Research Article

A Convex Relaxation Bound for Subgraph Isomorphism

Christian Schellewald

Independent Research, 7052 Trondheim, Tyholtveien 68, Norway

Correspondence should be addressed to Christian Schellewald, christian.schellewald@gmail.com

In this work a convex relaxation of a subgraph isomorphism problem is proposed, which leads to a new lower bound that can provide a proof that a subgraph isomorphism between two graphs can not be found. The bound is based on a semidefinite programming relaxation of a combinatorial optimisation formulation for subgraph isomorphism and is explained in detail. We consider subgraph isomorphism problem instances of simple graphs which means that only the structural information of the two graphs is exploited and other information that might be available (e.g., node positions) is ignored. The bound is based on the fact that a subgraph isomorphism always leads to zero as lowest possible optimal objective value in the combinatorial problem formulation. Therefore, for problem instances with a lower bound that is larger than zero this represents a proof that a subgraph isomorphism can not exist. But note that conversely, a negative lower bound does not imply that a subgraph isomorphism must be present and only indicates that a subgraph isomorphism can not be excluded. In addition, the relation of our approach and the reformulation of the largest common subgraph problem into a maximum clique problem is discussed.

1. Introduction

The subgraph isomorphism problem is a well-known combinatorial optimization problem and often involves the problem of finding the appropriate matching too. It is also of particular interest in computer vision where it can be exploited to recognise objects. For example, if an object in an image is represented by a graph, the object could be identified as subgraph within a possibly larger scene graph. Several approaches have been proposed to tackle the subgraph isomorphism problem and we refer to a few [1–4] and their references therein.

Error-correcting graph matching [5]—also known as *error-tolerant* graph matching—is a general approach to calculate an assignment between the nodes of two graphs. It is based on the minimisation of *graph edit costs* which result from some predefined edit operations when one graph is turned exactly into the other. Commonly introduced graph edit operations are deletion, insertion, and substitution of nodes and edges. Each graph-edit operation has a cost

assigned to it which is application dependent. The minimal graph edit cost defines the so-called *edit distance* between two graphs. The idea to define such a distance for graph matching goes back to Sanfeliu and Fu [6] in 1983. Before that, the edit distance was mainly used for string matching. Several approaches for error correcting graph matching have been proposed that are based on different methods like tree search [3], genetic algorithms [7], and others (see e.g., [5]) to name a few. In this paper we propose and apply a semidefinite programming (SDP) relaxation to a quadratic optimization formulation for subgraph isomorphism. In combinatorial optimization, semidefinite programming represents a valuable approach to approximate NP-hard problems and to obtain bounds for these problems. The current increase of interest in semidefinite programming is largely driven by the successful extension of interior point algorithms for linear programming to semidefinite programming (see e.g., [8]). A well-known example for semidefinite programming is the MAX-CUT approximation by Goemans and Williamson [9]. With the increase of the computational power of computers, SDP turned out to be useful for an increasing range of real world problems. For example, it was applied to several problems in the field of computer vision including segmentation/partitioning, grouping, restoration [10] matching [11], graph seriation [12], and camera calibration [13].

2. Contribution and Aim of the Paper

The main contribution of this paper lies in the convex relaxation of a subgraph isomorphism problem and the identification of a lower bound for this optimization problem. The computation of that bound is based on the SDP approximation of a combinatorial optimization formulation for subgraph isomorphism. The combinatorial optimization formulation and its convex relaxation is explained in detail. The approach is designed to find a subgraph isomorphism which maps the entire node-set of the possibly smaller graph to a subset of nodes in the second graph. We also discuss an interesting relation to an approach that is based on a reformulation of the largest common subgraph problem into a largest clique problem [14, 15].

3. Organisation of the Paper

After providing the notation we use, we introduce a combinatorial quadratic optimization formulation for the subgraph isomorphism problem that can be interpreted as an error-correcting graph matching approach.

   The integer optimization problem we end up with is generally an indefinite quadratic integer optimization problem which is known to be NP-hard [16] as Pardalos and Vavasis showed that indefinite quadratic programs are NP-hard problems (see [17]). Then we explain in detail a convex SDP relaxation of the combinatorial problem that leads to a lower bound for the subgraph isomorphism problem. The bound can be computed with standard methods for semidefinite programs (see, e.g., [18–20]). Finally, our experiments show that the bound can be tight enough to prove that no subgraph isomorphism between two graphs can be found.

4. Preliminaries

In this paper, we consider simple graphs $G = (V, E)$ with nodes $V = \{1, \ldots, n\}$ and edges $E \subset V \times V$. We denote the first (possibly smaller) graph with G_K and the second graph with

G_L. The corresponding sets V_K and V_L contain $K = |V_K|$ and $L = |V_L|$ nodes, respectively. We assume that $L \geq K$, which is in fact no constraint as we can always choose G_L to represent the larger graph. We make extensive use of the direct product $C = A \otimes B$, which is also known as Kronecker product [21]. It is the product of every matrix element A_{ij} of $A \in \mathrm{IR}^{n \times m}$ with the whole matrix $B \in \mathrm{IR}^{p \times q}$ resulting in the larger matrix $C \in \mathrm{IR}^{np \times mq}$.

A subgraph isomorphism is a mapping $m : V_K \mapsto V \subset V_L$ of all nodes in the graph G_K to a subset V of V_L with K nodes of the graph G_L such that the structure is preserved. That means that any two nodes i and j from G_K that are adjacent must be mapped to nodes $m(i)$ and $m(j)$ in G_L that are adjacent too. If the nodes i and j in G_K are not adjacent they must be mapped to nonadjacent nodes in G_L. The same has to be true for the inverse mapping $m^{-1} : V \mapsto V_K$ which maps the nodes V of the subgraph to the nodes V_K of G_K.

## 5. Combinatorial Objective Function

In this section we propose and show the correctness of a combinatorial problem formulation for finding a subgraph isomorphism. The general idea is to find a bipartite matching between the set of nodes from the smaller graph to a subset of nodes of the larger graph. The matching is evaluated by an objective function that can be interpreted as a comparison of the structure between all possible node pairs in the first graph and the structure of the node pairs (to which the nodes are matched) in the second graph. A matching that leads to no structural differences has no costs and represents a subgraph isomorphism. Mathematically, the evaluation can be formulated as a quadratic objective function $x^{\top} \tilde{Q} x$, where $x \in \mathrm{IR}^{KL}$ resents a mapping and $\tilde{Q} \in \mathrm{IR}^{KL \times KL}$ contains the problem data of the subgraph isomorphism problem. The full task of finding a subgraph isomorphism can be stated as the following combinatorial quadratic optimization problem, which details are explained below:

$$\min_{x} \quad x^{\top} \tilde{Q} x,$$

$$\text{s.t.} \quad A_K x = e_K, \quad A_L x \leq e_L, \quad x \in \{0,1\}^{KL}. \tag{5.1}$$

