Dragoš Cvetković, Mirjana Čangalović
and Vera Kovačević-Vujčić

OPTIMIZATION AND HIGHLY INFORMATIVE GRAPH INVARIANTS

CONTENTS

1. Introduction 7
1.1. Travelling Salesman Problem 8
1.2. Graph Spectra and Other Graph Invariants 9
1.3. Semidefinite Programming 13
2. Using Spectral Invariants in Problems of Combinatorial Optimization 17
2.1. Discrete Semidefinite Programming Model for TSP 17
2.2. Complexity Indices for TSP 21
2.3. Data Clustering 27
3. Defining Graph Invariants by Solving Optimization Problems on Graphs 30
3.1. Star Partitions and Canonical Star Bases 30
3.2. The Maximal Clique Problem and Bounded Multiplicities 32
3.3. Other Highly Informative Graph Invariants 35
References 36

Abstract. It is known that graph invariants, which contain a great quantity of information on graph structure (for example, spectral invariants), are obtained by solving some extremal problems on graphs. Recently, such highly informative graph invariants are applied in solving optimization problems on graphs (e.g., the travelling salesman problem (TSP)). Using these paradigms, several relations, interconnections and interactions between graph theory and mathematical programming are described in this study. A model of TSP based on semidefinite programming and algebraic connectivity of graphs is described. A class of relaxations of this TSP model is defined and some solution techniques based on this class are proposed. Several examples of graph invariants defined by some kind of optimization tasks are also presented. Using several spectrally based graph invariants we treat the graph isomorphism problem.

1. Introduction

In this study we want to elaborate the following two assertions:

Assertion 1. Graph invariants, which contain a great quantity of information on graph structure, are obtained by solving some extremal problems on graphs.

Assertion 2. Highly informative graph invariants are useful in solving optimization problems on graphs.

If these assertions were mathematical statements, they should be proved in mathematical sense. We believe that they are true in an informal sense. Our experience in research shows much evidence of their validity. In this study we shall present several mathematical results which support them. Using these paradigms, several relations, interconnections and interactions between graph theory and mathematical programming are described.

In this introductory section we present some of the basic results from mathematical programming and graph theory which are necessary for the presentation of main ideas in Sections 2 and 3. In 1.1 an important optimization problem, the travelling salesman problem, is introduced. A highly informative graph invariant, the spectrum of a graph, is described in 1.2. Subsection 1.3 is devoted to semidefinite programming, a recently developed optimization technique and an important branch of mathematical programming.

Section 2 elaborates Assertion 2 while Section 3 elaborates Assertion 1.

Key words and phrases. Graph spectra, Algebraic connectivity, Graph isomorphism problem, Semidefinite programming, Travelling salesman problem, Branch-and-bound methods, Complexity indices, Clustering problems.

1.1. Travelling Salesman Problem. There is partly a joke, partly an advice in mathematics saying that if you do not know how to solve a problem you should find the first derivative and make it equal to zero. The point is that a great number of mathematical problems are optimization problems or can be reduced to them.

We shall begin with an exception.

Suppose that a salesman, starting from his home city, is to visit exactly once each city on a given list of cities and then to return home. It is reasonable for him to select the order in which he visits the cities so that the total of the distances travelled in his tour is as small as possible. This problem is called the *travelling salesman problem* (TSP).

TSP is a typical problem of *combinatorial optimization*. There is an extensive literature on and an impressive theory of TSP. The theory includes algorithms and heuristics (with an emphasis on complexity questions) for solving TSP as well as several variations and related problems. There are applications of TSP in operations research and engineering. A nice monograph [51] summaries various aspects of the work that has been done concerning TSP. See also expository articles [49], [50].

Finding the travelling salesman's shortest route to pass n cities in such a way that each city is visited exactly once represents the traditional formulation of TSP. It is assumed that non-negative distances d_{ij} between the cities i, j $(1 \leqslant i < j \leqslant n)$ are given and also that the travelling salesman starts his trip from an arbitrary city. If the travelling salesman does not return to the starting city, then the minimal traversed route is called an *open route* or simply a *path*.

This problem cannot be solved using derivatives. This is because the problem has a discrete character: we have to minimize a function defined on a finite set (the set of permutations of n cities in this case). Such problems belong to the area of combinatorial optimization. There is the obvious *brute force method* to solve such optimization problems: to calculate the value of the objective function for all points in the domain and to select minimum values. However, in the case of TSP and of many other combinatorial optimization problems the execution time of a brute force algorithm on best computers would last for thousands of years for quite modest dimensions of the problem instances (say a couple of dozens of cities in the case of TSP). Since applications require solving large scale problems, many "clever" algorithms and heuristics have been developed and a theory of complexity of algorithms and problems has been established.

One of most popular among algorithms which avoid total search is branch and bound. We shall describe branch and bound technique in a general framework with emphasis on the relevant details concerning the solving of the TSP.

For the sake of simplicity, we restrict ourselves to the following optimization (minimization or maximization) problem on weighted graphs (networks), which is still very general:

Let $\mathcal{A}$ be the set of all subgraphs of a graph G (with weights inherited from G). Let $\mathcal{F} \subseteq \mathcal{A}$ be the set of all subgraphs of G which posses some additional properties. The subgraphs from $\mathcal{F}$ are called *feasible*. We seek in $\mathcal{F}$ the elements with extremal (minimal or maximal) weights.

Let us assume that our optimization problem is a minimization problem. In the case of maximization the procedure would be similar.

In order to solve such a problem by a branch and bound algorithm, let $\mathcal{R}$ ($\mathcal{F} \subseteq \mathcal{R} \subseteq \mathcal{A}$) be a set of subgraphs for which there exists a polynomial time algorithm (say α) for finding the optimal element in $\mathcal{R}$. The set $\mathcal{R}$ corresponds to some relaxed variant of our problem (some feasibility conditions need not hold anymore).

To describe the algorithm (search procedure), we first introduce a *search tree* T as an auxiliary tool. T is a *rooted tree* with the root at a vertex r; all other vertices are the *descendants* of r. If f is any vertex, then its out-neighbors (called *sons* of f, f being their *father*) are denoted by $s_1, \ldots, s_n$. Each vertex, say f, corresponds to some subset $\mathcal{R}(f)$ of $\mathcal{R}$ and to a subproblem of the original problem (usually obtained by including and/or excluding some edges of G from the solution). The root r corresponds to the whole set $\mathcal{R}$. If f is a father and the solution of the relaxation task on the corresponding subproblem is not feasible and its length is smaller than the current lower bound (set at the beginning), then after branching at f by some branching rules (which "destroy" some "unfeasible details" in the solution of the relaxation task), the set $\mathcal{R}(f)$ is split into mutually disjoint subsets $\mathcal{R}(s_1), \ldots, \mathcal{R}(s_n)$ yealding new subproblems and new vertices in the search tree T. By solving the relaxation problem at some tree vertex with the use of the algorithm α, we obtain a lower bound for a feasible solution at this vertex. A global upper bound is provided at the beginning by taking any feasible subgraph (usually found by some quick heuristic). The branch and bound algorithm terminates when all subproblems in the search tree T are exhausted.

The above described general scheme of a branch and bound algorithm can be specified to solve the TSP by taking $\mathcal{F}$ to be the set of all Hamiltonian paths (or cycles or circuits – depending on the variant considered).

For a more detailed treatment of branch and bound algorithms see for example [51, pp. 361–401].

1.2. Graph Spectra and Other Graph Invariants. The *adjacency matrix* of a (multi)(di)graph G, with vertex set $\{1, 2, \ldots, n\}$, is the $n \times n$ matrix $A = (a_{ij})$ whose (i, j)-entry a_{ij} is equal to the number of edges, or arcs, originating at the vertex i and terminating at the vertex j. Two vertices of G are said to be *adjacent* if they are connected by an edge or arc. Unless we indicate otherwise we shall assume that G is an undirected graph without loops or multiple edges. The *degree* of a vertex is the number of vertices adjacent to that vertex.

The characteristic polynomial $\det(\lambda I - A)$ of the adjacency matrix A of G is called the *characteristic polynomial* of G and denoted by $P_G(\lambda)$. The eigenvalues of A (i.e., the zeros of $\det(\lambda I - A)$) and the spectrum of A (which consists of the n eigenvalues) are also called the *eigenvalues* and the *spectrum* of G, respectively. The spectrum of G is denoted by *spec* G. These notions are independent of vertex labelling because a reordering of vertices results in a similar adjacency matrix. The eigenvalues of G are usually denoted by $\lambda_1, \ldots, \lambda_n$; they are real because A is symmetric. Unless we indicate otherwise, we shall assume that $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$

and use the notation $\lambda_i = \lambda_i(G)$ for $i = 1, 2, \ldots, n$. Clearly, isomorphic graphs have the same spectrum.

The eigenvalues of A are the numbers λ satisfying $Ax = \lambda x$ for some non-zero vector $x \in \mathbb{R}^n$. Each such vector x is called an *eigenvector* of the matrix A (or of the labelled graph G) belonging to the eigenvalue λ. The relation $Ax = \lambda x$ can be interpreted in the following way: if $x = (x_1, x_2, \ldots, x_n)^T$, then $\lambda x_u = \sum_{v \sim u} x_v$ where the summation is over all neighbours v of the vertex u. If λ is an eigenvalue of A then the set $\{x \in \mathbb{R}^n : Ax = \lambda x\}$ is a subspace of $\mathbb{R}^n$, called the *eigenspace* of λ and denoted by $\mathcal{E}(\lambda)$ or $\mathcal{E}_A(\lambda)$. Such eigenspaces are called eigenspaces of G. Of course, relabelling of the vertices in G will result in a permutation of coordinates in eigenvectors (and eigenspaces).

The largest eigenvalue λ_1 of a graph G is called the emphindex of G; since adjacency matrices are non-negative there is a corresponding eigenvector whose entries are all non-negative.

Next we present certain notation, definitions and results from graph theory.

As usual, K_n, C_n and P_n denote respectively the *complete graph*, the *cycle* and the *path* on n vertices.

mG denotes the union of m disjoint copies of G. We write $V(G)$ for the vertex set of G, and $E(G)$ for the edge set of G.

If uv is an edge of G we write $G - uv$ for the graph obtained from G by deleting uv. For $v \in V(G)$, $G - v$ denotes the graph obtained from G by deleting the vertex v and all edges incident with v. More generally, for $U \subseteq V(G)$, $G - U$ is the subgraph of G induced by $V(G) \setminus U$.

A function defined on a family $\mathcal{G}$ of graphs is called a *graph invariant* for graphs in $\mathcal{G}$ if it is the same for isomorphic graphs in $\mathcal{G}$. Usually, graph invariants are numbers (integers, reals, etc.) but can be more complex objects (families of numbers, vectors, matrices, etc.).

Highly informative graph invariants from the title have not been defined precisely; we use this term informally. We shall say that a graph invariant is *highly informative* if it can be obtained quickly (possibly by a polynomial time algorithm) and if it contains a lot of information on the graph structure. It would be desirable that the invariant fully determines the graph (up to isomorphism as it is usually said). Such invariants would be obviously useful in solving the *graph isomorphism problem*, i.e., the problem of deciding whether or not two given graphs are isomorphic.

Let us consider some examples of graph invariants

1. *Vertex degrees.* The family of vertex degrees can be quickly calculated. However, the degree of a vertex is a kind of local invariant; it does not depend on the structure of the whole graph. Only neighbors of the vertex in question contribute to the value of its degree. It is not surprising that the family of vertex degrees does not say much on the graph structure, i.e. usually there are several graphs having a given family of vertex degrees. For example, a graph on 8 vertices having all vertex degrees equal to 2 can be one of the following three graphs: C_8, $C_5 \cup C_3$, $C_4 \cup C_4$.

