Minimum cost subgraph matching using a binary linear program

Julien Lerouge, Maroua Hammami, Pierre Héroux, Sébastien Adam

Minimum Cost Subgraph Matching Using a Binary Linear Program

Julien Lerouge   Maroua Hammami   Pierre Héroux
Sébastien Adam

Abstract

This article presents a binary linear program for the Minimum Cost Subgraph Matching (MCSM) problem. MCSM is an extension of the subgraph isomorphism problem where the matching tolerates substitutions of attributes and modifications of the graph structure. The objective function proposed in the formulation can take into account rich attributes (e.g. vectors mixing nominal and numerical values) on both vertices and edges. Some experimental results obtained on an application-dependent dataset concerning the spotting of symbols on technical drawings show that the approach obtains better performance than a previous approach which is only substitution-tolerant.

Subgraph Matching Edit Distance Error-Tolerant Binary Linear Programming Symbol Spotting

1 Introduction

Solving the subgraph isomorphism problem aims at determining whether or not a *pattern graph* is isomorphic to one or many subgraphs of a *target graph*. This problem has been the subject of many contributions in the literature [3, 4, 18, 20, 10, 22] since it finds many applications in various fields such as biosciences, chemistry, knowledge management, social network analysis, image scene analysis, etc.

In the context of structural pattern recognition, algorithms that solve subgraph isomorphism are particularly useful since they can be used to simultaneously consider the segmentation and the recognition of objects of interest, represented as pattern graphs, in a whole image represented as the target graph. As compensation, two drawbacks can be mentioned. The first one is the computational complexity. Since it is proven that subgraph isomorphism is an NP-complete problem [9], the processing of large graphs (as those that can be extracted from images) is untractable. The second one is the requirement of a strict matching (isomorphism) between the pattern graph and the subgraph of the target graph. This requirement is a bottleneck for pattern recognition applications, where graphs to be analyzed are usually affected by distortions due to the intrinsic variability of patterns in image, to digitization procedure, or to the graph construction processing steps (skeletonization, region segmentation...).

In this context, the matching must tolerate differences by relaxing some constraints. Hence, practical algorithms must accomodate at least with substitution of attributes on vertices and edges, and in a more general framework, have to cope with structural differences. Thus, the problem of subgraph matching turns from a subgraph isomorphism search (*i.e.* a decision problem which states if some isomorphisms exist) into the search for the subgraph in the target that minimizes a matching cost with the pattern graph (*i.e.* an optimization problem). This search, sometimes described as error-tolerant [15, 16], is called the Minimum Cost Subgraph Matching (MCSM) problem in this article.

Our contribution in this paper is a Binary Linear Program (BLP) that solves the MCSM problem in the presence of both structural and attribute distortions. It extends the formulation proposed in [13] which has been shown to be very general and efficient but which is only substitution-tolerant. The proposed new BLP tolerates deletions of vertices and/or edges in the pattern graph. It can take into account rich attributes (e.g. vectors mixing nominal and numerical values) on both vertices and edges. Its implementation can be tuned to fit the graphs under consideration through a learning procedure.

Some experimental results obtained on an application-dependent dataset concerning the localization of symbols on technical drawings show that the approach can handle problems that can not be solved by existing approaches. The proposed system detects more symbols and is faster than the existing approach which is only substitution-tolerant.

The paper is organized as follows. Section 2 introduces MCSM. Section 3 presents binary linear programming as a way to formulate the MCSM problem. Section 4 describes the experimental protocol and discusses the obtained results.

2 Minimum cost subgraph matching

Definition 1. An attributed (or labeled) simple graph is 4-tuple $G = (\mathcal{V}, \mathcal{E}, \mu, \nu)$ where

- $\mathcal{V}$ is the finite set of vertices of G
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the finite set of edges of G
- $\mu : \mathcal{V} \to \mathcal{L}_{\mathcal{V}}$ is the vertex labeling function with $\mathcal{L}_{\mathcal{V}}$ the vertex-label set
- $\nu : \mathcal{E} \to \mathcal{L}_{\mathcal{E}}$ is the edge labeling function with $\mathcal{L}_{\mathcal{E}}$ the edge-label set.

In this case, an edge can be unambiguously identified by the couple of its starting and ending vertices. Items of $\mathcal{E}$ can be denoted $e = (u, v)$, meaning an edge from vertex u to vertex v[1]. If $\mathcal{E}$ is a symmetric relation, $(u, v) \in \mathcal{E} \Leftrightarrow$

[1] For the sake of simplicity of notations, uv will also be used to represent an edge from u to v

$(v,u) \in \mathcal{E}, \forall u, v \in \mathcal{V} \times \mathcal{V}$, then the graph G is said to be undirected. Conversely, it is referred to as a directed graph.

Definition 2. Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mu_1, \nu_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mu_2, \nu_2)$ be two attributed simple graphs. G_1 is a subgraph of G_2, written $G_1 \subseteq G_2$, if and only if

- $\mathcal{V}_1 \subseteq \mathcal{V}_2$
- $\mathcal{E}_1 \subseteq \mathcal{E}_2$
- $\mu_1(v) = \mu_2(v), \forall v \in \mathcal{V}_1$
- $\nu_1(e) = \nu_2(e), \forall e \in \mathcal{E}_1$.