The constraints make use of the matrices $A_K = I_K \otimes e_L^{\top} \in \mathrm{IR}^{K \times KL}$ and $A_L = e_K^{\top} \otimes I_L \in \mathrm{IR}^{L \times KL}$ and ensure that the vector x is a binary 0,1-indicator vector which represents a bipartite matching between the two node sets of the graphs such that each node in graph G_K has a single partner in the graph G_L. Here $e_n \in \mathrm{IR}^n$ represents a vector with all elements 1. For our purposes, the elements of the indicator vector $x \in \{0,1\}^{KL}$ are arranged in the following order:

$$x = (x_{11}, \ldots, x_{L1}, x_{12}, \ldots, x_{L2}, \ldots, x_{1K}, \ldots, x_{LK})^{\top}. \tag{5.2}$$

Using double indices a nonzero vector element $x_{ji} = 1$ indicates that the node i of the first set of nodes V_K is matched to the node j in the second set V_L and otherwise $x_{ji} = 0$. We illustrate such an indicator vector in Figure 1 where a bipartite matching between two small
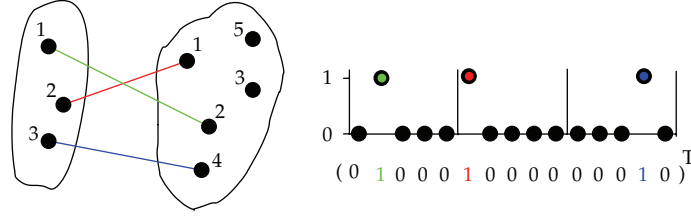
Figure 1: An illustration of the 0,1-indicator vector with $K = 3$ and $L = 5$. The right side shows the indicator vector representation of the particular bipartite matching which is shown on the left hand side of this figure. For each of the $K = 3$ mappings the vector contains exactly a single "1" within each of the K consecutive partitions with $L = 5$ elements.

sets of nodes and the corresponding indicator vector is shown. Note that for $K = 3$ and $L = 5$ the corresponding constraint matrices become

$$A_K = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

$$A_L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5.3}$$

The *relational structure matrix* $\tilde{Q}$ which contains the structural information of the graphs and appears within the objective function of the optimization problem (5.1) can be written in a form that exploits the Kronecker product.

Definition 5.1. Relational Structure Matrix:

$$\tilde{Q} = N_K \otimes \overline{N}_L + \overline{N}_K \otimes N_L. \tag{5.4}$$

Here N_K and N_L are the 0,1-adjacency matrices of the two graphs G_K and G_L. The matrices $\overline{N}_K$ and $\overline{N}_L$ represent the *complementary adjacency matrices* which are computed as the following.

Definition 5.2. Complementary Adjacency Matrices

$$\overline{N}_L = E_{LL} - N_L - I_L, \qquad \overline{N}_K = E_{KK} - N_K - I_K. \tag{5.5}$$

Here $E_{nn} \in \mathrm{IR}^{n \times n}$ are matrices with all elements equal to one and $I_n \in \mathrm{IR}^{n \times n}$ denotes the unit matrix. The complementary adjacency matrices have elements $(\overline{N})_{ij} = 1$ if the corresponding nodes i and j are not directly connected in the graph. To illustrate that, the adjacency matrix N_K for a small graph along with its complementary adjacency matrix are shown in **Figure 2**. There, the matrices are represented as a binary image (0 = black, 1 = white).

We show this as similar representations will be used to illustrate some particular matrices that appear later in this paper.
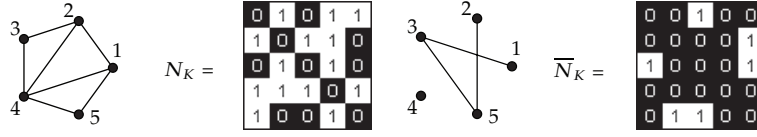
Figure 2: An example graph and its adjacency matrix N_K along with its complementary counterpart $\overline{N}_K$. The matrices are represented as an appropriate binary image (0 = black, 1 = white).

5.1. Subgraph Isomorphism

In the following, we show that a 0,1-solution vector x^* of the optimization problem (5.1) that has an optimal objective value of zero represents a subgraph isomorphism. The matrix $\widetilde{Q}$ is defined by (5.4). We first show that zero is the smallest possible value of the integer optimization problem. Then we show that every deviation from a subgraph isomorphism results in an objective value larger than 0, meaning that an objective value of zero represents a subgraph isomorphism.

Proposition 5.3. *The minimal value of the combinatorial optimization problem* (5.1) *is zero.*

Proof. The elements of $\widetilde{Q}$ and x are all nonnegative. In fact all their elements are either zero or one. Therefore, the lowest possible value of the quadratic cost term which can be rewritten as the following sum:

$$
\begin{aligned}
x^\top \widetilde{Q} x &= x^\top \left(N_K \otimes \overline{N}_L + \overline{N}_K \otimes N_L \right) x \\
&= \sum_{a,r}^{K,L} \sum_{b,s}^{K,L} \underbrace{\left[(N_K)_{ab} \left(\overline{N}_L \right)_{rs} + \left(\overline{N}_K \right)_{ab} (N_L)_{rs} \right]}_{\widetilde{Q}_{ra,sb}} x_{ra} x_{sb}
\end{aligned}
\tag{5.6}
$$

is zero. □

Proposition 5.4. *A solution with the minimal value of zero of the quadratic optimization problem* (5.1) *represents a subgraph isomorphism.*

In order to prove this we look closer at the term $\widetilde{Q}_{ra,sb} x_{ra} x_{sb}$ within the sum of (5.6) and show that it leads to a cost larger than 0 only if the considered matching violates the condition for a subgraph isomorphism.

Proof. Due to the constraints, x represents a bipartite matching and only if the product $x_{ra} x_{sb}$ is equal to one can the term within the sum (5.6) be different from zero, and $\widetilde{Q}_{ra,sb} = [(N_K)_{ab}(\overline{N}_L)_{rs} + (\overline{N}_K)_{ab}(N_L)_{rs}]$ has to be considered in detail. We refer to $\widetilde{Q}_{ra,sb}$ also as *structure comparison term*. There are the following two cases that lead to $x_{ra} x_{sb} = 1$ in the sum (5.6).

Case A. The node a and node b refer to the same node in G_K meaning that $a = b$. As the diagonals of N_K and $\overline{N}_K$ are zero, one finds that $(N_K)_{aa} = 0$ and $(\overline{N}_K)_{aa} = 0$. Then the term $[(N_K)_{aa}(\overline{N}_L)_{rs} + (\overline{N}_K)_{aa}(N_L)_{rs}]x_{ra}x_{sa}$ is always equal to zero and does not contribute to the sum.

Table 1: List of all outcomes of the *structure comparison term* when two different nodes a and b of graph G_K are mapped to two different nodes r and s in the second graph G_L. The first column describes the relation between the concerned nodes and the last column shows the associated cost $= [(N_K)_{ab}(\overline{N}_L)_{rs} + (\overline{N}_K)_{ab}(N_L)_{rs}]$. Only in cases I and IV is the structure preserved and can lead to an isomorphism. No cost is added in this cases. The other cases (II and III) do not preserve the structure and result in a total cost larger than 0. For details see the text.