2. *Spectrum.* Family of graph eigenvalues is obtained by considering extremal values of the Rayleigh quotient of the adjacency matrix. Eigenvalues depend in general case on all details on graph structure. Therefore more can be said on graph structure in the case that we know graph eigenvalues than in the case of knowing vertex degrees. Let us analyze the situation with graphs in which the vertex degrees are equal to 2. Such graphs are called *regular graphs of degree* 2.

Regular graphs of degree 2 are unions of cycles. One can verify by direct calculation that eigenvalues of the cycle C_n are real parts of the n-th roots of 2^n, i.e.,

$$spec\ C_n = \{Re\ \sqrt[n]{2^n}\} = \left\{2\cos\frac{2\pi}{n}j \;\middle|\; j = 0, 1, \ldots, n-1\right\}$$

The largest eigenvalue is $\lambda_1 = 2$ ($j = 0$) and the next one is two-fold: $\lambda_2 = \lambda_3 = 2\cos\frac{2\pi}{n}$ (for $j = 1$ and $j = n-1$). Suppose now that $G = \bigcup_{i=1}^{k} C_{n_i}$. Then

$$spec\ G = \bigcup_{i=1}^{k} \left\{2\cos\frac{2\pi}{n_i}j \;\middle|\; j = 0, 1, \ldots, n_i - 1\right\}$$

Given *spec* G, we can first establish that G is regular (by Theorem 3.22 of [31]) of degree 2. This is already information contained in the family of vertex degrees. But here we have more. Finding the second largest eigenvalue in modulus in *spec* G, we can determine the size n_i of the largest cycle in G. Gradually, by analyzing the whole spectrum we can determine the sizes of all cycles of G, i.e., determine G up to isomorphism.

In this way we have proved the following theorem (see [12] or [31, p. 167]).

Theorem 1.1. *A regular graph of degree* 2 *is characterized by its spectrum.*

The reader might think that unions of cycles are not so interesting graphs to justify the space devoted to their spectral characterizations. However, the importance of this theorem will be shown in Subsection 2.1.

It seems that graph theoretical invariants, which contain a lot of information about the graph structure and thus are useful for the graph isomorphism problem, are obtained by solving some kind of optimization problem. Eigenvalues are also obtained in this way (as extrema of the Rayleigh quotient). The same holds for angles of a graph [17]. See 3.3 for other examples.

3. *A binary number.* A graph G can be characterized by the largest (or least) binary number obtained by concatenation of rows (or rows of the upper triangle) of adjacency matrices of G. The ordering of vertices which yields the characterizing binary number can be considered as a canonical vertex ordering. One can consider several variations of this idea but it turns out that the known algorithms for finding the graph characterizing quantity are exponential (cf. [62], [4]). Here a high price has been paid. We have an invariant which tells everything about the graph but it is time consuming to determine it. (However, this does not mean that under certain circumstances the extremal binary number has not been successfully used in recognizing graphs).

From the point of view of practical computation it is not very important to decide whether the graph isomorphism problem is NP-complete or belongs to P. Experience has shown that any reasonable algorithm for graph isomorphism testing performs well in average. However, the problem has great theoretical significance. Leaving aside the implications in the theory of complexity of algorithms and problems, one can say that the understanding of the kind of difficulties arising in the graph isomorphism problem enables the understanding of difficulties that appear in treating graph theory problems in general.

After having got equanted with these three examples we might be inclined to believe that spectral type graph invariants represent a good compromise between different reqirements on graph invariants. Therefore we describe another variant in defining graph eigenvalues.

Let $G = (V, E)$ be an undirected simple graph, where $V = \{1, \ldots, n\}$ is the set of vertices and E is the set of edges. The *Laplacian* $L(G)$ of graph G is a symmetric matrix defined as $L(G) = D(G) - A(G)$, where $D(G)$ is the diagonal matrix with vertex degrees on the diagonal and $A(G)$ is the adjacency matrix of G.

The matrix $L(G)$ is positive semidefinite. If $\mu_1 \leqslant \cdots \leqslant \mu_n$ are eigenvalues of $L(G)$, then $\mu_1 = 0$ with the corresponding eigenvector $e = (1, \ldots, 1)$. All other eigenvalues have eigenvectors which belong to the set

$$S = \left\{ x = (x_1, \ldots, x_n) \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} x_i = 0, \sum_{i=1}^{n} x_i^2 = 1 \right\}$$

According to Fiedler, the second smallest eigenvalue μ_2 of $L(G)$, is called the *algebraic connectivity* of G and denoted by $a(G)$. In [37] the following results are proved:

Theorem 1.2. *The algebraic connectivity $a(G)$ has the properties:*

(*i*) $a(G) = \min_{x \in S} x^T L(G) x$

(*ii*) $a(G) \geqslant 0$, $a(G) > 0$ *if and only if G is connected.*

Fiedler shows that the notion of the Laplacian and the algebraic connectivity can be generalized to graphs with positively weighted edges.

A C-edge-weighted graph $G_C = (V, E, C)$ is defined by a graph $G = (V, E)$ and a symmetric nonnegative weight matrix C such that $c_{ij} > 0$ if and only if $\{i, j\} \in E$. Now the Laplacian $L(G_C)$ is defined as $L(G_C) = \mathrm{diag}(r_1, \ldots r_n) - C$, where r_i is the sum of the i-th row of C. The Laplacian $L(G_C)$ has similar characteristics as $L(G)$. Namely it is symmetric, positive semidefinite with the smallest eigenvalue $\mu_1 = 0$ and the corresponding eigenvector e. As before, the algebraic connectivity $a(G_C)$ is the second smallest eigenvalue of $L(G_C)$, which enjoys similar properties to those in Theorem 1.2.

Theorem 1.3. (M. Fiedler [37]) *The generalized algebraic connectivity $a(G_C)$ has the following properties:*

(*i*) $a(G_C) = \min_{x \in S} x^T L(G_C) x$

(*ii*) $a(G_C) \geqslant 0$, $a(G_C) > 0$ *if and only if G_C is connected.*

1.3. Semidefinite Programming. Semidefinite programming (SDP) has been one of the most active research areas in mathematical programming during the last decade. It is related to minimization of a linear function on the set of positive semidefinite matrices subject to linear constraints.

Recall that a symmetric matrix is called *positive semidefinite* (*positive definite*) if its eigenvalues are nonnegative (positive).

In order to define a semidefinite program, we need to introduce the appropriate notation. Let $S^{n\times n}$ denote the set of symmetric $n\times n$ matrices and let $S_+^{n\times n}$ denote the set of positive semidefinite $n\times n$ matrices, Then $S_+^{n\times n}$ is a closed convex cone in $\mathbb{R}^{n\times n}$ of dimension $n(n-1)/2$. We write $X \geqslant 0$ ($X > 0$) to denote that X is a symmetric positive semidefinite (positive definite) matrix, and we write $X \geqslant Y$ to denote that $X - Y \geqslant 0$. For $A, B \in \mathbb{R}^{n\times n}$ the Frobeinus product is defined by

$$A \circ B = tr(A^T B) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{ij}$$

If $A, B \in S^{n\times n}$ it follows that $A \circ B = tr(AB)$.

If $A, B \in S_+^{n\times n}$ it can be proved that $A \circ B \geqslant 0$ and that $A \circ B = 0$ implies $AB = 0$ (see [65]).

Now a semidefinite program (SDP) can be formulated as:

$$\begin{array}{lll} & \textit{minimize} & C \circ X \\ (1) & \textit{subject to} & A_i \circ X = b_i,\ i = 1, \ldots, m \\ & & X \geqslant 0 \end{array}$$

where C, $A_1, \ldots, A_m \in S^{n\times n}$, $b = (b_1, \ldots, b_m) \in \mathbb{R}^m$ are given parameters and the unknown $n \times n$ matrix X is symmetric positive semidefinite. In the sequel P and P° will denote the feasible set of problem (1) and its relative interior, i.e.,

$$P = \{X \in \mathbb{R}^{n\times n} \mid A_i \circ X = b_i,\ i = 1, \ldots, m,\ X \geqslant 0\}$$
$$P^\circ = \{X \in \mathbb{R}^{n\times n} \mid A_i \circ X = b_i,\ i = 1, \ldots, m,\ X > 0\}$$

Without loss of generality we may assume that matrices $A_1, \ldots, A_m$ are linearly independent. It is easy to see that then (1) can be written in the form

$$\begin{array}{lll} & \textit{minimize} & c_0 + c^T z \\ (2) & \textit{subject to} & F_0 + \sum_{i=1}^{p} z_i F_i \geqslant 0 \end{array}$$

where $z \in \mathbb{R}^p$ is the unknown vector, $p = n(n+1)/2 - m$, and $F_i \in S^{n\times n}$, $i = 0, \ldots, p$, $c_0 \in R$, $c \in \mathbb{R}^p$ are the corresponding parameters. Indeed, problem (1) has n^2 scalar variables and $m + n(n-1)/2$ linear equations (m given explicitly and $n(n-1)/2$ following from the fact that X is symmetric). Hence there are $n^2 - m - n(n-1)/2 = n(n+1)/2 - m = p$ free variables which uniquely determine the remaining ones, i.e., there exist (symmetric) matrices F_0, $F_1, \ldots, F_p$ such that

$$\{X \in S^{n\times n} \mid A_i \circ X = b_i,\ i = 1, \ldots, m\} = \{X = F_0 + z_1 F_1 + \cdots + z_p F_p \mid z \in \mathbb{R}^p\}$$

This implies that the feasible sets of (1) and (2) are equal. Moreover,

$$C \circ X = C \circ (F_0 + z_1 F_1 + \cdots + z_p F_p) = C \circ F_0 + z_1 C \circ F_1 + \cdots + z_p C \circ F_p$$

and we can take $c_0 = C \circ F_0$, $c = (C \circ F_1, \ldots, C \circ F_p)$.

Theoretical properties of the SDP problem have been studied in sixties, seventies and early eighties by several authors, e.g. Bellman, Fan [7], Craven, Mond [11], Fletcher [38], Rockafellar [63] Wolkowicz [67], etc. We shall state here only the main results. The dual problem associated to (1) is the following SDP problem of the type (2):

$$\begin{aligned} &\textit{maximize} && b^T y \\ &\textit{subject to} && \sum_{i=1}^{m} y_i A_i \leqslant C, \end{aligned}$$

which can be equivalently reformulated as:

$$\begin{aligned} &\textit{maximize} && b^T y \\ &\textit{subject to} && \sum_{i=1}^{m} y_i A_i + Z = C \\ & && Z \geqslant 0 \end{aligned} \tag{3}$$

The feasible set of (3) and its relative interior will be denoted by D and D°, respectively, i.e.

$$D = \left\{ (Z, y) \in \mathbb{R}^{n \times n} \times \mathbb{R}^m \;\middle|\; \sum_{i=1}^{m} y_i A_i + Z = C,\ Z \geqslant 0 \right\}$$

$$D^\circ = \left\{ (Z, y) \in \mathbb{R}^{n \times n} \times \mathbb{R}^m \;\middle|\; \sum_{i=1}^{m} y_i A_i + Z = C,\ Z > 0 \right\}$$

It is easy to prove the following week duality theorem.

Theorem 1.4. *If $X \in P$, $(Z, y) \in D$, then $C \circ X \geqslant b^T y$.*

Proof. We have

$$C \circ X = \sum_{i=1}^{m} y_i A_i \circ X + Z \circ X = \sum_{i=1}^{m} y_i b_i + Z \circ X = b^T y + Z \circ X \tag{4}$$

As Z, $X \in S_+^{n \times n}$ it follows that $Z \circ X \geqslant 0$ and (4) implies $C \circ X \geqslant b^T y$. □

Let p^* and d^* be the optimal values of primal (1) and dual (3), i.e.,

$$p^* = \inf_{X \in P} C \circ X, \qquad d^* = \sup_{(Z,y) \in D} b^T y$$

Theorem 1.4 implies $p^* \geqslant d^*$. Let P^* and D^* be the corresponding sets of optimal solutions, i.e.,

$$P^* = \{X \in P \mid C \circ X = p^*\}, \qquad D^* = \{(Z, y) \mid b^T y = d^*\}.$$

It is easy to construct examples demonstrating that the sets P^* (D^*) can be empty even if p^* (d^*) is finite, which is not the case in linear programming. The next

theorem gives conditions which guarantee that P^* and D^* are nonempty and that the duality gap $p^* - d^*$ is equal to zero.

