. 25, 1992, 161-175.

[77] M. Grötschel, M. Jünger AND G. Reinelt, A cutting plane algorithm for linear ordering problem, *Oper. Res.*, vol. 32, 1984, 1195-1220.

[78] M. Grötschel, M. Jünger AND G. Reinelt, On the acyclic subgraph polytope, *Math. Programming*, vol. 33, 1985, 28-42.

[79] M. Grötschel, M. Jünger AND G. Reinelt, Acyclic subdigraphs and linear orderings: Polytopes, facets, and a cutting plane algorithm, *Graphs and Orders*, 1985, 217-264.

[80] S. W. Hadley, F. Rendl AND H. Wolkowicz, A new lower bound via projection for the quadratic assignment problem, *Math. Oper. Res.*, vol. 17, 1992, 727-739.

[81] M. M. Halldórsson, Approximating the minimum maximal independence number, *Info. Proc. Letters*, vol. 46, 1993, 169-172.

[82] M. M. Halldórrson, Approximations via partitioning, Technical Report IS-RR-95-0003F, JAIST, March 1995.

[83] F. Harary, *Graph Theory*, Addition-Wesley, Reading, MA, 1969.

[84] F. Harary AND S. Hedetniemi, The achromatic number of a graph, *Jr. of Combi. Theory*, vol. 8, 1970, 154-161.

[85] F. Harary, Maximum versus minimum invariants for graphs, *Jr. of Graph Theory*, vol. 7, 1983, 275-284.

[86] R. Hassin AND S. Rubinstein, Approximations for the maximum acyclic subgraph problem, *Info. Proc. Letters*, vol. 51, 1994, 133-140.

[87] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, In *Proc. 37th IEEE Sympo. on Foundation of Comput. Sci.*, 1996, 627-636.

[88] M. S. Hecht AND J. D. Ullman, Flow graph reducibility, SIAM J. Comput., vol. 1, 1972, 188-202.

[89] M. S. Hecht AND J. D. Ullman, Characterization of reducible flow graphs, *Jr. of the ACM*, vol. 21, 1974, 367-375.

[90] D. S. Hochbaum, *Approximation Algorithms for NP-hard Problems*, PWS Publishing Company, 1997.

[91] M. Held AND R. M. Karp, A dynamic programming approach to sequencing problems, *SIAM J. Appl. Math.*, vol. 10, 1962, 196-210.

[92] J. L. Hennessy AND Patterson D.A., *Computer Architecture: A quantitative Approach* (2nd Edn.), Morgan Kaufmann, 1996.

[93] J. Hopcroft AND R. M. Karp, An $n^{2.5}$ algorithm for maximum matching in bipartite graphs, *SIAM Journal on Computing*, vol. 2, 1973, 223-231.

[94] O. H. Ibarra AND C. E. Kim, Fast approximation for the knapsack and sum of subset problems, *Journal of the ACM*, vol. 22, 1975, 463-468.

[95] R. W. Irving, On approximating the minimum independent dominating set, *Info. Proc. Lett.*, vol. 37, 1991, 197-200.

[96] D. Johnson, Approximation algorithms for combinatorial problems, *Jr. of Comp. and Syst. Sci.*, vol. 9, 1974, 256-278.

[97] D. Johnson, Finding all the elementary circuits of a directed graph, *SIAM J. Computing*, vol. 4, 1975, 77-84.

[98] V. Kann, *On the Approximability of NP-complete Optimization Problems*, Ph. D. thesis, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm 1992.

[99] V. Kann, Polynomially bounded minimization problems that are hard to approximate, *Nordic Journal of Computing*, vol. 1, 1994, 317-331.

[100] V. Kann, Strong lower bound on the approximability of some NPO PB-complete maximization problems, 1995.

[101] S. Khanna AND R. Motwani, Towards a syntactic characterization of PTAS, *Proc. 28th ACM Symposium on Theory of Computing*, 1996, 329-337.

[102] S. Khanna, R. Motwani, M. Sudan AND U. Vazirani, On syntactic versus computational views of approximability, in *Proc. 35th Ann. IEEE Symp. on Foundations of Computer Science*, 1994, 819-836.