Node configurations	$(N_K)_{ab}$	$(\overline{N}_L)_{rs}$	$(\overline{N}_K)_{ab}$	$(N_L)_{rs}$	Cost
I: a,b adjacent; r,s adjacent	1	0	0	1	0
II: a,b adjacent; r,s not adjacent	1	1	0	0	1
III: a,b not adjacent; r,s adjacent	0	0	1	1	1
IV: a,b not adjacent; r,s not adjacent	0	1	1	0	0

Case B. The nodes a and b in G_K refer to different nodes in G_K ($a \neq b$). Due to the bipartite matching constraint, a value $x_{ra}x_{sb} = 1$ represents the situation $x_{ra} = 1$ and $x_{sb} = 1$, where the nodes a and b in G_K are mapped to two different nodes, r and s, in the second graph G_L respectively. Then four possible cases for the structure comparison term $[(N_K)_{ab}(\overline{N}_L)_{rs} + (\overline{N}_K)_{ab}(N_L)_{rs}]$ exist which result in a cost of either zero or one. These subcases are considered separately below.

The subcases we have to consider in B include all four possible structural configurations between the two pairs of nodes (a,b and r,s) and are listed in Table 1. The two mappings which do not preserve the structure, therefore violating the isomorphism conditions, result in a cost of one. However, these four cases (I–IV) are described in more detail below and we will see that a cost is added for every mapping that results in a difference between the structure of graph G_K and the considered subgraph of the second graph G_L.

(I) If the two nodes a and b in the first graph G_K are neighbours, $(N_K)_{ab} = 1$, then no cost is added in (5.6) if the nodes r and s in the second graph are neighbours too: $(\overline{N}_L)_{rs} = 0$.

(II) Otherwise, if a and b are neighbours in G_K, $(N_K)_{ab} = 1$, and the corresponding nodes r and s are no direct neighbours in the second graph, $(\overline{N}_L)_{rs} = 1$, then a cost of 1 is added.

The mappings which correspond to configuration case I and II are visualised in Figure 3.

(III) Analogously, the structure comparison term penalises assignments where pairs of nodes (a and b) in the graph G_K become neighbours (r and s') in the second graph G_L which were not adjacent before.

(IV) Finally, if a and b are not adjacent in the first graph G_K and the nodes r and s in G_L are also not adjacent, no cost is added.

Figure 4 illustrates the situations III and IV in detail.

The itemisation of these four possible cases shows that only mappings that lead to a change in the structure are penalised with a cost. Structure preserving mappings which are compatible with a subgraph isomorphism are without costs and if all mappings are structure preserving it represents a subgraph isomorphism.

(a) I: Good assignment (no costs)   (b) II: Bad assignment (costly)
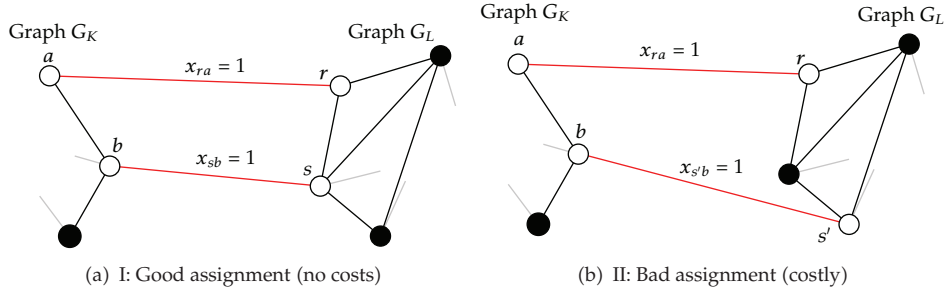
Figure 3: (a) (Case I) adjacent nodes a and b in the graph G_K are assigned to adjacent nodes r and s in the graph G_L. (b) (Case II) adjacent nodes a and b are no longer adjacent in the graph G_L after the assignment. The left mapping leads to no additional costs while the right undesired mapping adds 1 to the total cost.

(a) III: Bad assignment (costly)   (b) IV: Good assignment (no costs)
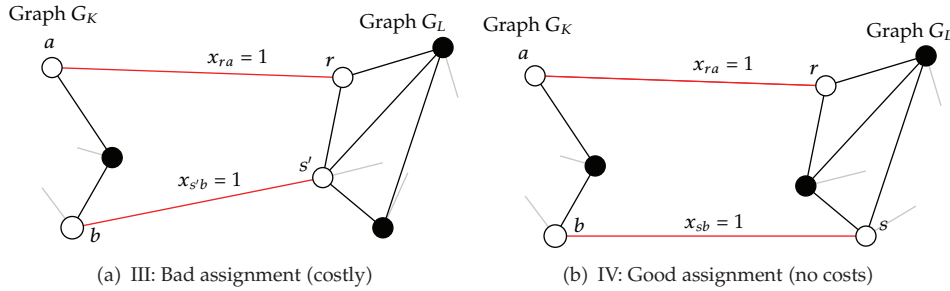
Figure 4: (a) (Case III) a pair of nodes a and b becomes neighbours r and s' after the assignment. This undesired assignment adds 1 to the costs. (b) (Case IV) nodes a and b which are not adjacent in the object graph G_K are assigned to nodes which are also not adjacent in the scene graph G_L. This assignment is associated with no additional cost in (5.6).

Note that due to the symmetry of the adjacency matrices the quadratic cost term $x^\top \widetilde{Q} x$ is symmetric and every difference in the compared structures of the two graphs is considered twice, resulting actually in a cost of 2 for every difference in the structure.

Finally, the sum (5.6) and therefore the objective function $x^\top \widetilde{Q} x$ evaluate the full matching encoded in x. And only for matchings which lead to no difference in the mapped substructure—and vice versa—all the terms within the sum (5.6) are zero. In this case, the bipartite matching, represented by the vector x, is a subgraph isomorphism. $\qquad\square$

We wish to emphasise that the minimisation of (5.1) represents the search for a bipartite matching which has the smallest possible structural deviation between G_K and the considered subgraph of G_L. The optimization problem (5.1) can therefore be seen as a *graph edit distance* with a cost of 2 for each addition or removal of an edge that is needed to turn the first graph G_K into the considered subgraph of the second graph G_L.

6. Example Problem

In the next section we discuss in detail how we obtain the semidefinite relaxation of (5.1). In order to ease the reproduction of our approach, we illustrate some occurring details based on the particular subgraph isomorphism problem that is shown in Figure 5. We will also be
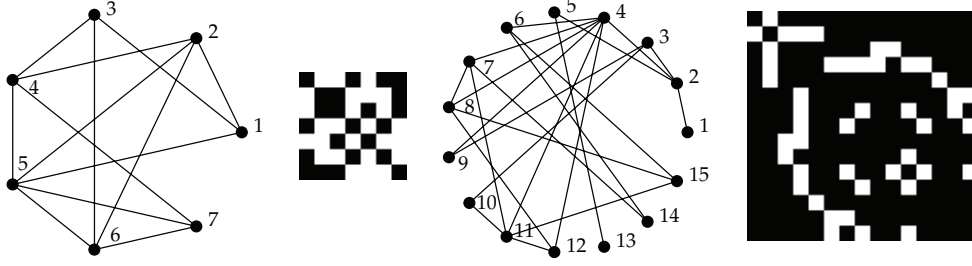
Figure 5: A randomly created subgraph problem instance with $K = 7$ and $L = 15$. The two graphs are shown along with their binary adjacency matrices. Is there a subgraph isomorphism? For the shown problem instance we can compute a lower bound larger than 0, which proves that no subgraph isomorphism is present.

able to conclude from a lower bound larger than 0 that for this problem instance no subgraph isomorphism can be found.