Theorem 1.5. (*i*) *Suppose that one of the following conditions hold:* 1° $P^\circ \neq \emptyset$, 2° $D^\circ \neq \emptyset$. *Then* $p^* = d^*$.

(*ii*) *Suppose that* 1° *and* 2° *hold. Then* $P^* \neq \emptyset$, $D^* \neq \emptyset$.

The proof is an application of the duality theory from convex analysis (see e.g., [59], [63]).

If both conditions 1° and 2° hold it is easy to see that the set $P^* \times D^*$ is equal to the set of solutions to the system

$$\begin{array}{ll} \text{(a)} & X\,Z = 0 \\ \text{(b)} & \displaystyle\sum_{i=1}^{m} y_i\ A_i + Z - C = 0 \\ \text{(c)} & A_i \circ X - b_i = 0,\ i = 1, \ldots, m \\ \text{(d)} & X \geqslant 0,\ Z \geqslant 0 \end{array} \tag{5}$$

Indeed, if X and (Z, y) are optimal solutions of problems (1) and (3) their feasibility implies conditions (5b)–(5d). Moreover, $C \circ X = p^* = d^* = b^T y$. Since by (4) $Z \circ X = C \circ X - b^T y$ it follows that $Z \circ X = 0$, which implies $X\,Z = 0$, i.e., (5a) holds.

Let now $(X, Z, y) \in \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n} \times \mathbb{R}^m$ be a solution of (5a)–(5d). Then X and (Z, y) are feasible solutions of (1) and (3) and hence, by Theorem 1, $C \circ X \geqslant p^* \geqslant d^* \geqslant b^T y$. As $X\,Z = 0$ implies $Z \circ X = 0$ from (4) it follows $C \circ X = b^T y$, i.e., $C \circ X = p^*,\ b^T y = d^*$.

A strong impulse to further development of semidefinite programming was given by Nesterov and Nemirovski in a series of papers [55, 56, 57, 58, 59] written between 1988 and 1991 and by Alizadeh [2], who have shown independently that interior point methods for linear programming can be directly extended to SDP. For example, the parametrized logarithmic barrier problem for linear programming extends to SDP as:

$$\begin{array}{ll} \textit{minimize} & C \circ X - t \ln(\det X) \\ \textit{subject to} & A_i \circ X = b_i,\ i = 1, \ldots, m \\ & X > 0 \end{array} \tag{6}$$

where $\ln(\det X)$ replaces the logarithmic barrier function $\sum_{j=1}^{n} \ln x_j$. The optimality conditions for this problem can be written as

$$\begin{array}{ll} \text{(a)} & X\,Z - tI = 0 \\ \text{(b)} & \displaystyle\sum_{i=1}^{m} y_i\ A_i + Z - C = 0 \\ \text{(c)} & A_i \circ X - b_i = 0,\ i = 1, \ldots, m \\ \text{(d)} & X > 0,\ Z > 0 \end{array} \tag{7}$$

which in fact is a parametrization of optimality conditions (5a)–(5d). Under the assumptions 1° and 2° from Theorem 1.5 it can be shown that for each $t > 0$ system (7a)–(7d) has the unique solution (X_t, Z_t, y_t). Moreover, $\lim_{t\to 0+}(X_t, Z_t, y_t) = (X^*, Z^*, y^*)$, where X^* solves (1) and (Z^*, y^*) solves (3) (for a proof see [46]).

The key idea of interior-point methods for SDP is to use Newton method in order to get approximate solutions of the parametrized system (7a)–(7d). A typical algorithm can be described as follows:

Algorithm:
Input: $X_0 \in P^\circ$, $(Z_0, y_0) \in D^\circ$, $\varepsilon > 0$
Initialization: Set $k = 0$, $t_0 = X_0 \circ Z_0/n$
Repeat until $t_k < \varepsilon$ do
(1) Set in (7a)–(7d) $t = t_k$
(2) Compute the Newton direction $(\triangle X_k, \triangle Z_k, \triangle y_k)$ at (X_k, Z_k, y_k).
(3) Choose $\alpha_k > 0$ such that
$(X_{k+1}, Z_{k+1}, y_{k+1}) = (X_k, Z_k, y_k) + \alpha_k(\triangle X_k, \triangle Z_k, \triangle y_k) \in P^\circ \times D^\circ$
(4) Set $t_{k+1} = X_{k+1} \circ Z_{k+1}/n$, $k \leftarrow k + 1$
End.

It should be noted that (7a) can be represented in many different ways, including for example $(X\,Z + Z\,X)/2 - t\,I = 0$, resulting in many different nonequivalent Newton directions, and hence different SDP methods. In terms of theoretical performance, the best SDP methods are guaranteed to reduce duality gap of the iterates by a fixed proportion in $O(\sqrt{n})$ iterations. This is identical to the complexity result for linear programming with n variables, even though the number of scalar variables in SDP is much larger (there are $n(n+1)/2$ entries in the symmetric matrix X). More precisely, the algorithm stops in $O(\sqrt{n}\log\frac{X_0\circ Z_0}{n\epsilon})$ iterations, while the complexity of a single iteration of the algorithm is typically $O(\max\{m^2n^2, mn^3, m^3\})$. This gives the overall complexity bound $O(\max\{m^2n^{2.5}, mn^{3.5}, m^3n^{o.5}\})$.

There are many active research areas in semidefinite programming varying from development of different interior point algorithms and investigating their properties to writing efficient SDP codes capable of handling large sparse SDP problems. Special attention is payed to applications of SDP, which are very wide. The types of constraints that can be modelled in the SDP framework include linear inequalities, convex quadratic inequalities, lower bounds on matrix norms, lower bounds on determinants of symmetric positive semidefinite matrices, lower bounds on the geometric mean of a nonnegative vector, etc. Using these and other constructions the following problems can be stated as SDP problems: optimizing a convex quadratic form subject to convex quadratic inequalities, minimizing the volume of an ellipsoid that covers a given set of points and ellipsoids, maximizing the volume of an ellipsoid that is contained in a given polytope, a variety of maximum eigenvalue and minimum eigenvalue problems, etc. In particular, there is a growing interest in applications of SDP in combinatorial optimization where it is used in order to get satisfactory lower bounds on the optimal objective function value. Some examples are SDP relaxations for the max-cut problem, graph coloring problem and the travelling salesman problem. The next section gives a detailed description of SDP

approach to the travelling salesman problem. A comprehensive survey of theory, algorithms and applications of semidefinite programming can be found in a recently published monograph [68].

2. Using Spectral Invariants in Problems of Combinatorial Optimization

In this section we elaborate Assertion 2 by describing the use of algebraic connectivity of a graph in solving TSP and in some clustering problems. In Subsection 2.1 we describe a model of TSP based on semidefinite programming and algebraic connectivity of graphs. Another way of using graph spectra in treating TSP is given in Subsection 2.2, where we introduce complexity indices for TSP. Subsection 2.3 describes some problems of clustering binary vectors and provides another example of using highly informative graph invariants in solving optimization problems.

2.1. Discrete Semidefinite Programming Model for TSP. Let $G = (V, E)$ be a complete undirected graph, where, as before, $V = \{1, \dots, n\}$ is the set of vertices and E is the set of edges. To each edge $\{i, j\} \in E$ a distance (cost) d_{ij} is associated such that the distance matrix $D = (d_{ij})_{n\times n}$ is symmetric and $d_{ii} = 0$, $i = 1, \dots, n$. Now the symmetric travelling salesman problem (TSP) can be formulated as follows: *find a Hamiltonian circuit of G with minimal cost.*

Algebraic connectivity of a Hamiltonian circuit is well known in the theory of graph spectra (see e.g. [31]). Since the graph is regular of degree 2, we have $L = 2I - A$. Hence, the Laplacian of a circuit with n vertices has the spectrum

$$2 - 2\cos(2\pi j/n),\ j = 1, \dots, n$$

and the second smallest eigenvalue is obtained for $j = 1$ and $j = n - 1$, i.e., $\mu_2 = \mu_3 = 2 - 2\cos(2\pi/n)$. This value will be denoted by h_n, i.e., $h_n = 2 - 2\cos(2\pi/n)$.

Now, Theorem 1.1 of Section 1.2 will be transformed into a form which is very useful in solving TSP. The next theorem, which gives a basis for the discrete semidefinite programming model of TSP, has been proved in [24] as a consequence of a more general result. For the sake of completeness we supply here a self-contained proof following [25].

Theorem 2.6. *Let H be a spanning subgraph of G such that $d(i) = 2$, $i = 1, \dots, n$, where $d(i)$ is the degree of vertex i with respect to H, and let $L(H) = (l_{ij})_{n\times n}$ be the corresponding Laplacian. Let α and β be real parameters such that $\alpha > h_n/n$, $0 < \beta \leqslant h_n$. Then H is a Hamiltonian circuit if and only if the matrix $X = L(H) + \alpha J - \beta I$ is positive semidefinite, where J is the $n \times n$ matrix with all entries equal to one and I is the unit matrix of order n.*

Proof. Let $0 = \mu_1 \leqslant \mu_2 \leqslant \dots \leqslant \mu_n$ be the eigenvalues of $L(H)$ and let $x^1 = e$ and $x^i \in S, i = 2, \dots, n$, be the corresponding eigenvectors which form a basis for $\mathbb{R}^n$. It is easy to check that J has two eigenvalues: 0, with multiplicity $n - 1$ and the corresponding eigenvectors $x^2, \dots, x^n$, and n with e as its eigenvector. Therefore

$$Xe = (L + \alpha J - \beta I)e = (\alpha n - \beta)e$$
$$Xx^i = (L + \alpha J - \beta I)x^i = (\mu_i - \beta)x^i,\ i = 2, \dots, n$$

which means that $\alpha n - \beta$ and $\mu_i - \beta,\ i = 2, \ldots, n$ are eigenvalues of X with eigenvectors $e, x^2, \ldots, x^n$, respectively.

The conditions of Theorem 2.1 garantee that H is a 2-matching, i.e., it is either a Hamiltonian circuit or a collection of at least two disjoint subcircuits. In the first case $\mu_2 = h_n$, while in the second, according to Theorem 1.2, $\mu_2 = 0$. As $\alpha > h_n/n$ in both cases it follows that $\alpha n - \beta > \mu_2 - \beta$, i.e., the smallest eigenvalue of X is equal to $\mu_2 - \beta$.

Suppose that H is a Hamiltonian circuit. Then $\beta \leqslant h_n$ implies $\mu_2 - \beta = h_n - \beta \geqslant 0$, i.e., matrix X is positive semidefinite. Suppose now that X is positive semidefinite. Then $\mu_2 - \beta \geqslant 0$ and $\beta > 0$ imply $\mu_2 = a(H) > 0$ and by Theorem 1.2 it follows that H is a connected 2-matching, i.e., a Hamiltonian circuit. □

It follows from Theorem 2.1 that a spanning subgraph H of G is a Hamiltonian circuit if and only if its Laplacian $L(H)$ satisfies the following conditions:

$$l_{ii} = 2,\ i = 1, \ldots, n \tag{8}$$

$$X = L(H) + \alpha J - \beta I \text{ is positive semidefinite},\ \alpha > h_n/n,\ 0 < \beta \leqslant h_n \tag{9}$$

Starting from (8) and (9) the following discrete semidefinite programming model of TSP can be defined

$$\textit{minimize } F(X) = \sum_{i=1}^{n}\sum_{j=1}^{n}\Big(-\frac{1}{2}d_{ij}\Big)x_{ij} + \frac{\alpha}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \tag{10}$$

subject to

$$x_{ii} = 2 + \alpha - \beta,\ i = 1, \ldots, n \tag{11}$$

$$\sum_{j=1}^{n} x_{ij} = n\alpha - \beta,\ i = 1, \ldots, n \tag{12}$$

$$x_{ij} \in \{\alpha - 1, \alpha\},\ i, j = 1, \ldots, n, i < j \tag{13}$$

$$X \geqslant 0, \tag{14}$$

where $X \geqslant 0$ denotes that the matrix $X = (x_{ij})_{n\times n}$ is symmetric and positive semidefinite and α and β are chosen according to Theorem 2.1. Matrix $L = X + \beta I - \alpha J$ represents the Laplacian of a Hamiltonian circuit if and only if X satisfies (11)–(14). Indeed, constraints (11)–(13) provide that L has the form of a Laplacian with diagonal entries equal to 2, while condition (14) guarantees that L corresponds to a Hamiltonian circuit. Therefore, if X^* is an optimal solution of problem (10)–(14), then $L^* = X^* + \beta I - \alpha J$ is the Laplacian of an optimal Hamiltonian circuit of G with the objective function value $\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}(-\frac{1}{2}d_{ij})l^*_{ij} = F(X^*)$.