[103] R. M. Karp, Reducibility among combinatorial problems, in R.E. Miller and T.W. Thatcher (eds.), Complexity of computer computations, Plenum Press, New York, 1972, 85-103.

[104] P. Kolaities AND M. Vardi, 0-1 laws and decision problems for fragments of second-order logic, *Proc. 3rd IEEE Symp. on Logic In Comp. Sci.*, 1998, 2-11.

[105] B. Korte AND W. Oberhofer, Zur triangulation von input-output-matrizen, *Jahrbuch. Nationaloekon. Statist.*, vol. 182, 1969, 398-433.

[106] B. Korte AND R. Schrader, On the existence of fast approximation schems, in *Nonlinear Programming*, Academic Press, New York, 1981, 415-437.

[107] B. Korte AND D. Hauseman, An analysis of the greedy heuristic for independence systems, *Annals of Discrete Mathematics*, vol. 2, 1978, 65-74.

[108] J. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, *Proceeding of the AMS*, vol. 7, 1956, 48-50.

[109] A. Kunzmann AND H. J. Wunderlich, An analytical approach to the partial scan problem, *J. of Electronic Testing: Theory and Applications*, vol. 1, 1990, 163-174.

[110] L. A. Levin, Universal sorting problems, *Problems of Information Transmission*, vol. 9, 1973, 265-266.

[111] E. L. Lawler AND D. E. Wood, Branch and Bound methods: a survey, *Operations Research*, vol. 14, 1966, 699-719.

[112] T. Leighton AND S. Rao, An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms, In *Procc. of the 29th Annual Symposium on FOCS*, 1988, 422-431.

[113] E. L. Lloyd, M. L. Soffa AND C. C. Wang, On locating minimum feedback vertex sets, *Journal of Computer System and Sciences*, vol. 37, 1988, 292-311.

[114] F. L. Luccio, Almost exact minimum feedback vertex set in meshes and butterflies, *Information Processing Letters*, vol. 66, 1998, 59-64.

[115] C. L. Luchesis AND D. H. Younger, A minmax theorem for directed graphs, *J. London Math. Soc.*, vol. 17, 1978, 369-374.

[116] C. Lund AND M. Yannakakis, On the hardness of approximating minimization problems, *J, ACM*, vol. 41, 1994, 960-981.

[117] E. W. Mayr, H. J. Prömel AND A Steger (Eds.), *Lectures on Proof Verifications and Approximation Algorithms*, Springer, 1998.

[118] D. F. Manlove, *Minimaximal and maximinimal optimization problems: a partial order-based approach*, Ph. D. Thesis, University of Glasgow 1998.

[119] K. Mehlhorn, *Graph Algorithms and NP-completeness*, Vol. 2 of *Data Structures and Algorithms*, Springer-Verlag, 1984.

[120] S. Mishra, K. Sikdar AND M. Satpathy, Optimizing register spills for eager functional languages, *Proc. of Computational Science-ICCS* 2001 (Part II), International Conference, San Francisco, May 2001, LNCS Volume 2074, 128- 137 (accepted in *Future Generation Computer Systems*, Elsevier).

[121] S. Mishra AND K. Sikdar, On Approximability of Linear Ordering and Related NP-optimization Problems on Graphs, (Extended Abstract), *Electronic Notes in Discrete Mathematics*, vol. 8, Elsevier Science Publishers, edited by H. Broersma, U. Faigle, J. Hurink and S. Pickl, 2001, Full version submitted to *Disc. Appl. Maths.*.

[122] S. Mishra AND K. Sikdar, On the hardness of approximating some NP-optimization problems related to minimum linear ordering problem, *Theoretical Informatics and Applications*, vol. 35, 2001, 287-309.

[123] J. W. Moon, *Topics on Tournaments*, Holt, Rinehat and Winston, Inc., New York, 1968.

[124] R. Mueller, On the partial order polytope of a digraph, *Mathematical Programming*, vol. 73, 1996, 31-49.

[125] G. L. Nemhauser AND L. E. Trotter, Vertex packing: structural properties and algorithms, *Math. Prog.*, vol. 8, 1975, 232-248.

[126] A. Newman, *Approximating the maximum acyclic subgraph*, MS Thesis, M.I.T., June 2000.