7. Convex Problem Relaxation

In the following we explain the convex relaxation of the combinatorial isomorphism approach (5.1). It will be relaxed to a (convex) semidefinite program (SDP) which has the following standard form:

$$
\begin{aligned}
L_B = \min, \quad & \mathrm{Tr}[QX], \\
\text{s.t.} \quad & \mathrm{Tr}[A_i X] = c_i, \quad \text{for } i = 1, \ldots, m, \ X \succeq 0.
\end{aligned}
\tag{7.1}
$$

The constraint $X \succeq 0$ means that the matrix X has to be positive semidefinite. This convex optimization problem can be solved with standard methods like interior point algorithms (see, e.g., [22]). Note that the solution of the convex relaxation (7.1) provides a lower bound L_B to the combinatorial optimization problem (5.1). A lower bound with a value $L_B > 0$ shows that no subgraph isomorphism can be present as the objective value of (5.1), for instance with subgraph isomorphism is zero. Below, we describe in detail how we derive such a semidefinite program from (5.1).

7.1. SDP Objective Function

In order to obtain an appropriate SDP relaxation for the combinatorial subgraph matching problem, we start with the reformulation of the objective function of (5.1)

$$
f(x) = x^\top \widetilde{Q} x = \mathrm{Tr}\left[x^\top \widetilde{Q} x\right] = \mathrm{Tr}\left[\widetilde{Q} x x^\top\right] = \mathrm{Tr}\left[\widetilde{Q} \widetilde{X}\right].
\tag{7.2}
$$

Here we exploited the invariance of the trace operator under cyclic exchange. Besides being symmetric, the matrix $\widetilde{X} = x x^\top$ is positive semidefinite and has rank 1. The objective function is relaxed by dropping the rank 1 condition of $\widetilde{X}$ and leaving the constraint $\widetilde{X} \succeq 0$ untouched. This makes the set of feasible matrices convex (see, e.g., [8]) and lifts the original
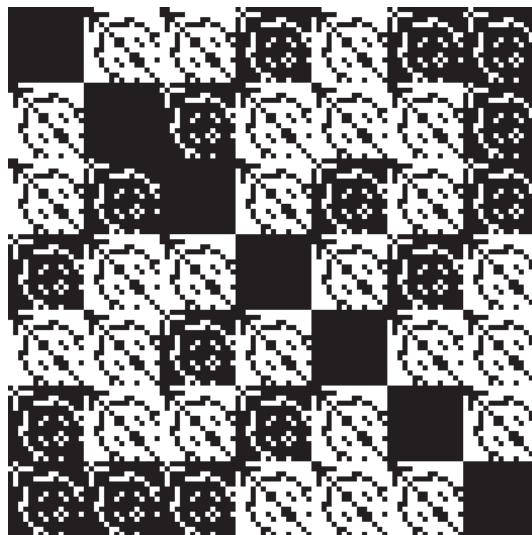
Figure 6: Here the relational structure matrix $\widetilde{Q} = N_K \otimes \overline{N}_L + \overline{N}_K \otimes N_L$ for the example subgraph isomorphism problem that is shown in Figure 5 is depicted.

problem (5.1) defined in a vector space with dimension KL into the space of symmetric, positive semidefinite matrices with the dimension $KL \times KL$. We note that the diagonal elements of the relaxed $\widetilde{X}$ reflect the approximation of the vector x which we are searching for as $x_i^2 = x_i$ is true for 0,1-binary values. We further remark that the computed lower bound L_B in (7.1) can be negative although $\widetilde{X}$ is constraint to be positive semidefinite.

The relational structure matrix $\widetilde{Q}$ is a binary matrix and the particular matrix which represents the example subgraph isomorphism problem introduced in Section 6 is depicted in Figure 6. One easily detect the patterns resulting from the Kronecker product formulation in (5.4). On the fine scale, the pattern of the adjacency matrix of the graph G_L can be seen. The adjacency matrix of the graph G_K is present on a larger scale (compare the adjacency matrices in Figure 5).

7.2. SDP Constraints

In the convex relaxation (7.1), we have to incorporate constraints in the form $\mathrm{Tr}[A_i X] = c_i$, where A_i and X are matrices and $c_i \in \mathrm{IR}$ is a scalar value. The aim is to approximate the original constraints as good as possible in this way. Below we describe four different types (a–d) of constraint matrices A_i along with their corresponding values for c_i that we used to tighten the relaxation and to approximate the bipartite matching constraints within the standard SDP formulation (7.1).

(a) In order to get a tight relaxation, we exploit the fact that $x_i = x_i^2$ holds true for 0,1-variables by introducing an additional first row and column in the problem matrix resulting in matrices $Q \in \mathrm{IR}^{(KL+1) \times (KL+1)}$ and $X \in \mathrm{IR}^{(KL+1) \times (KL+1)}$ with the dimension increased by one compared to $\widetilde{Q}$ and $\widetilde{X}$. In Q the additional elements were set to zero in order not to change the value of the objective function. The elements of the new row and the new column in X are forced to be equal to the corresponding diagonal elements by using the following KL

constraint matrices $^{\text{int}}A^j \in \text{IR}^{(KL+1)\times(KL+1)}$, $j = 2,\ldots,KL + 1$ which are elements that can be defined by:

$$^{\text{int}}A^j_{kl} = 2\delta_{kj}\delta_{lj} - \delta_{kj}\delta_{l1} - \delta_{lj}\delta_{k1} \quad \text{for } k,l = 1,\ldots,KL + 1. \tag{7.3}$$

Here we make use of the Kronecker delta δ_{ij} which is one if $i = j$ and zero otherwise. Note that the Kronecker delta representation allows for an easy creation of the matrices in any computer programming language. Here each constraint matrix $^{\text{int}}A^j$ has only 3 nonzero elements. The corresponding constraint variables $^{\text{int}}c^j$ are all zero.

(b) We restrict the first element in the matrix X to $X_{11} = 1$, using a single additional constraint matrix $^{\text{one}}A \in \text{IR}^{(KL+1)\times(KL+1)}$ whose elements can be expressed as

$$^{\text{one}}A_{kl} = \delta_{k1}\delta_{l1} \quad \text{for } k,l = 1,\ldots,KL + 1. \tag{7.4}$$

The matrix $^{\text{one}}A$ has only $^{\text{one}}A_{11} = 1$ as nonzero element and the corresponding constraint variable $^{\text{one}}c$ is 1.