The well-known integer programming formulation of TSP reads:

$$\textit{minimize} \quad \sum_{i\in V}\sum_{j>i} d_{ij}x_{ij} \tag{15}$$

subject to

$$\sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} = 2, \qquad i \in V \ ; \tag{16}$$

$$\sum_{i\in S} \sum_{\substack{j\in S \\ j>i}} x_{ij} \leqslant |S| - 1, \quad \text{for all } S \subset V,\ S \neq \emptyset; \tag{17}$$

$$x_{ij} = 0 \text{ or } 1 \qquad i,j \in V,\ j > i. \tag{18}$$

The subtour elimination inequalities (17) can also be written as

$$\sum_{i\in S} \sum_{\substack{j\in V-S \\ j>i}} x_{ij} + \sum_{i\in V-S} \sum_{\substack{j\in S \\ j>i}} x_{ij} \geqslant 2, \qquad \text{for all } S \subset V,\ S \neq \emptyset.$$

Each of n constraints in group (16) requires exactly two edges to be incident to every vertex and constraints of type (17) are subtour elimination constraints excluding the subtours with less than n vertices.

It is important to note that in our discrete semidefinite programming model of TSP (10)–(14), the single condition (14) replaces all subtour elimination constraints in the standard integer programming model!

A natural semidefinite relaxation of the travelling salesman problem is obtained when discrete conditions (13) are replaced by inequality conditions:

$$\textit{minimize } F(X) \tag{19}$$

subject to

$$x_{ii} = 2 + \alpha - \beta,\ i = 1, \ldots, n \tag{20}$$

$$\sum_{j=1}^{n} x_{ij} = n\alpha - \beta,\ i = 1, \ldots, n \tag{21}$$

$$\alpha - 1 \leqslant x_{ij} \leqslant \alpha,\ i,j = 1, \ldots, n,\ i < j \tag{22}$$

$$X \geqslant 0 \tag{23}$$

It is easy to see that the relaxation (19)–(23) can be expressed in the standard form of an SDP problem. Indeed, constraint (20) can be represented as $A_i \circ X = 2 + \alpha - \beta$, where $\circ$ is the Frobenius product and A_i is a symmetric $n \times n$ matrix with 1 at the position (i,i) and all other entries equal to 0. Similarly, condition (21) is equivalent to $B_i \circ X = 2(n\alpha - \beta)$, where B_i has 2 at the position (i,i) while all the remaining elements of the i-th row and the i-th column are equal to 1, and all the other entries are zero. Finally, condition (22) can be expressed as $2(\alpha - 1) \leqslant C_{ij} \circ X \leqslant 2\alpha$, where C_{ij} has 1 at the positions (i,j) and (j,i) and zero otherwise. Since SDP problem (19)–(23) depends on parameters α and β it represents a class of semidefinite relaxations of TSP. In the sequel, members of this class will be referred to as SDP relaxations.

Let us denote by P and P° the feasible set of problem (19)–(23) and its relative interior. For each $X \in P$ the corresponding Laplacian $L = X + \beta I - \alpha J$ can be interpreted as the Laplacian of the weighted graph $G_L = (V, E_L, C_L)$, where

$E_L = \{\{i,j\} \in E \mid l_{ij} < 0\}$ and $C_L = 2I - L$. If α and β satisfy the conditions of Theorem 2.1 then, using similar arguments as in the proof of Theorem 2.1, it can be shown that $X \geqslant 0$ is equivalent to $a(G_L) \geqslant \beta$ (see also [24]). Hence, by Theorem 1.3 graph G_L is connected. It immediately follows that 2-matchings with disjoint subcircuits cannot correspond to any X in P.

It is easy to see that $P^\circ \neq \emptyset$. Indeed, if e.g. $\hat{L} = \Big(2 + \dfrac{2}{n-1}\Big)I - \dfrac{2}{n-1}J$, then $\hat{X} = \hat{L} + \alpha J - \beta I = \Big(2 + \dfrac{2}{n-1} - \beta\Big)I + \Big(\alpha - \dfrac{2}{n-1}\Big)J$ has the eigenvalues $2 + \dfrac{2}{n-1} - \beta$ with the multiplicity $n-1$ and $n\alpha - \beta$ with the multiplicity 1. Since $n\alpha - \beta > 0$ and $2 + \dfrac{2}{n-1} - \beta \geqslant 2 + \dfrac{2}{n-1} - h_n > 0$ for $n \geqslant 4$, it follows that $\hat{X} \in P^\circ$, $n \geqslant 4$.

For $\beta < h_n$ matrices X which correspond to Laplacians of Hamiltonian circuits are in P°, while for $\beta = h_n$ these matrices belong to $P \setminus P^\circ$. It is clear that the best relaxation is obtained for $\beta = h_n$. Concerning the parameter α, it is always sufficient to choose $\alpha = 1$.

The semidefinite relaxation (19)–(23) is substantially different from the existing TSP relaxations. It should be pointed out that it cannot be theoretically compared neither with 2-matching nor with 1-tree. Indeed, if we consider TSP model (10)–(14) it is easy to see that X which corresponds to the Laplacian of a 2-matching satisfies (11)–(13) but need not satisfy (14). In the case of 1-tree, the condition (11) is relaxed, while (12), (13) and (14) hold (see [24]). Preliminary numerical experiments on randomly generated problems with $10 \leqslant n \leqslant 20$ which are reported in [24], indicate that SDP relaxation gives considerably better lower bounds than both 1-tree and 2-matching.

We have implemented two branch and bound algorithms with the SDP relaxation (with $\alpha = 1$, $\beta = h_n$) and one with the 1-tree relaxation. The last one was implemented to check the correctness of the results. All algorithms are based on the general branch and bound scheme as described in [51]. We used a FORTRAN implementation of the branch and bound shell from the package TSP-SOLVER [21], [29]. An initial upper bound was obtained in all cases by the 3-optimal heuristic. The depth first search was used to select the next subproblem.

The two branch and bound algorithms differ only in their branching rules (the first one defined by Vollgnant and Jonker, see [25]):

Algorithm 1. At the first vertex of degree greater than 2 in the weighted graph representing the SDP solution an edge is excluded in each son;

Algorithm 2. The first non-integer entry of the SDP solution matrix is replaced in the sons by 0 and 1 respectively.

Inequality conditions (22) were handled adding $n^2 - n$ slack variables each represented by a 1×1 block as accepted by the software.

For solving the SDP relaxation tasks we used a modification of CSDP 2.3 software package developed by Borchers [8, 9] in C language. The package is based on a predictor–corrector variant of the interior point algorithm presented by Helmberg,

Rendl, Vanderbei, Wolkowicz [43]. The experiments were performed on an Alpha 800 5/400 computer. Preliminary computational results were reported in [25]. A part of the results is presented in the next subsection. Further numerical evidence with larger TSP instances is given in [47], [48].

2.2. Complexity Indices for TSP. In this section we will illustrate how some invariants related to the solution of SDP relaxation (19)–(23) could be implemented as complexity indices in an adaptive solution approach for TSP. Such an approach, introduced in [29], is based on the following principles:

For a given branch and bound (B&B) algorithm and a given maximal number of relaxation tasks R_S which are allowed to be solved within the algorithm, TSP instances are classified into two classes: *hard* and *easy* instances. The hard class contains TSP instances for which solving more than R_S relaxation tasks is required to reach an optimal solution, while the easy class consists of the remaining instances.

Since hard instances usually require a lot of computing time, there is some interest to recognize such instances before the algorithm starts. If a concrete instance is recognized to be hard, instead of finding an exact solution, a suboptimal solution could be found by an efficient heuristic.

The recognition of hard and easy instances is realized using the notion of complexity indices.

Given an instance of the TSP, *the instance complexity* of this instance for the given B&B algorithm can be defined as *the number of relaxation tasks which need to be solved within the applied algorithm to reach an optimal solution.*

Any number assigned to an instance which contains some information on the instance complexity (with respect to a given B&B algorithm) will be called a *complexity index.*

Usually, complexity index is a numerical graph invariant of a (weighted) graph related to the solution of the relaxation task for the instance considered. In the context of this study, special attention will be paid to highly informative graph invariants, since we might expect that just these will serve as good complexity indices.

Here we assume that there exists an efficient (polynomial) algorithm for determining the index under the consideration.

Since an instance complexity of the TSP for a given B&B algorithm is related to the number of relaxation tasks, it is reasonable to determine the value of a complexity index on the basis of solved relaxation tasks within the algorithm. This is based on the expectation that the branch and bound algorithm will run for longer, the more relaxation solution(s) are distanced from an optimal solution, and that this information could be extracted from one or several relaxation solution(s). Each type of a possible relaxation used in some variant of B&B algorithm offers a variety of complexity indices. In this way complexity indices depend upon a B&B algorithm and so special complexity indices for each variant of a B&B algorithm can be introduced.

There are no theoretical results described in the literature which would indicate the existence of efficient complexity indices for a particular instance of NP-hard

problems, in spite of the fact that this would be of obvious practical importance. As a mater of fact, we do not see how a theory of complexity indices for instances of NP-hard problems could be set up based on known results.

The idea of complexity indices has been initiated in [54] in relation to the TSP. The indices offered have been intuitively justified and their validity supported by some experimental results. The largest eigenvalue of the adjacency matrix of a minimal spanning tree has been introduced in [15]] as a complexity index for the travelling salesman problem and its validity supported by some results from the theory of graph spectra [31].

As a selection criterion for the most informative index, the measure of statistical dependence of the value of the complexity index and the number of the solved relaxation tasks is used. This measure ought to reflect as much as possible the extent and the type of the dependence. It is also pointed out in [15] that the efficiency of complexity indices is related to statistical distribution of the set of instances which are intended to be solved.

Let $\mathcal{N}$ be a set of the TSP instances defined by distance matrices with elements (i.e., arc lengths) from a given distribution. Let the output of the applied branch and bound algorithm be presented by two sequences of real numbers (b_i) and (c_i), $i = 1, 2, \ldots, |\mathcal{N}|$, where b_i is the number of solved relaxation tasks and c_i is the value of the corresponding complexity index, both referring to the i-th instance of the TSP in the set $\mathcal{N}$. Under this assumptions we can interpret sequences (b_i) and (c_i) as the realizations of random variables B and C in a statistical experiment.

The measure of dependence of a complexity index and the number of solved relaxation tasks can be interpreted as a degree of dependence of the random variables C and B and estimated by the methods of correlation analysis.

The coefficient of linear correlation for two sequences (b_i) and (c_i) is defined by

$$C_{BC} = \frac{1}{\vartheta_B \vartheta_C} \sum_{i=1}^{|\mathcal{N}|} (b_i - \bar{m}_B)(c_i - \bar{m}_C),$$

where $\bar{m}_B$, $\bar{m}_C$ and ϑ_B, ϑ_C are mean values and variances of the corresponding sequences (b_i) and (c_i), respectively.