[127] Z. Nutov AND M. Penn, On non-$\{0, \frac{1}{2}, 1\}$ extreme points of the transitive tournament polytope, *Linear Algebra Appl.*, vol. 233, 1996, 149-159.

[128] P. Orponen AND H. Manila, On approximation preserving reductions: Complete problems and robust measures. Technical Report C-1987-28, Department of Computer Science, University of Helsinki, 1987.

[129] P. M. Pardalos, E. Rendl AND H Wolkowicz, The quadratic assignment problem: A survey and recent developments, in *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, vol. 16, 1994, 1-41.

[130] C. H. Papadimitriou, *Theory of Computational Complexity* Addision-Wesley Publishing Company, Inc. 1994.

[131] C. H. Papadimitriou AND K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, Englewood Cliffs, NJ, 1982.

[132] C. H. Papadimitriou AND M. Yannakakis, Optimization, Approximation, and Complexity Classes. *J. Comput. System Sci.*, vol. 43, 1991, 425-440.

[133] V. R. Pratt, Every prime has a succinct certificate, *SIAM J. Comput.*, vol. 4, 1975, 214-220.

[134] A. Paz AND S. Moran, Nondeterministic polynomial optimization problems and their applications, *Theo. Comp. Sci.*, vol. 15, 1981, 251-277.

[135] M. Penn AND Z. Nutov, Minimum feedback arc set and maximum integral dicycle packing in $K_{3,3}$-free digraphs, 1993.

[136] K. Peters, R. Laskar AND S. T. Hedetniemi, Maximinimal/Minimaximal connectivity in graphs. *Ars Combinatoria*, vol. 21, 1986, 59-70.

[137] Peyton Jones S.L., *Implementation of Functional Programming languages*, Printice Hall, 1987.

[138] R. C. Prim, Shortest connection network and some generalizations, BSTJ, vol. 36, 1957, 1387-1401.

[139] M. Queyranne, Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems, *Oper. Res Letters*, vol. 4, 1986, 231-234.

[140] V. Ramachandran, Finding a minimum feedback arc set in reducible flow graphs, *J. Algorithmsi*, vol. 9, 1988, 299-313.

[141] S. Sahni AND T. Gonzalez, P-complete approximation problems, *Journal of the ACM*, vol. 23, 1976, 555-565.

[142] P. D. Seymour, Packing directed circuits fractionally, *Combinatorica*, vol. 15, 1995, 281-288.

[143] D. B. Shmoys, Computing near-optimal solutions to combinatorial optimization problems, in W. Cook, L. Lovasz, P. Seymour (eds.), *Special year on combinatorial optimization*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 20, AMS, 1995.

[144] M. Sipser, *Introduction to the Theory of Computation*, PWS Publishing, 1997.

[145] P. Slater, Inconsistencies in a schedule of paired comparisons, *Biometrica*, vol. 48, 1961, 303-312.

[146] E. Speckenmeyer, On the feedback problems in digraphs, in *Graph-Theoretic Concepts in Computer Science*, 15th International Workshop WG'89 (LNCS-411), Castle Rddue, The Netherlands, June, 1989.

[147] L. Steinberg, The backbord wiring problem: A placement algorithm, *SIAM Review*, vol. 3, 1961, 37-50.

[148] R. E. Tarjan, *Data Structures and Network Algorithms*, SIAM, Philadelphia, P. A., 1983.

[149] R. E. Tarjan AND Trojanowski, Finding maximum independent set, *SIAM Computing*, vol. 6, 1977, 537-546.

[150] L. Trevisan, *Reductions and (Non)-Approximability*, Ph. D. thesis, Computer Science Department, University of Rome "La Sapienza", 1997.

[151] S. Ueno, Y. Kajtani AND S. Gotoh, On the nonseparating independent set problem and feedback set problem for graphs with no vertex exceeding three, *Disc. Math.*, vol. 72, 1988, 355-360.

[152] Vizing, V. G., On an estimate of the chromatic class of a p-graph, Diskret. Analiz., vol, 3, 1964, 23-30.

[153] H. P. Young, On permutations and permutation polytopes, *Math. Prog. Study*, vol. 8, 1978, 128-140.