(c) The equality constraints $\sum_{j=1}^{L} x_{ij} = 1$, $i = 1,\ldots,K$, which are part of the bipartite matching constraints $(A_K x = e_K)$ represent the constraint that each node of the smaller graph is mapped to exactly one node of the scene graph. To model these, we define K constraint matrices $^{\text{sum}}A^j \in \text{IR}^{(KL+1)\times(KL+1)}$, $j = 1,\ldots,K$, which ensure (taking the chosen order of the elements into account) that the sum of the appropriate partition of the diagonal elements in X is 1. Note that we operate on the diagonal elements of X as they reflect the vector x. The matrix elements for the jth constraint matrix $^{\text{sum}}A^j$ can be expressed as follows:

$$^{\text{sum}}A^j_{kl} = \sum_{i=(j-1)L+1}^{jL+1} \delta_{ik}\delta_{il} \quad \text{for } k,l = 1,\ldots,KL + 1. \tag{7.5}$$

This defines K matrices where the appropriate partition with L consecutive elements on the diagonal are set to one. All other elements are zero. For these constraints the corresponding values of $^{\text{sum}}c_j$ are all 1.

(d) All integer solutions $\tilde{X} = xx^\top \in \text{IR}^{KL\times KL}$, where x represents a bipartite matching, have zero values at those matrix elements where the matrix $\tilde{Z} = I_K \otimes (E_{LL} - I_L) + (E_{KK} - I_K) \otimes I_L$ has nonzero elements. In order to approximate the bipartite matching constraints we want to force the corresponding elements in (the enlarged) matrix $X \in \text{IR}^{(KL+1)\times(KL+1)}$ to be zero. The matrices $E_{nn} \in \text{IR}^{n\times n}$ have all elements 1 and $I_n \in \text{IR}^{n\times n}$ represent the unit matrices. We can force the elements to be zero with the help of the constraint matrices which we denote A^{ars}, $A^{\hat{s}\hat{a}\hat{b}} \in \text{IR}^{(KL+1)\times(KL+1)}$ and that are determined by the indices a, r, s and $\hat{s}, \hat{a}, \hat{b}$. They have the following matrix elements:

$$\begin{aligned} A^{ars}_{kl} &= \delta_{k,(aL+r+1)}\delta_{l,(aL+s+1)} + \delta_{k,(aL+s+1)}\delta_{l,(aL+r+1)}, \\ A^{\hat{s}\hat{a}\hat{b}}_{kl} &= \delta_{k,(\hat{s}K+\hat{b}+1)}\delta_{l,(\hat{s}K+\hat{a}+1)} + \delta_{k,(\hat{s}K+\hat{a}+1)}\delta_{l,(\hat{s}K+\hat{b}+1)}, \end{aligned} \tag{7.6}$$

with $k,l = 1,\ldots,KL + 1$.
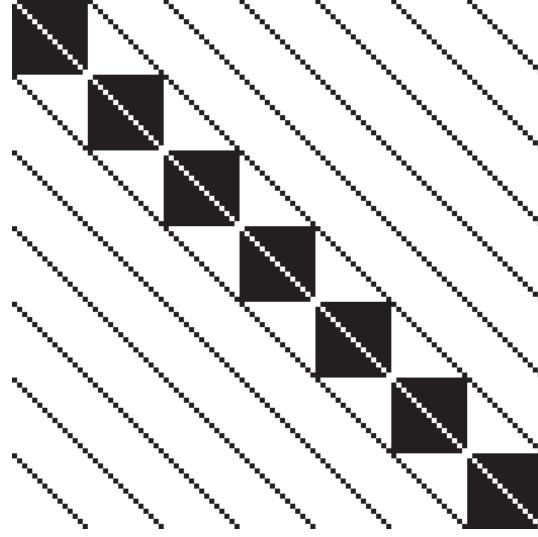
Figure 7: The shown matrix $\widetilde{Z} = ((E_{KK} - I_K) \otimes I_L + I_K \otimes (E_{LL} - I_L))$ indicates all matrix elements (here in black) that are forced to be zero within the solution matrix $\widetilde{X}$ for an example problem of the size $K = 7$, $L = 15$. A single constraint forces two symmetric elements in (the enlarged) matrix X to be zero.

The indices a, r, s and $\widehat{s}, \widehat{a}, \widehat{b}$ attain all valid combinations of the following triples where $s > r$ and $\widehat{b} > \widehat{a}$:

$$
\begin{aligned}
(a, r, s) &: a = 1, \ldots, K; \quad r = 1, \ldots, L; \quad s = (r+1), \ldots, L, \\
\left(\widehat{s}, \widehat{a}, \widehat{b}\right) &: \widehat{s} = 1, \ldots, L; \quad \widehat{a} = 1, \ldots, K; \quad \widehat{b} = (\widehat{a} + 1), \ldots, K.
\end{aligned}
\tag{7.7}
$$

Even if the number of indices is high the structure of a single matrix is fairly simple as every matrix has only two nonzero elements. For all these constraints the corresponding constants c have to be zero. With this we get $(LL-L)K/2 + (KK-K)L/2$ additional constraints that ensure zero values within the matrix X, where the corresponding elements in Z (enlarged by one dimension compared to $\widetilde{Z}$) have nonzero values. Therefore $\mathrm{Tr}[ZX] = 0$ is valid. Note that each constraint forces two elements in the solution matrix to be zero. The black pixels in the matrix shown in Figure 7 indicate all the elements that are forced to be zero for the example problem with $K = 7$ and $L = 15$.

Altogether this sums up to $(LL - L)K/2 + (KK - K)L/2 + KL + K + 1$ constraints of the form $\mathrm{Tr}[A_i X] = c_i$ within the convex semidefinite problem formulation (7.1). By solving the convex relaxation (7.1) we get a lower bound to the combinatorial optimization problem (5.1).

7.3. Computational Effort

The most computational effort within the SDP approach is needed for the computation of the solution of the SDP relaxation (7.1). We used external SDP solvers for this task. An independent benchmarking for several SDP solvers can be found in [23]. A comparison of three SDP solvers (DSDP [19], PENSDP [20] and CSDP [18]) on the basis of our own data is

Table 2: Mean computation time for three SDP solvers needed to solve the SDP relaxations of our experiments. The CSDP-solver fits best our needs.

Problem size (K/L)	CSDP 6.0.1	DSDP 5.8	PENSDP 2.2
7/15	(13.9 ± 1.2)s	(17.7 ± 1.9)s	(19 ± 2)s

shown in Table 2. There we compared the mean computation time needed to solve the SDP relaxations for random problems of the size $K = 7$ and $L = 15$. The comparison was made on a 2.8 GHz PC with 2 GB storage. We found that the CSDP-solver fits best our needs.

The CSDP-solver is a variant of the predictor corrector algorithm suggested by Helmberg et al. [24] and is described in detail in [18]. Below we sketch the storage requirement and the computational time for this solver.