Under the assumption that the random variables B and C obey the normal distribution, the correlation coefficient C_{BC} is a reliable estimation of linear dependence of the random variables B and C.

The efficiency of the complexity index can be statistically estimated measuring the linear correlation between the index value and the number of relaxation tasks solved within the B&B algorithm.

Several invariants can be considered as complexity indices for the TSP with respect to B&B algorithms based on SDP relaxation [26]:

Let X be the solution of SDP relaxation (19)–(23) and $L = X + h_n I - J$. Then L determines the weighted graph $W_L = (V, E_L, C_L)$, where $E_L = \{\{i, j\} \in E \mid l_{ij} < 0\}$ and $C_L = 2I - L$, the corresponding unweighted graph $G_L = (V, E_L)$ and a stochastic matrix $S_L = I - \frac{1}{2}L$. The most efficient indices introduced in [26] are the following:

I_1: the number of edges of G_L

I_2: the second smallest eigenvalue of the Laplacian of G_L

I_3: the entropy of S_L, i.e., value equal to $\sum_{(i,j)\in E_L} \frac{l_{ij}}{2} \log_2 \Big(- \frac{l_{ij}}{2} \Big) - \frac{n}{2}$

I_4: $\sum_{i=1}^{n} |\mu_i - \mu_i^*|$, where $\mu_1, \mu_2, \ldots, \mu_n$ and $\mu_1^*, \mu_2^*, \ldots, \mu_n^*$ are sequences of non-decreasing eigenvalues of the Laplacians of G_L and a Hamiltonian circuit, respectively.

I_5: the same sum as in I_4 but with eigenvalues of the Laplacian of W_L instead of G_L.

I_6: the number of vertices of the G_L with degrees greater than 2.

The efficiency of indices I_k, $k = 1, \ldots, 6$, has been investigated in [26]:

For each dimension 20, 25, 30, 35 we consider 50 randomly generated TSP instances with distances uniformly distributed in the interval [1,999]. To each instance one of B&B algorithms based on SDP relaxation is applied (see Subsection 2.1).

The coefficients of the linear correlation between values of indices $I_k (k = 1, \ldots, 6)$ and the number of relaxation tasks for dimensions $n = 20, 25, 30, 35$ are summarized in Table 1. Results indicate that the most reliable indices are I_1, I_4 and I_6 with almost significant correlation.

TABLE 1. Values of the linear correlation coefficients

index / n	I_1	I_2	I_3	I_4	I_5	I_6
20	0.53	0.35	0.51	0.53	0.53	0.53
25	0.48	0.49	0.21	0.48	0.48	0.49
30	0.29	0.21	0.32	0.29	0.42	0.33
35	0.56	0.52	0.37	0.56	0.38	0.55
average value	0.47	0.39	0.35	0.47	0.45	0.48

Index I_4 is a spectrally based invariant and, having in view facts from Subsection 1.2, one would expect that it performs better than I_1 and I_6. The obtained experimental results presented in this subsection indicate the lack of theoretical explanations of phenomena with complexity indices, the need for experiments with instances of higher dimensions and, perhaps, the need for better classification of graph invariants than the intuitive approach, adopted in this study on highly informative graph invariants.

An idea how to improve the results with complexity indices is already given in another context in [29, pp. 23–25]. One can consider linear combinations of already defined complexity indices as well as the invariants of some short edge subgraphs of the input weighted graph.

We shall describe now our adaptive procedure for solving TSP.

The most important parameter in the adaptive solution approach for TSP is I^* which represents the estimated value of the complexity index I corresponding to the maximal allowed number of solved relaxation tasks R_S. In general R_S depends on the problem dimension n and this function is chosen by the user, while I^* depends on both n and statistical distribution of TSP instances. The value of I^* is used to classify instances in hard and easy classes in the following way [29]:

(1) easy instances, $I \leqslant I^*$, and (2) hard instances, $I > I^*$.

More precisely, when $I \leqslant I^*$, then the number of generated subproblems within the branch and bound procedure is expected to be less than R_S, while for $I > I^*$ this number is expected to be greater than R_S.

In the case when the solution process for TSP is based on SDP relaxation easy instances can be solved by one of B&B algorithms from Subsection 2.1, while hard instances are handled by a heuristic developed in [27]. The heuristic uses limited branching based on the number of edges with weights equal to one in the graph W_L. Namely, already mentioned experiments with a set of 50 randomly generated TSP instances for each of dimensions n=20, 25, 30, 35 show that high percentage of edges from W_L with weights equal to one stay in the optimal Hamiltonian circuit, which is illustrated in Table 2. This suggests the following modification of the branch and bound algorithms from Subsection 2.1. Starting from an initial upper bound obtained by the 3-opt heuristic and the solution X of the initial SDP relaxation, all edges from W_L with weights equal to one are fixed, and the branching is performed on the remaining edges.

Table 2

1	2	3	4
20	79.8%	97.7%	84%
25	84.1%	98.2%	78%
30	82.2%	98.1%	74%
35	85.4%	97.4%	60%

The proposed heuristic solves the TSP by limited branching and therefore it has in the worst case exponential complexity. Nevertheless it performs very well in practice, which is illustrated in Table 3. The test examples are the same as in Table 2.

The columns in Table 2 contain the following data:

(1) dimension n of TSP;
(2) the average percentage of the edges with weights equal to one in W_L w.r.t. n;
(3) the average percentage of edges from W_L with weights equal to one which stay in the optimal solution;
(4) the percentage of TSP instances for which all edges from W_L with weights equal to one stay in the optimal solution.

TABLE 3. Performance of the heuristic

1	2	3	4	5	6
20	93.6%	53.2%	0.14%	5	31
25	85.1%	59.6%	0.12%	7	80
30	79.2%	72.9%	0.37%	8	47
35	62.5%	89.6%	0.62%	10	92

The columns in Table 3 contain the following data:

(1) dimension of TSP;
(2) percentage of instances for which optimal solution was reached;
(3) percentage of instances for which the heuristic improved the initial upper bound obtained by 3-opt heuristic;
(4) average relative distance from the optimal objective function value $(((f_H - f_{\text{opt}})/f_{\text{opt}})\%)$;
(5) average number of subproblems solved within the heuristic;
(6) maximal number of subproblems solved within the heuristic.

In [27] the described adaptive solution procedure was tested on the same set of randomly generated instances used to generate Tables 1,2 and 3.

Before solving the instances we need first to decide upon the value of R_S-the maximal number of relaxation tasks which are allowed to be solved within the adaptive procedure. Here for each dimension we take that R_S is equal to the maximal number of subproblems solved within the heuristic given in column 6 of Table 3. As suggested in [27] the most reliable estimation for I_k^* was achieved by averaging those values in the correlation field which were equal (or approximatively equal) to R_S.

The performance of the adaptive solution procedure can be measured by two parameters f and t determined on the basis of the whole set of considered TSP instances. The parameter f represents the average percentage difference between the objective function value f_{ad} obtained by the adaptive procedure and the optimal objective function value f_{opt} determined by the B&B algorithm, i.e.

$$f = \frac{f_{\text{ad}} - f_{\text{opt}}}{f_{\text{opt}}}\%.$$

Value t is the "saving" of CPU-time, i.e., the percentage difference between the average number of solved relaxation tasks within the B&B and the adaptive procedure.

In order to measure the quality of the adaptive solution with one parameter we use the value $k = t/(1 + f/100)$ proposed in [29].

The adaptive procedure was tested on the same sets of TSP instances for each of complexity indices $I_k,\ k = 1, \ldots, 6$ as before. According to [27], the best performance was obtained for I_6.

In Table 4 we report on relevant details for this case.

TABLE 4. Results of the adaptive procedure for I_6

n	20	25	30	35
I_6^*	14	12	11	12
No. of hard instances	10	11	25	23
No. of detected hard instances	11	15	27	23
No. of easy instances	37	36	23	25
No. of detected easy instances	36	32	21	25
No. of incorrectly classified instances	7	16	10	10
exactness of decision	85.1%	66.00%	79.2%	79.2%
f	0.01%	0.05%	0.21%	0.21%
average No. of subproblems in adaptive algorithm	13	36	56	61
average No. of subproblems in optimal algorithm	23	69	198	238
t	43.5%	47.8%	71.7%	74.4%
k	43.5%	47.8%	71.6%	74.2%

TABLE 5

index	I_1	I_2	I_3	I_4	I_5	I_6
average exactness of decision	76.8%	73.7%	69.5%	76.3%	71.0%	77.4%
average value k	57.1%	46.4%	56.6%	56.1%	59.3%	59.3%

The results for the remaining complexity indices were also reasonable. Table 5 summarizes the most important indicators of the efficiency of the adaptive procedure: the average exactness of decision per dimension and the average value k per dimension.

On the basis of results presented in Tables 4 and 5 we can conclude that both the exactness of decision and the quality measure k are satisfactory for all complexity indices.

Another use of complexity indices in the TSP solving procedures is described in [32]. A new search strategy in B&B algorithms has been developed. The traditional backtrack strategies (depth-first search and breath-first search) are not optimal and therefore the so called *jumptrack* strategies have been considered in the literature. In such strategies any active subproblem can be selected following certain criteria. Usually, one selects a subproblem with smallest lower bound. Complexity indices have been introduced in [32] to help to select next subproblem. An ordered list of *interesting* subproblems has been introduced. The strategy takes the first subproblem from the list and branches it using the depth-first search until a complexity

index starts to increase. Several variants of such a search strategy have been described in [32]. Computational results show better behaviour of these strategies compared with other ones.

2.3. Data Clustering. In this subsection we consider the problem of clustering data (see, e.g., [1], [3]). Clusters in some cases are obtained by solving some optimization problems. Again the algebraic connectivity of a graph is useful.

The algebraic connectivity is known to be a very useful parameter for describing the "shape" of a graph (see, e.g., [31, p. 266]). Indeed, low algebraic connectivity shows small connectivity and girth and high diameter, although such a statement lacks a precise formulation. In the context of clustering, low algebraic connectivity indicates that the graph has good clustering properties.

The data are usually represented by vectors from $\mathbb{R}^n$. Euclidean or other kind of distance function $d(x, y)$ is assumed to be defined for any $x, y \in \mathbb{R}^n$. Given a set of vectors from $\mathbb{R}^n$, the problem is to partition it into subsets called *clusters* under various conditions. Clustering methods are supposed to produce clusters which have the property that vectors from the same cluster in some sense are "closer" one to the other than the vectors from different clusters. The number of clusters may but need not to be given in advance. Sometimes cardinalities of clusters are given or limited by additional conditions.

There are difficulties in applying traditional clustering procedures to discrete data. We describe a graph theoretical approach in clustering binary vectors. New clustering procedures are combined from several algorithms and heuristics from graph theory and combinatorial optimization.

We consider clustering of discrete data. A typical example of discrete data are binary vectors, i.e. elements of B^n where $B = \{0, 1\}$. When standard clustering procedures (see, e.g., [1], [3]) are applied to binary vectors, the resulting clustering has usually a low quality. Among other things, the clustering is highly dependent of the ordering of vectors.

To avoid these difficulties it seems reasonable to use specific properties of discrete data and to apply combinatorial, including graph theoretical, tools in handling the problem. We have developed a number of complex graph theoretical procedures for clustering binary vectors [16], [18], [20]. See also [22] and [61].

A *hypercube* H_n of dimension n is the graph whose vertex set is B^n and two n-tuples are adjacent if they differ in exactly one coordinate. The number of coordinates in which n-tuples $x, y \in B^n$ differ is called the *Hamming distance* between x and y.

For a graph G we define its *k-th power* G^k. The graph G^k has the same vertex set as G and vertices x and y are adjacent in G^k if they are at (graph theoretical) distance at most k in G. For $k = 0$ the graph G^k consists of isolated vertices. For $k = 1$ we have $G^k = G$. If X is a subset of the vertex set of a graph G, then $G(X)$ denotes the subgraph of G induced by X.