7.4. Storage Requirement

In [25] the author points out that the CSDP-solver requires approximately

$$8\left(m^2 + 11\left(n_1^2 + n_2^2 + \cdots + n_s^2\right)\right) \tag{7.8}$$

bytes of storage. Here m is the number of constrains and $n_1^2, n_2^2, \ldots, n_s^2$ are the sizes of block diagonal matrices used to describe the $n \times n$ problem and solution matrices of the semidefinite program. In our case we have $n_1 = n_2 = \cdots = n_s = 1$ with $s = n$. In terms of the graph sizes K and L we have

$$n = KL + 1,$$
$$m = 1 + K + \frac{K^2 L}{2} + \frac{KL^2}{2}. \tag{7.9}$$

Using that, we compute that a subgraph matching problem instance with $K = 7$ and $L = 15$ needs about 10 MB while a problem instance with $K = 9$ and $L = 14$ already needs 17 MB of storage.

According to [18] the most difficult part in an iteration of the CSDP solver is the computation of the Cholesky factorisation of a $m \times m$ matrix which requires a time proportional to $O(m^3)$. This result is based on the here fulfilled assumption that individual constraint matrices have $O(1)$ nonzero elements and that m is somewhat larger than n.

8. Relation to the Maximum Clique Formulation

In this section we discuss the connections that can be drawn between our subgraph isomorphism approach and the maximum clique formulation for finding a maximum common subgraph isomorphism. Details about the maximum clique search in arbitrary graphs can be found for example in [14, 15, 26] and references therein. For our discussion we introduce and use the following abbreviations. We denote the adjacency matrix of the *association graph*, which is introduced to transform the maximum common subgraph problem

into a maximum clique problem, with $A = (N_K \otimes N_L + \overline{N}_K \otimes \overline{N}_L)$. This is equivalent to the elementwise expression

$$A_{ra,sb} = \begin{cases} 1 - ((N_K)_{ab} - (N_L)_{rs})^2, & \text{for} \quad a \neq b, \ r \neq s, \\ 0, & \text{otherwise}, \end{cases} \tag{8.1}$$

that is, for example, provided in [15]. This can be seen by rewriting the Kronecker product representation of A directly into the elementwise expression $A_{ra,sb} = [(N_K)_{ab}(N_L)_{rs} + (1 - (N_K)_{ab} - \delta_{ab})(1 - (N_L)_{rs} - \delta_{rs})]$ and exploiting that for 0,1-binary elements $N_{ij} = (N_{ij})^2$ is true. We use $Q = (N_K \otimes \overline{N}_L + \overline{N}_K \otimes N_L)$ for the structure matrix and $Z = ((E_{KK} - I_K) \otimes I_L + I_K \otimes (E_{LL} - I_L))$ for the indicator matrix, where the integer solution—holding the bipartite matching constraints—is expected to have zero values (compare also Figure 7). Furthermore, the unit matrix is denoted by $I = (I_K \otimes I_L)$ and $E = (E_{KK} \otimes E_{LL})$ is the matrix with all elements one. Note that all the involved matrices have a dimension of $KL \times KL$. Using these abbreviations one finds the following connection between the matrices involved in the maximum clique formulation (A and I) and our SDP-approach (Q and Z):

$$E = Q + Z + A + I. \tag{8.2}$$

Equation (8.2) shows explicitly the complementary nature of the matrices in the SDP approach and the maximum clique formulation. We can rewrite the combinatorial minimisation problem (5.1) as

$$\begin{aligned} &\min_x \quad x^\top Q x + x^\top Z x, \\ &\text{s.t.,} \quad A_K x = e_K, \quad A_L x \leq e_L, \quad x \in \{0,1\}^{KL}, \end{aligned} \tag{8.3}$$

where $x^\top Z x = 0$ is guaranteed by the bipartite matching constraint. Recall, that the 0,1-indicator-vector $x \in \mathrm{IR}^{KL}$ with length KL has K elements set to one ($e^\top x = K$), a single one within each consecutive segment of length L. Exploiting this, we obtain $x^\top E x = K^2$ and $x^\top I x = K$. Then using (8.2) we find

$$x^\top E x - x^\top I x = \left(K^2 - K \right) = x^\top (Q + Z) x + x^\top A x, \tag{8.4}$$

and due to the fixed value on the left-hand-side of (8.4) minimizing $x^\top (Q + Z) x$ is equivalent to maximising $x^\top A x$ with the same constraints for x. Therefore, maximising $x^\top A x$ holding the bipartite matching constraint is the search for a set of nodes with size K that deviates the least from a clique. In the maximum clique problem, that arises from the largest common subgraph problem, one is searching for a 0,1-vector x that indicates the nodes belonging to the maximum clique. Thereby making no further assumptions about the constraints. However, from the construction of the association graph from the largest common subgraph problem, one finds that the largest clique w that can be found in the association graph (in the presents of a full subgraph isomorphism) has $w = K$ members and that the obtained indicator vector will hold the bipartite matching constraint. That becomes clear as two nodes in the association graph that violate the bipartite matching constraints are not connected (the adjacency matrix

A has zero elements where Z has elements one.) and therefore cannot be part of the same clique. That means also, that the reformulation of the largest common subgraph problem into a maximum clique problem for graphs G_K and G_L with $K \leq L$ has a trivial upper bound of K for the largest clique size w. Using the SDP relaxation (7.1) and that $\operatorname{Tr}[ZX] = 0$ we obtain a lower bound L_B for $x^{\top}(Q + Z)x$ which we can use in (8.4) and get the following expression:

$$\left(1 - \frac{1}{K}\right) - \frac{L_B}{K^2} \geq \frac{x^{\top}Ax}{K^2}. \tag{8.5}$$

Knowing from the Motzkin Straus theorem [27] (or see, e.g., [26]) that $\max(x^{\top}Ax/K^2) = 1 - (1/K)$ holds exactly if there is a maximum clique with size K indicated by a binary vector x, we see that a positive lower bound $L_B > 0$ shows that a clique with K nodes in the association graph cannot be found (meaning also that no full subgraph isomorphism can be found in the original problem). Computing the following upper bounds for the clique-size of the (unconstraint) maximum clique search that were, for example, used in [26]:

$$w \leq \frac{3 + \sqrt{9 + 8(m - n)}}{2} = U_{B1}, \tag{8.6}$$

$$w \leq \rho(A) + 1 = U_{B2}, \tag{8.7}$$

$$w \leq N_{-1} + 1 = U_{B3}, \tag{8.8}$$

$$w \leq n - \frac{\operatorname{rank}\overline{A}}{2} = U_{B4}, \tag{8.9}$$

one finds that these are largely above the bound $w \leq K$ which is inherent in the reformulation of the largest common subgraph problem into a maximum clique problem. In the experiment section we will see that the bounds for the clique size are not even close to the trivial bound arising from the largest common subgraph problem reformulation. Of course they are still useful for the general maximum clique problem of arbitrary graphs. We used the same notation as the authors in [14] and [26]. Therefore, $\rho(A)$ is the spectral radius of the adjacency matrix A of the association graph, $n = KL$ is the number of nodes and m the number of edges in the association graph and $\delta = 2m/n^2$ is the density of ones in A. Furthermore, N_{-1} denotes the number of eigenvalues of A that are smaller or equal to minus one (-1) and $\overline{A} = E - A - I$ is the inverse/complementary adjacency matrix. Computing these bounds for our example subgraph matching problem (see Section 6), one obtains largest upper bounds for the clique size in the range from 35 to 61 (see also the bounds listed in Table 3) which is far above the trivial bound of 7. Note that the computed upper bounds approximately agree with the results for this density ($\delta = 0.346$) in [26]. However, the lower bound $L_B = 0.855 > 0$ proves that the maximum clique size is lower than $K = 7$ which accounts for a tight upper bound $w \leq 6$. As the clique-based upper bounds (7.1)–(8.9) are largely above the trivial bound $w \leq 7$ this suggests that additional knowledge in the clique formulation (namely the bipartite matching constraints) for this particular problem-formulation can and should be exploited to improve the upper bounds. We summarise our results/experiments in the next section.

Table 3: Bounds for the example problem. The largest clique that can be obtained in the related maximum clique problem is $w = 6$ and the lower bound $L_B = 0.855 > 0$ translates to the fact that the size of the maximum clique must be lower than $K = 7$, providing a tight upper bound (ubnd) $w \leq 6$.

K	$n = KL$	δ	w	L_B (7.1)	U_{B1} (8.6)	U_{B2} (8.7)	U_{B3} (8.8)	U_{B4} (8.9)
7	105	0.346	6	0.855	61.602	38.344	35	52.50

Table 4: Summarisation of the results. We performed 1000 ($N_{\text{Exp.}}$) random subgraph experiments from which 661 ($N_{\text{noIso.}}$) instances are known to have no subgraph isomorphism. The column denoted with N_{failed} shows how often the bound was not tight enough to detect that no subgraph isomorphism can be found. In $N_{\text{bnd}>0}$ cases the bound is tight enough. Without preprocessing the problem matrix size is $n \times n = (KL + 1) \times (KL + 1) = 106 \times 106$ while with the described pruning the size varies from problem instance to problem instance. However the mean problem matrix is reduced to about 69×69 ($\bar{n} \times \bar{n}$).

| Preproc.	K/L	$N_{\text{Exp.}}$	$N_{\text{noIso.}}$	$\bar{n} \times \bar{n}$	$N_{\text{bnd}>0}$	$N_{\text{bnd}\leq0}$	N_{failed}
Non	7/15	1000	661	106×106	123 ($\approx$19%)	877	538
Pruning	7/15	1000	661	69×69	327 ($\approx$49%)	673	334

9. Results to the Non-Isomorphism Bound

For the illustrative example (see Figure 5) we computed a lower bound $L_B = 0.855 > 0$ using the SDP relaxation (7.1), which proves that a subgraph isomorphism does not exist in this problem instance. The bound and optimal values along with the distribution of the objective values for that problem are shown in Figure 8. Note that the objective values of (5.1) that can be attained are restricted to discrete values as the quadratic term $\alpha x^\top \tilde{Q} x$ can only have values which are multiples of 2α. With $\alpha = 1.0$ the optimal objective value is 2.0. Note that we did not apply any preprocessing like the elimination of mappings that could not lead to a subgraph isomorphism for the illustrative example.

For a further investigation of the bound (7.1), we created 1000 (connected) random subgraph matching problem instances for which we have chosen the size of the two graphs (G_K and G_L) to be $K = 7$ and $L = 15$. The edge probability of the graph G_K was set to 0.5 and the probability for an edge in the second graph was set to 0.2. The ground truth for these experiments were computed using the VFlib [28].

The experiments reveal that for various problem instances, the relaxation is tight enough to conclude that no subgraph isomorphism can exist. We obtained 123 problem instances with a lower bound $L_B > 0$ which proves that no subgraph isomorphism can occur in these problem instances. The other 877 problem instances have a lower bound $L_B \leq 0$. From the ground truth data we know that for 538 of these problem instances the combinatorial optimum is larger than 0 indicating that for these cases the relaxation is not tight enough to detect that a subgraph isomorphism cannot be found. These results are summarised in Table 4 in the row labeled with "non-" preprocessing. The upper bounds for the equivalent maximum clique formulation of the same 1000 problems are all largely above the trivial bound of $K = 7$ that is inherent in the problem formulation (see Table 5) and in no case this allows to prove that no subgraph isomorphism can occur.

In the next section we investigate the improvements to the bound computation when mappings are removed from the problem formulation that cannot lead to a subgraph isomorphism.
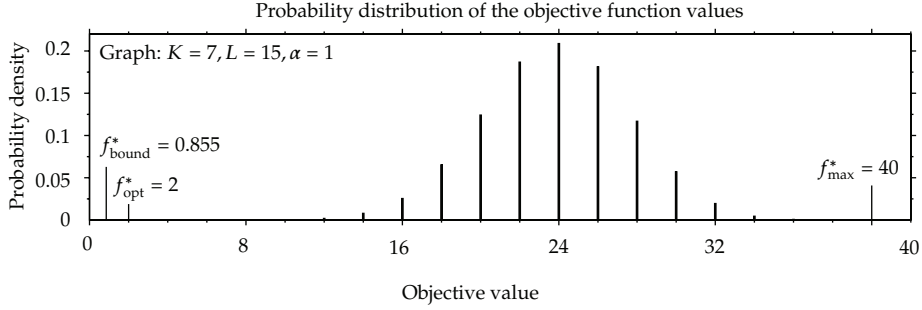
Figure 8: The distribution of the objective values for the subgraph isomorphism problem which is shown in Figure 5. The objective values are restricted to discrete values, as the quadratic term $\alpha x^\top \tilde{Q} x$ can only attain values which are multiples of 2α. With $\alpha = 1.0$ the optimal objective value is 2.0 and the obtained lower bound is 0.855 > 0, which is a nonisomorphism proof for this problem instance.

Table 5: None of the upper bounds for the maximum clique size comes close to the trivial bound $K = 7$ that is inherent in the reformulation of the largest common subgraph problem into the maximum clique problem. The clique bounds are computed for the equivalent maximum clique formulation of the same 1000 problems used in Table 4.

Bound	K/L	Mean bound	Min.	Max
U_{B1}	7/15	66.00 ± 3.32	53.31	74.55
U_{B2}	7/15	44.06 ± 3.88	30.69	54.34
U_{B3}	7/15	40.09 ± 2.80	32	54
U_{B4}	7/15	52.55 ± 0.19	52.50	54.50

9.1. Towards Larger Problem Instances

We implemented a simple pruning technique to reduce the dimension of the SDP problem size. The basic idea of this is to eliminate all mappings $i \mapsto j$ for which the degree (the number of incident edges) of a node i in the first graph is larger than the degree of node j in the second graph. Such a mapping cannot lead to a subgraph isomorphism. The computational advantage is that every removed mapping reduces the dimension of the problem matrix by one ($n \times n \mapsto n - 1 \times n - 1$) and also allows to remove the corresponding constraints from the semidefinite problem. A feasibility test then checks whether the remaining mapping opportunities can still lead to a bipartite matching such that all nodes of the smaller graph can be mapped to different nodes in the larger graph. Note that such an infeasible situation might also results in an infeasible semidefinite problem formulation. For this pruning approach one has to keep in mind that the *new bound* of the remaining problem does not necessarily represent a lower bound of the original problem. This is because in the case of a nonisomorphism a removed mapping could be part of the matching which belongs to the global optimum of the original problem. In fact the combinatorial optimum for the remaining problem can only increase (or stay the same) and the computed bound is a lower bound to the new problem only. However, for problem cases with an isomorphism the optimum does not change (it is still zero) and therefore a bound $L_B > 0$ of the new problem still proves that an isomorphism cannot exist.