Let $X \subset B^n$ be a set of binary vectors (n-tuples) which is to be clustered. Our procedures for clustering make use of the graph sequence

$$H_n^0(X),\ H_n^1(X),\ H_n^2(X), \ldots, H_n^n(X)$$

which is called the *basic graph sequence*.

Note that two vectors $x, y \in X$ are at the Hamming distance k if they are not adjacent in $H_n^{k-1}(X)$ and are adjacent in $H_n^k(X)$. For $i = 1, \ldots, n$ the graph $H_n^i(X)$ has all edges from $H_n^{i-1}(X)$ plus those ones connecting vectors at *Hamming* distance i. $H_n^0(X)$ has only isolated vertices while $H_n^n(X)$ is a complete graph.

Let the vertex set X of a graph G be partitioned into subsets $X_1, X_2, \ldots, X_m$. A *condensation* of G is a weighted graph on vertices $x_1, x_2, \ldots, x_m$ (called *supervertices*) in which x_i and x_j are connected by an edge if there is at least one edge between X_i and X_j in G. Both supervertices and edges in the condensation carry weights. The weight of the supervertex x_i is equal to $|X_i|$ while the weight of the edge between x_i and x_j is equal to the number of edges between X_i and X_j. We consider a condensation as a multigraph where edge weights are interpreted as edge multiplicities while supervertices as vertices and supervertex weights are ignored.

In the clustering procedure, which will be described, some algorithms and heuristics, described in the literature, will be used. We shall define them here (see [18] for details).

Algorithm CP. This is an algorithms for finding components of a graph.

Algorithm JM. This is an algorithm for partitioning a connected (multi-) graph into two parts.

Heuristic KL. This is a heuristic for partitioning the vertex set of a (multi-) graph into two parts of given cardinalities with a minimum number of edges between vertices from different parts [45].

Let X be a set of binary vectors of dimension n and suppose we have to cluster it into k ($k > 1$) clusters. For $k = 2$ we consider the problem in two variants: 1° Cluster cardinalities are not given, 2° Cluster cardinalities are given.

Our procedure consists of two phases.

Phase 1. We form the basic graph sequence. Let c_i be the number of components of $H_n^i(X)$. Components are sequentially determined in graphs from the basic sequence by algorithm CP. We have $c_0 = |X| \geqslant c_1 \geqslant c_2 \geqslant \cdots \geqslant c_n = 1$.

There is a non-negative integer s such that $c_s \geqslant k > c_{s+1}$. If $c_s = k$, the components of $H_n^s(X)$ are clusters and the procedure is finished. If $c_s > k > c_{s+1}$ we proceed to Phase 2.

Phase 2. We distinguish cases: 1) $k = 2$ and 2) $k > 2$.

Case $k = 2$**.** Now $H_n^{s+1}(X)$ is connected and we consider the condensation of the graph $H_n^{s+1}(X)$ in which components of $H_n^s(X)$ play role of supervertices.

We consider two subcases:

1° Cluster cardinalities are not given;

2° Cluster cardinalities are given.

Subcase 1°**.** If $c_s > 10$ any of the following two procedures can be applied to the condensation of $H_n^{s+1}(X)$:

a) algorithm JM;

b) heuristic KL.

In any of these cases we get two clusters and the whole procedure is finished.

In variant b) the user can select the range of cluster cardinalities and the number of randomly generated starting clusterings. The result in variant a) can serve as a hint for the range of cluster cardinalities in variant b).

If $c_s \leqslant 10$, we form all partitions of the vertex set of the condensation of $H_n^{s+1}(X)$ into two parts since there are only $2^{c_s} - 2$ such partitions. We find the best partition with respect to a selected quality criterion (e.g., minimizing the edge number between two parts). The whole procedure is thus finished.

Subcase 2°**.** We apply algorithm JM to the condensation of $H_n^{s+1}(X)$. If the partition thus obtained shows cluster cardinalities required, we have done. Otherwise we apply heuristic KL to the graph H_n^{s+1} where the starting partitions are formed on the basis of information obtained by the working of the algorithm JM. Let p, q $(p \geqslant q)$ be the required cluster cardinalities. Let algorithm JM have given a solution with cluster cardinalities r, s $(r \geqslant s)$. If $p < r$, from the cluster of cardinality r we choose those p vertices for which moduli of the coordinates of the eigenvector from algorithm JM are as great as possible. If $p > r$, then $q < s$, and from the cluster of cardinality s we choose q vertices as above. The result of the working of heuristic KL for the starting partition so formed is compared with result for other, randomly generated, starting partitions.

Case $k > 2$**.** Now we have $c_s > k > c_{s+1}$ and we get a clustering into k clusters in one of the following two ways

1) by splitting some of c_{s+1} components of the graph $H_n^{s+1}(X)$ into parts;

2) by uniting some of c_s components of $H_n^s(X)$.

We use first way if k is closer to c_{s+1} than to c_s and the second one otherwise.

Splitting components we perform by partitioning a component into two parts and by iterating this procedure. First we partition components of $H_n^{s+1}(X)$ which do not exist in $H_n^s(X)$ and if there are not sufficiently many such components we treat sequentially those which exist in $H_n^i(X)$ and not in $H_n^{i-1}(X)$ for $i = s, s-1, \ldots$. For components of $H_n^i(X)$ $(i = s+1, s, \ldots)$ we form condensations with supervertices corresponding to components of $H_n^{i-1}(X)$ and for each condensation we determine the ratio of the algebraic connectivity and the number of vertices. Condensations are ordered by this ratio and partitioned sequentially into two parts starting from those with a smallest ratio. In each step of partitioning the newly generated components are treated as above. For partitioning components into two parts we apply the procedure from the case $k = 2$ above.

When uniting components we consider all possibilities of uniting if $c_s - k < 4$. Otherwise we apply the Ward method, which is one of the best hierarchical clustering methods (see, e.g., [1], [3]).

Algorithm JM and the calculation of the algebraic connectivity have complexity $O(|X|^3)$ while other parts of the procedure have lower complexities. Therefore the whole procedure has complexity $O(|X|^3)$ and this is the same as in many standard clustering procedures. However, theoretical reasons and numerical experiments show that the graph theoretical procedure is superior to standard procedures in clustering binary vectors.

3. Defining Graph Invariants by Solving Optimization Problems on Graphs

Section 3 elaborates Assertion 1. Subsection 3.1 introduces star bases and a canonical star basis of a graph. This basis is obtained by solving several optimization tasks and is useful in treating the graph isomorphism problem. It is shown in 3.2 that using canonical star basis one can construct a polynomial time algorithm for checking the isomorphism of graphs with bounded multiplicities of eigenvalues. Other examples of defining highly informative graph invariants are given in 3.3.

3.1. Star Partitions and Canonical Star Bases. Spectral techniques in graph theory are based on the eigenvalues of the adjacency and other graph matrices. These techniques have been further developed by considering, in addition, some invariants of eigenspaces of graphs, namely graph angles. Introduction of star partitions and canonical bases can be considered as a result of efforts in the same direction – to enrich spectral techniques in graph theory.

Let G be a graph with vertices $1,\dots,n$ and $(0,1)$-adjacency matrix A. Let $\mu_1,\dots,\mu_m$ $(\mu_1 > \cdots > \mu_m)$ be the distinct eigenvalues of A, with corresponding eigenspaces $S_1,\dots,S_m$. For each $i \in \{1,\dots,m\}$, let k_i be the multiplicity of μ_i.

Let us consider the spectral decomposition of A:

$$A = \mu_1 P_1 + \cdots + \mu_m P_m.$$

Thus P_i represents the orthogonal projection onto S_i and, if $\{\mathbf{e}_1,\dots,\mathbf{e}_n\}$ is the standard orthonormal basis of $\mathbb{R}^n$, the vectors $P_1e_1,\dots,P_ie_n$ constitute a *eutactic star* (see [33]). Norms of vectors from these eutactic stars are *angles* of a graph. Rows (or columns) of P_i are vectors of the eutactic star associated with S_i. The Gram matrix of these vectors is just the matrix P_i.

A partition $X_1\dot\cup\cdots\dot\cup X_m$ of the vertex set $\{1,\dots,n\}$ is called a *star partition*, with *star cells* $X_1,\dots,X_m$, if for each $i \in \{1,\dots,m\}$ the vectors P_ie_j $(j \in X_i)$ are linearly independent. In this situation a comparison of dimensions shows that $|X_i| = k_i$ $(i = 1,\dots,m)$ and the vectors P_ie_j $(j \in X_i)$ form a basis $\mathcal{B}_i$ of S_i. Then $\mathcal{B}_1 \cup \cdots \cup \mathcal{B}_m$ is a basis of $\mathbb{R}^n$, called in [33] a *star basis* corresponding to A (a construction applicable to any symmetric matrix with real entries).

It was shown in [33] that the following three theorems hold.

Theorem 3.7. *Every graph G has a star partition.*

Theorem 3.8. *The partition $X_1\dot\cup\cdots\dot\cup X_m$ is a star partition if and only if for each $i \in \{1,\dots,n\}$, μ_i is not an eigenvalue of $G - X_i$.*

Theorem 3.9. (Reconstruction theorem) *A graph G is uniquely determined by an eigenvalue μ_i, the subgraph $G - X_i$, and the subgraph $G - E(X_i)$ where X_i is a star cell belonging to μ_i and $E(X)$ is the set of edges of G whose both end points are in the set X.*

It has been shown in [34] that it is possible to construct one star partition of a graph G in time bounded by a polynomial function of the number n of vertices of the graph G. One approach is related to the Hungarian algorithm for constructing

a perfect matching in a bipartite graph. The complexity of this approach is at most $O(n^4)$. The other approach uses matroid theory and a rough complexity estimation is $O(n^5)$.

Let S be the matrix of a basis $\mathcal{S}$ of eigenvectors (in particular, a star basis or even some kind of the canonical basis) of $\mathbb{R}^n$ corresponding to a graph G. Columns of S are vectors of $\mathcal{S}$. We have $S = (S_1 \cdots S_m)$, where S_i is the matrix of the star basis $\mathcal{S}_i$ of the eigenspace $\mathcal{E}(\mu_i)$. The adjacency matrix A can be expressed in the form $A = S\Lambda S^{-1}$ where Λ is a diagonal matrix with eigenvalues of A at the main diagonal. Putting $(S^{-1})^T = (X_1 \cdots X_m)$ and using relation $S^{-1}S = I$ we get $X_i = (S_i^T S_i)^{-1} S_i^T$. This is sufficient to prove the following proposition.

Proposition 1. We have $A = \mu_1 S_1 (S_1^T S_1)^{-1} S_1^T + \cdots + \mu_m S_m (S_m^T S_m)^{-1} S_m^T$.

Proposition 1 gives the spectral decomposition of the adjacency matrix A of a graph. Hence, $P_i = S_i (S_i^T S_i) S_i^T$ is the projection onto the i-th eigenspace $\mathcal{E}(\mu_i)$. The matrix $W_i = S_i^T S_i$ is the Gram matrix of the star basis $\mathcal{S}_i$ of the eigenspace $\mathcal{E}(\mu_i)$. (If the basis is orthonormal W_i is equal to a unit matrix).

If $\mathcal{S}$ is a precisely defined canonical basis, eigenvalues $\mu_1, \ldots, \mu_m$ and matrices $S_1, \ldots, S_m$ comprise the canonical code of a graph. All graph properties can be derived from the canonical code of the graph at least by reconstructing the adjacency matrix A and applying some graph theoretical algorithms. Of course, it would be of interest to derive procedures for a more direct link between the canonical code of the graph and the graph properties we are interested in.

The problem is how to select a canonical basis of eigenvectors. Without additional restrictions there are infinitely many such bases and if we want to select one by an optimization task, the set of feasible solutions is infinite and compact so that we are led to a problem of continual (global) optimization. However, if we restrict ourselves to star bases, we encounter a problem of finding an extremal value of a function defined on a finite set since the number of star bases is finite. Hence, we have to solve a problem of *combinatorial optimization*.