Applying the above-described technique to the problem instances in the previous section, the size of the SDP problem matrices reduces from 106×106 ($KL + 1 \times KL + 1$) to

69×69 in the mean. The number of cases where the relaxation is not tight enough improved from 538 to 334 out of 661 cases with a combinatorial optimum larger than 0 (see also the second row in Table 4).

We expect that the bound could be slightly tightened by including also the L inequalities in the SDP relaxation that are currently not considered. However, first findings indicate that the influence might be negligible which could be caused due to the fact that they are already modeled by the incorporated constraints. Another computational improvement within the SDP algorithms could result from an exploitation of the highly regular structure within the SDP matrix.

10. Discussion

In this paper we proposed a convex relaxation bound to the subgraph isomorphism problem and showed that the bound is not only of theoretical interest but also applies to several instances of subgraph matching problems. It would be interesting to investigate which criteria a subgraph matching problem has to fulfill to result in a tight relaxation. Such insights could be useful in the process of creating or obtaining object graphs from images for object recognition tasks. At the current stage, reasonable tight bounds result from semidefinite problems with a problem matrix size of up to 200×200 elements. An improvement of the proposed method could be expected when also the inequalities are included in the SDP relaxation. However, for increasing problem instances the relaxation will get less tight and a lower bound not larger than zero becomes more likely. But note that even less tight solutions in practice still lead to good integer solutions (see, e.g., [11]).

Acknowledgments

References

[1] J. R. Ullmann, "An algorithm for subgraph isomorphism," *Journal of the Association for Computing Machinery*, vol. 23, no. 1, pp. 31–42, 1976.

[2] H. G. Barrow and R. M. Burstall, "Subgraph isomorphism, matching relational structures and maximal cliques," *Information Processing Letters*, vol. 4, no. 4, pp. 83–84, 1976.

[3] B. T. Messmer and H. Bunke, "A new algorithm for error-tolerant subgraph isomorphism detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 5, pp. 493–504, 1998.

[4] D. Eppstein, "Subgraph isomorphism in planar graphs and related problems," *Journal of Graph Algorithms and Applications*, vol. 3, no. 3, pp. 1–27, 1999.

[5] H. Bunke, "Error correcting graph matching: on the influence of the underlying cost function," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 9, pp. 917–922, 1999.

[6] A. Sanfeliu and K. S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 13, no. 3, pp. 353–362, 1983.

[7] Y.-K. Wang, K.-C. Fan, and J.-T. Horng, "Genetic-based search for error-correcting graph isomorphism," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 27, no. 4, pp. 588–597, 1997.

[8] H. Wolkowicz, R. Saigal, and L. Vandenberghe, Eds., *Handbook of Semidefinite Programming*, Kluwer Academic Publishers, Boston, Mass, USA, 2000.

[9] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the Association for Computing Machinery*, vol. 42, no. 6, pp. 1115–1145, 1995.

[10] J. Keuchel, C. Schnörr, C. Schellewald, and D. Cremers, "Binary partitioning, perceptual grouping, and restoration with semidefinite programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 11, pp. 1364–1379, 2003.

[11] C. Schellewald and C. Schnörr, "Probabilistic subgraph matching based on convex relaxation," in *Proceedings of the 5th International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR '05)*, vol. 3757 of *Lecture Notes in Computer Science*, pp. 171–186, 2005.

[12] H. Yu and E. R. Hancock, "Graph seriation using semi-definite programming," in *Proceedings of the 5th IAPR International Workshop on Graph-Based Representations in Pattern Recognition (GbRPR '05)*, vol. 3434 of *Lecture Notes in Computer Science*, pp. 63–71, 2005.

[13] M. Agrawal and L. S. Davis, "Camera calibration using spheres: a semi-definite programming approach," in *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV '03)*, vol. 2, pp. 782–789, 2003.

[14] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo, "The maximum clique problem," in *Handbook of Combinatorial Optimization*, D.-Z. Du and P. M. Pardalos, Eds., pp. 1–74, Kluwer Academic, Boston, Mass, USA, 1999.

[15] M. Pelillo, "Replicator equations, maximal cliques, and graph isomorphism," *Neural Computation*, vol. 11, no. 8, pp. 1933–1955, 1999.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, Calif, USA, 1991.

[17] P. M. Pardalos and S. A. Vavasis, "Quadratic programming with one negative eigenvalue is NP-hard," *Journal of Global Optimization*, vol. 1, no. 1, pp. 15–22, 1991.

[18] B. Borchers, "CSDP, a C library for semidefinite programming," *Optimization Methods and Software*, vol. 11, no. 1, pp. 613–623, 1999.

[19] S. J. Benson and Y. Ye, "DSDP3: dual scaling algorithm for general positive semidefinite programming," Tech. Rep. ANL/MCS-P851-1000, Argonne National Labs, 2001.

[20] M. Kočvara and M. Stingl, "Pennon: a code for convex nonlinear and semidefinite programming," *Optimization Methods & Software*, vol. 18, no. 3, pp. 317–333, 2003.

[21] A. Graham, *Kronecker Products and Matrix Calculus with Applications*, Ellis Horwood and John Wiley & Sons, 1981.

[22] Y. Ye, *Interior Point Algorithms: Theory and Analysis*, John Wiley & Sons Inc., New York, NY, USA, 1997.

[23] H. D. Mittelmann, "An independent benchmarking of SDP and SOCP solvers," *Mathematical Programming Series B*, vol. 95, no. 2, pp. 407–430, 2003.

[24] C. Helmberg, F. Rendl, R. J. Vanderbei, and H. Wolkowicz, "An interior-point method for semidefinite programming," *SIAM Journal on Optimization*, vol. 6, no. 2, pp. 342–361, 1996.

[25] Brian Borchers. CSDP 4.8 User's Guide, 2004.

[26] M. Budinich, "Exact bounds on the order of the maximum clique of a graph," *Discrete Applied Mathematic*, vol. 127, no. 3, pp. 535–543, 2003.

[27] T. S. Motzkin and E. G. Straus, "Maxima for graphs and a new proof of a theorem of Turán," *Canadian Journal of Mathematics. Journal Canadien de Mathématiques*, vol. 17, pp. 533–540, 1965.

[28] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "Performance evaluation of the vf graph matching algorithm," in *Proceedings of the 10th International Conference on Image Analysis and Processing (ICIAP '99)*, p. 1172, IEEE Computer Society, Washington, DC, USA, 1999.