Let us note that the number of star bases of a graph, although finite, is very large since we have to consider star bases corresponding to all orders (permutations) of the vertex set.

Given a graph G on n vertices, the notion of a canonical basis of eigenvectors of G for $\mathbb{R}^n$ which is related to the notions of the *star basis* and the *star partition* of G has been introduced in [33]. This canonical basis is called the *canonical star basis*. The canonical star basis is unique for a graph i.e. it does not depend on the vertex labelling. Finding this canonical basis involves several extremal tasks similarly as in finding an extremal binary number (see Subsection 1.2). To construct this basis we have introduced a total order of graphs, called CGO (canonical graph ordering) and a quasi-order of vertices called CVO (canonical vertex ordering). Both CGO and CVO are defined recursively in terms of graphs with fewer than n vertices.

The definition of the canonical star basis enables the formulation of the following theorem (whose proof is obvious):

Theorem 3.10. *Two graphs are isomorphic if and only if they have the same eigenvalues and the same canonical bases.*

Several improvements to the procedure from [33] have been proposed in [19]. See also [35]. One can show that the procedure is reduced in some parts to special cases of some well-known combinatorial optimization problems, such as the maximal matching problem, the minimal cut problem, the maximal clique problem, etc. This sheds some light on the algorithmical complexity of the procedure for finding a canonical basis, i.e., tells something about the graph isomorphism problem.

A procedure for testing the isomorphism of graphs, which is based on the spectral decomposition of matrices has been described in [69].

Since the canonical star basis together with eigenvalues of G determines G up to isomorphism, algorithms for finding the canonical basis and some of its variations are studied in [19]. The emphasis is given on the following three special cases: graphs with distinct eigenvalues, graphs with bounded eigenvalue multiplicities and strongly regular graphs. This technique provides another proof of a result of L. Babai et al. [5] that isomorphism testing for graphs with bounded eigenvalue multiplicities can be performed in polynomial time (see next subsection). One can show that the canonical basis in strongly regular graphs is related to the graph decomposition into two strongly regular induced subgraphs (these decompositions are described in [41]). Examples of distinguishing between cospectral strongly regular graphs by means of the canonical basis are provided. The behaviour of star partitions of regular graphs under operations of complementation and switching is studied in [19] as well.

The canonical star basis (and star bases in general) can be very useful in studying other problems.

3.2. The Maximal Clique Problem and Bounded Multiplicities. The procedure of finding the canonical star basis can be designed so that it contains a kind of the maximal clique problem. This is especially useful in the case of graphs with bounded multiplicities of eigenvalues. This subsection is mainly written following [19].

A star basis of a graph G with distinct eigenvalues $\mu_1, \ldots, , \mu_m$ of multiplicities $k_1, \ldots, k_m$ is characterized by weighted graphs $W_1, \ldots, W_m$ of orders $k_1, \ldots, k_m$, respectively (see the comment after Proposition 1 in the previous subsection). In *orthodox* star bases (i.e., bases among which the canonical star basis is selected) the sequence $W_1, \ldots, W_m$ is lexicographically maximal using ordering of weighted graphs specially defined in [33], [19], [35], i.e., canonical weighted graphs ordering (CWGO) and canonical weighted graphs vertex ordering (CWGVO). Instead of finding several (or all) star bases and selecting maximal ones among them, we can find maximal sequences $W_1, \ldots, W_m$ and check whether star bases with such sequences exist.

Let us assume that G is a connected graph. Then μ_1 is a simple eigenvalue and W_1 is reduced to squares of coordinates of the eigenvector belonging to μ_1. As proved in [34] any vertex can form a star cell X_1 for μ_1. Hence, for W_1 we select the square of a coordinate of the eigenvector belonging to μ_1 with a maximal modulus.

Next we have to select X_2, the star cell belonging to μ_2. That means we have to find a principal submatrix W_2 of P_2 of order k_2 so that the weighted graph determined by W_2 is maximal. Now we can consider P_2 as a weighted graph in

which we have to find a clique of order k_2 with a maximal weight if we define the weight of a clique just to be the matrix W_2. We decide which of two given cliques has greater weight in accordance with ordering of weighted graphs (CWGO). The complexity of this decision depends on the order of the clique.

Finding a maximal clique in a weighted graph has been considered in [6]. The problem is essentially similar to the problem of finding a (maximal) clique in a graph without weights on edges [42]. Roughly speaking, we have to check all $\binom{n}{k_i}$ principal submatrices of order k_i. If k_i is fixed, $\binom{n}{k_i}$ is a polynomial in n of degree k_i, i.e., $\binom{n}{k_i} = O(n^{k_i})$. If the size of the clique is not restricted, the problem of finding a maximal clique (decision version) is known to be NP-complete. Note that $\binom{n}{cn}$ for a fixed c $(0 < c < 1)$ is not polynomially bounded.

Once we have found X_2 such that W_2 is maximal we can decide easily whether a star partition exists in which X_2 is a cell. (A necessary but not sufficient condition for this is that the graph $G - X_2$ does not have an eigenvalue μ_2. An example is provided by the Petersen graph; there is no star partition in which $X_1 \cup X_2$ induces C_6.)

More generally, given any partially built partition we can in a polynomial time, using algorithms from [34], extend it to a star partition or establish that this cannot be done.

It should be noted that our reductions of the graph isomorphism problem to some well known combinatorial optimization problems do not involve general cases of these problems; in fact, we have special cases determined by special features of weight matrices in question (eigenvector and projector matrices). This is important especially in the case when the general problem is NP-complete (NP-hard) as in the case of the problem of finding a maximal clique. Note that such reductions of the graph isomorphism problem to special cases of NP-hard problems (of unknown complexities) have been already noticed elsewhere (see [64] where a special case of the maximal clique problem occurs).

It has been proved by L. Babai et al. [5] that isomorphism testing for graphs with bounded multiplicities of eigenvalues can be performed in a polynomial time.

Using above ideas we can confirm this result.

If eigenvalue multiplicities are bounded by an absolute constant a, then the size of the maximal clique we have to find is limited also to at most a. It is known that a maximal clique of limited size can be found in a polynomial time, i.e., in time $O(n^a)$ in this case. We can in a polynomial time examine and keep information on all induced subgraphs whose vertex sets have cardinalities equal to eigenvalue multiplicities.

Hence we can find an orthodox star basis in a polynomial time. In fact, we can find all orthodox star bases in a polynomial time and go on in finding quasi-canonical bases and the canonical basis.

Ordering vertices in star cells of orthodox star bases by corresponding CWGVOs can be done in a time bounded by a function of the constant a. We can imagine

that for testing isomorphism of graphs in which each eigenvalue has multiplicity at most a we have prepared a (finite) table of automorphism groups and CVO for graphs with at most a vertices. (It is even possible to use all orderings-at most $a!$-of vectors in an orthodox star basis.) Hence, all maximization procedures, needed to find the canonical star basis, can be performed in polynomial time and the result by L. Babai et al. follows.

Remark 1. The result can be extended to graphs (as noted also in [5]) in which all but one eigenvalue have bounded multiplicities and this property is hereditary (holds also for any induced subraph). The hereditarity of the property in question was not assumed in [5]. Perhaps it can be avoided also here but we have assumed it since the multiplicity of an eigenvalue can increase when going from graphs to subgraphs and the subgraph induced by the star cell corresponding to the eigenvalue of unbounded multiplicity can have more than one eigenvalue above the bound for eigenvalues. In this case we modify the notion of an orthodox star basis in such a way that the matrix W_i corresponding to the eigenvalue μ_i, whose multiplicity is not bounded, is put at the end of the original matrix sequence $W_1, \ldots, W_m$ which should be lexicographically maximal (i.e., we have now the sequence $W_1, \ldots, W_{i-1}, W_{i+1}, \ldots, W_n, W_i$). Namely, we readily find in polynomial time star cells corresponding to matrices $W_1, \ldots, W_{i-1}, W_{i+1}, \ldots, W_n$ while the cell of unbounded size, corresponding to W_i is determined by the vertices which remain. It is also not necessary to order vertices in this star cell; it is sufficient to use the Reconstruction theorem (Theorem 3.3 from Subsection 3.1).

Remark 2. Let us finally note that the graph isomorphism problem can be also reduced to a problem of finding a certain kind of matching in an auxiliary bipartite graph. However, the number of vertices of this bipartite graph depends on multiplicities of eigenvalues. The graph in question is the incidence graph between the set of distinct eigenvalues of the original graph G and the set of subsets of the vertex set of G with cardinalities equal to multiplicities of eigenvalues. There is an edge between vertices representing an eigenvalue μ and a subset X of vertices if and only if $|X|$ is equal to the multiplicity of μ and $G - X$ does not have an eigenvalue μ. A star partition of G is represented by a matching which satisfies some additional requirements but we shall not go into details.

As indicated, using canonical star bases we can relate the graph isomorphism problem to some problems of combinatorial optimization (the problem of finding a maximal matching, the maximal clique problem, the problem of finding a bipartition with an extremal number of edges between the parts, etc.). Some of these problems can be solved in polynomial time while the others are known to be NP-hard. Arguments pro and contra the existence of a polynomial algorithm for the graph isomorphism problem both exist.

3.3. Other Highly Informative Graph Invariants.

In this final subsection we shortly report on some other graph invariants which can be considered as highly informative.

Spectra of weighted adjacency matrices have served to introduce a new important graph invariant in [66]. For a connected graph G we introduce the class $\mathring{A}_G$ of matrices $A = (a_{ij})$ for which $a_{ij} > 0$ if i and j are adjacent and $a_{ij} = 0$ otherwise. Let $\mu_1, \mu_2, \ldots, \mu_m$ $(\mu_1 > \mu_2 > \cdots > \mu_m)$ be distinct eigenvalues of A with multiplicities $k_1 = 1, k_2, \ldots, k_m$, respectively. Let $\mu(G) = \max k_2$, where maximum is taken over the class $\mathring{A}_G$. For example, $\mu(K_n) = n-1$ and $\mu(K_{3,3}) = 4$. It is proved that G is planar if and only if $\mu(G) \leqslant 3$. It is conjectured that $\mu(G) \geqslant \chi(G) - 1$, where $\chi(G)$ is the chromatic number of G. The validity of this conjecture would imply the four color theorem!

The Lovász *theta function* has been introduced in [53] when solving a long standing problem in information theory. The theta function can be defined in many equivalent ways: via an extremal problem concerning eigenvalues of graphs, via a semidefinite programming model and in some other ways. For a short review on this important graph invariant see [39].

See [10] for other interesting graph invariants.

References

[1] D. Acketa, *Selected Topics in Pattern Recognition with Applications*, (in Serbian), Mathematical Institute, Novi Sad, 1986

[2] F. Alizadeh, *Combinational optimization with interior point methods and semidefinite matrices*, Ph.D. Thesis, University of Minnesota, 1991

[3] M. R. Anderberg, *Cluster Analysis for Applications*, Academic Press, New York, 1973

[4] L. Babai, *On the complexity of canonical labelling of strongly regular graphs*, SIAM J. Computing 9 (1980), 212–216

[5] L. Babai, D. Yu. Grigorjev, D. M. Mount, *Isomorphism of graphs with bounded eigenvalue multiplicity*, Proc. 14th Annual ACM Symposium on Theory of Computing, San Francisko 1982, New York, 1982, 310–324

[6] E. Balas, V. Chvatal, J. Nešetril, *On the maximum weight clique problem*, Mathematics of Operation Research, 12:3 (1987), 522–535

[7] R. Bellman, K. Fan, *On systems of linear inequalities in Hermitean matrix variables*, V. Klee, ed., Proc. Symposium in Pure Mathematics, Amer. Math. Soc., Providence, RI, 1963, 1–11

[8] B. Borchers, *CSDP, a C library for semidefinite programming*, Optimization Methods and Software 11-12 (1999), 613–623

[9] B. Borchers, *CSDP 2.3 user's guide*, Optimization Methods and Software 11-12 (1999), 597–611

[10] N. S. Chaudhari, D. B. Phatak, *A new approach for graph isomorphism*, J. Combin. Inform. System Sci. 13:1-2 (1988), 1–19

[11] B. Craven, B. Mond, *Linear programming with matrix variables*, Linear Algebra Appl. 38 (1981), 73–80

[12] D. Cvetković, *Graphs and their spectra*. Univ. Beograd, Publ. Elektrotehn. Fak. Ser. Mat. Fiz., No. 354 – No. 356 (1971), 1–50

[13] D. Cvetković, *The main part of the spectrum, divisors and switching of graphs*, Publ. Inst. Math. (Beograd) 23(37) (1978), 31–38

[14] D. Cvetković, *Combinatorial Matrix Theory with Applications to Electrical Engineering, Chemistry and Physics*, (Serbo-Croatian), Naučna knjiga, Beograd, 1980

[15] D. Cvetković, *Adaptive solving of the travelling salesman problem*, (Serbian), unpublished report, Faculty of Electrical Engineering, Belgrade, 1987, 1–48

[16] D. Cvetković, *Combinatorial algorithms and heuristics for clustering points of a hypercube, I, II, III, IV*. (in Serbian), University of Belgrade, Faculty of Electrical Engineering, Belgrade, pp. 32 + 39 + 53 + 34, unpublished report, 1988

[17] D. Cvetković, *Some graph invariants based on the eigenvectors of the adjacency matrix*, R. Tošić, D. Acketa, V. Petrović, R. Doroslovački, eds., Graph Theory, Proc. Eighth Yugoslav Seminar on Graph Theory, Novi Sad, April 17-18, 1987, Univ. Novi Sad, Inst. Math., Novi Sad, 1989, 31–42

[18] D. Cvetković, *Graph theoretical procedures in clustering discrete data*, Univ. Beograd, Publ. Elektrotehn. Fak., Ser. Mat. 3 (1992), 21–26

[19] D. Cvetković, *Star partitions and the graph isomorphism problem*, Linear and Multilinear Algebra 39 (1995), 109–132

[20] D. Cvetković, *A graph theoretical procedure for clustering binary vectors*, Ars Combinatoria 46(1997), 267–276
[21] D. Cvetković, M. Čangalović, V. Dimitrijević, L. Kraus, M. Milosavljević, S. Simić, *TSP-SOLVER-A programming package for the traveling salesman problem.* Univ. Beograd, Publ. Elektrotehn. Fak. Ser. Mat. 1 (1990), 41–47
[22] D. Cvetković, M. Čangalović, Dj. Dugošija, V. Kovačević-Vujčić, S. Simić, J. Vuleta, red. D. Cvetković, V. Kovačević-Vujčić, *Combinatorial Optimization*, (in Serbian), *Matematička teorija i algoritmi*, Društvo operacionih istraživača Jugoslavije, Beograd, 1996.
[23] D. Cvetković, M.Čangalović, V. Kovačević-Vujčić, *Semidefinite programming and traveling salesman problem*, R. Petrović, D. Radojević, eds., Proceedings of XXV Yugoslav Symposium on Operations Research, Herceg Novi, Yugoslavia, 1998, 239–242
[24] D. Cvetković, M.Čangalović, V. Kovačević-Vujčić, *Semidefinite relaxations of the traveling salesman problem*, YUJOR 9:2 (1999), 157–168
[25] D. Cvetković, M. Čangalović, V. Kovačević-Vujčić, *Semidefinite programming methods for the symmetric traveling salesman problem*, G. Cornuejols, R. E. Burkard, G. J. Woeginger, eds., Integer Programming and Combinatorial Optimization, Proc. 7th Internat. IPCO Conf., Graz, Austria, June 1999, Lecture Notes Comp. Sci. 1610, Springer-Verlag, Berlin, 1999, 126–136
[26] D. Cvetković, M.Čangalović, V. Kovačević-Vujčić, *Complexity indices for the traveling salesman problem based on a semidefinite relaxation*, M. Vujošević, M. Martić, eds., Proceedings of XXVI Yugoslav Symposium on Operations Research, Beograd, Yugoslavia, 1999, 177–180
[27] D. Cvetković, M. Čangalović, V. Kovačević-Vujčić, *Adaptive approach to the traveling salesman problem using a semidefinite relaxation*, S. Cvetić, S. Guberinić, eds., Proceedings of XXVII Yugoslav Symposium on Operations Research, Beograd, Yugoslavia, 2000, 213–216
[28] D. Cvetković, V. Dimitrijević, M. Milosavljević, *A spectral traveling salesman problem complexity index*, (in Serbian), Zbornik radova XI bosansko-hercegovačkog simpozijuma iz informatike, Sarajevo-Jahorina '87, 1987, knj. 2, 276-1–276-5
[29] D. Cvetković, V. Dimitrijević, M. Milosavljević, *Variations on the Travelling Salesman Theme*, Libra Produkt, Belgrade, 1996
[30] D. Cvetković, M. Doob, I. Gutman, A. Torgašev, *Recent Results in the Theory of Graph Spectra*, North-Holland, Amsterdam, 1988
[31] D. Cvetković, M. Doob, H. Sachs, *Spectra of Graphs*, 3rd ed., Johann Ambrosius Barth, Heidelberg-Leipzig, 1995
[32] D. Cvetković, S. Mitrović-Minić, *A search strategy in branch and bound algorithms for the traveling salesman problem*, E. Krčmar-Nožić, ed., Proceedings of XX Yugoslav Symposium on Operations Research, Beograd, 1993, 137–140
[33] D. Cvetković, P. Rowlinson, S. Simić, *A study of eigenspaces of graphs*, Linear Algebra Appl. 182 (1993), 45–66
[34] D. Cvetković, P. Rowlinson, S. Simić, *On some algorithmic investigations of star partitions of graphs*, Discrete Appl. Math. 62 (1995),119–130
[35] D. Cvetković, P. Rowlinson, S. Simić, *Eigenspaces of Graphs*, Cambridge University Press, Cambridge, 1997
[36] M. Fiedler, *Algebraic connectivity of graphs*, Czechoslovak Math. J. 23 (1973), 298–305
[37] M. Fiedler, *Laplacian of graphs and algebraic connectivity*, Combinatorics and Graph Theory, vol. 25, Banach Center Publications, PWN-Polish Scientific Publishers, Warsaw, 1989, 57–70
[38] R. Fletcher, *Semidefinite matrix constraints in optimization*, SIAM J. Control Optim. 23 (1985), 493–513
[39] M. Goemans, *Semidefinite programming in combinatorial optimization*, Math. Program. 79 (1997), 143–161
[40] M. X. Goemans, D. P. Williamson, *Improved approximation algorithms for maximum cut and satisfability problems using semidefinite programming*, J. ACM 42 (1995), 1115–1145

[41] W. H. Haemers, D. G. Higman, *Strongly regular graphs with strongly regular decomposition*, Linear Algebra Appl. 114/115 (1989), 379–398
[42] P. Hansen, N. Mladenović, *A comparison of algorithms for the maximum clique problem*, YUJOR 2:1 (1992), 3–11
[43] C. Helmberg, F. Rendl, R. J. Vanderbei, H. Wolkowicz, *An interior point method for semidefinite programming*, SIAM J. Optimization 6 (1996), 342–361
[44] D. Karger, R. Motwani, M. Sudan, *Approximate graph coloring by semidefinite programming*, J. ACM 45 (1998), 246–265
[45] B. W. Kernighan, S. Lin, *An efficcient procedure for partioning graphs*, Bell Sys. Tech. J. 49 (1970), 291–307
[46] M. Kojima, S. Shindoh, S. Hara, *Interior point methods for the monotone semidefinite linear complementarity problem in symmetric matrices*, SIAM J. Optimization 7 (1997), 86–125
[47] V. Kovačević-Vujčić, M. Čangalović, J. Kratica, *Solving a semidefinite relaxation of the traveling salesman problem*, Central European J. Oper. Res. 10:4 (2002), 277–296.
[48] V. Kovačević-Vujčić, M. Čangalović, J. Kratica, *A predicter-correcter algorithm for a semidefinite relaxation of the traveling salesman problem*, J. Todorović, M. Vujošević, B. Vasić, eds., Proceedings of XXIX Yugoslav Symposium on Operations Research, Tara, 2002, IX13–IX16.
[49] G. Laporte, *The traveling salesman problem: An overview of exact and approximate algorithms*, European J. Operational Research 59 (1992), 231–247
[50] G. Laporte, *Exact algorithms for the traveling salesman problem and the vehicle routing problem*, Les Cahiers du GERAD G-98-37, July, 1998
[51] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys, *The Traveling Salesman Problem*, John Wiley& Sons, Chichester-New York-Brisbane-Toronto-Singapore, 1985
[52] J. H. van Lint, R. M. Wilson, *A Course in Combinatorics*, Cambridge University Press, Cambridge, 1992
[53] L. Lovász, *On the Shannon capacity of a graph*, IEEE Trans. Inform. Theory 25 (1979), 1–7
[54] M. Milosavljević, V. Dimitrijević, *A complexity analysis concerning the exact solution of the travelling salesman problem*, (in Serbian), Zbornik radova X bosansko-hercegovačkog simpozijuma iz informatike, Sarajevo-Jahorina '86, 1986, knj. 2, 271-1–271-6
[55] Y. Nesterov, A. Nemirovski, *Polynomial time barrier methods in convex programming*, Ekonomika i Matem. Metody 24 (1988), 1084–1091
[56] Y. Nesterov, A. Nemirovski, *Self-concordant functions and polynomial time methods in convex programming*, USSR Acad. Sci., Central Economic&Mathematical Institute, Moscow, 1989
[57] Y. Nesterov, A. Nemirovski, *Optimization over positive semidefinite matrices*, USSR Acad. Sci., Central Economic&Mathematical Institute, Moscow, 1990
[58] Y. Nesterov, A. Nemirovski, *Conic formulation of a convex programming problem and duality*, USSR Acad. Sci., Central Economic&Mathematical Institute, Moscow, 1991
[59] Y. Nesterov, A. Nemirovski, *Interior-Point Polynomial Methods in Convex Programming*, Studies in Apllied Mathematics, vol. 13, SIAM, Philadelphia, 1994
[60] C. H. Papadimitrou, K. Steiglitz, *Combinatorial Optimization; Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey, 1982
[61] S. Petrović, *Combinatorial algorithms and heuristics for clustering vertices of a graph and applications to pattern recognition*, (in Serbian), Master Thesis, University of Belgrade, Faculty of Electrical Engineering, Beograd, 1988
[62] R. C. Read, D. R. Corneil, *The graph isomorphism disease*, J. Graph Theory 1 (1977), 339–363
[63] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, 1970
[64] V. G. Timkovski, *Some recognition problems that are connected with the isomorphism of graphs*, (in Russian), Kibernetika (Kiev), 1988, no. 2, 13-18, 133; translated in Cybernetics 24:2 (1988), 153–160

[65] L. Vandenberghe, S. Boyd, *Semidefinite programming*, SIAM Review 38 (1996), 49–95
[66] Y. C. de Verdiére, *Sur un novel invariant des graphes et un critére planarité*, J. Combinatorial Theory B 50 (1990), 11–21
[67] H. Wolkowicz, *Some applications of optimization in matrix theory*, Linear Algebra Appl. 40 (1981), 101–118
[68] H. Wolkowicz, R. Saigal, L. Vandenberghe, *Handbook of Semidefinite Programming; Theory, Algorithms, and Applications*, Kluwer Academic Publishers, Boston-Dordrecht-London, 2000
[69] T. Živković, *On graph isomorphism and graph automorphism*, J. Math. Chem. 8:1-3 (1991), 19–37