Definition 3. A graph isomorphism between two graphs $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mu_1, \nu_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mu_2, \nu_2)$ is a bijection $f : \mathcal{V}_1 \to \mathcal{V}_2$ such that $\forall u, v \in \mathcal{V}_1 : (u, v) \in \mathcal{E}_1 \Leftrightarrow (f(u), f(v)) \in \mathcal{E}_2$

Definition 4. Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mu_1, \nu_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mu_2, \nu_2)$ be two graphs. An injective function $f : \mathcal{V}_1 \to \mathcal{V}_2$ is a subgraph isomorphism from G_1 to G_2 if there exists a subgraph $S \subseteq G_2$ such that $f(.)$ is a graph isomorphism between G_1 and S.

As previously mentionned, in the field of pattern recognition, as in many real-world applications, the search for an occurrence of a pattern into the target graph can not be performed through the search of a subgraph isomorphism. Indeed, occurrences of the searched pattern in the target graph may differ from the pattern model. These differences between the pattern graph and its occurrence in the target graph may affect attributes of vertices and edges, but also the structure of the graph.

As a consequence, the search for an exact matching between the pattern and a subgraph of the target graph is likely to be unsuccessful. In order to cope with the noise in structural representations the matching must be error-tolerant, and the objective turns into the search for the subgraph of the target graph whose dissimilarity with the pattern graph is minimal. Given $d : \mathcal{G} \times \mathcal{G} \to \mathbb{R}^+$, a measure of the difference between two graphs, we want to find G, the subgraph of the target graph G_2 whose distance to the pattern graph G_1 is minimal.

$$G = \underset{G_i \subseteq G_2}{\operatorname{argmin}}\, d(G_1, G_i) \tag{2.1}$$

The graph edit distance d_{GED} is commonly used to evaluate the dissimilarity between two graphs. Many approaches have been proposed to compute or approximate d_{GED} [8, 17], including linear programming [1, 12].

Definition 5. Let G_1 and G_2 be two graphs, the graph edit distance between G_1 and G_2 is defined by :

$$d_{GED}(G_1, G_2) = \min_{o=(o_1,\ldots,o_k)\in\mathcal{O}} \sum_i c(o_i) \tag{2.2}$$

where $\mathcal{O}$ is the set of all edit paths $o = (o_1, \ldots, o_k)$ allowing to transform G_1 into G_2. An elementary edit operation o_i is one of vertex substitution $(v_1 \rightarrow v_2)$, edge substitution $(e_1 \rightarrow e_2)$, vertex deletion $(v_1 \rightarrow \epsilon)$, edge deletion $(e_1 \rightarrow \epsilon)$, vertex insertion $(\epsilon \rightarrow v_2)$, edge insertion $(\epsilon \rightarrow e_2)$, with $v_1 \in \mathcal{V}_1$, $v_2 \in \mathcal{V}_2$, $e_1 \in \mathcal{E}_1$, $e_2 \in \mathcal{E}_2$ and ϵ a dummy element.

$c(.)$ is a cost function on elementary edit operations o_i, that satisfies :

- $c(v_1 \rightarrow v_2) \leq c(v_1 \rightarrow v) + c(v \rightarrow v_2)$
- $c(e_1 \rightarrow e_2) \leq c(e_1 \rightarrow e) + c(e \rightarrow e_2)$
- $c(v_1 \rightarrow \epsilon) \leq c(v_1 \rightarrow v) + c(v \rightarrow \epsilon)$
- $c(e_1 \rightarrow \epsilon) \leq c(e_1 \rightarrow e) + c(e \rightarrow \epsilon)$
- $c(\epsilon \rightarrow v_2) \leq c(\epsilon \rightarrow v) + c(v \rightarrow v_2)$
- $c(\epsilon \rightarrow e_2) \leq c(\epsilon \rightarrow e) + c(e \rightarrow e_2)$

If the cost function also satisfies the conditions of positive definiteness, symmetry and the triangle inequality of elementary edit operations o_i, the graph edit distance is a metric.

In equation 2.1, G is the subgraph of G_2 which minimizes its distance to G_1. This distance does not need to take care about the vertex or edge insertions in G_1. Indeed, if an edit path transforms a graph G_1 into a graph G_i, there exists a subgraph of G_i which results from the same edit path but without any insertion in G_1. The cost of this edit path is lower and the corresponding subgraph is a better candidate. Finally, our problem turns into the search for the minimum cost edit path which transforms G_1 into G_2 where insertion costs are zero. In this edit path, deletion and substitution edit operations transform G_1 into the subgraph of G_2 it is matched to, while zero cost insertion operations complete the subgraph up to G_2.

3 Formulation as a binary linear program

This article proposes to solve the MCSM problem defined in the preceding section using an optimization technique called binary linear programming, also known as $0-1$ linear programming. It is a restriction of integer linear programming (ILP), where variables are binary. These techniques are part of the more general concept of mathematical programming.

3.1 Binary linear programming

The general form of a binary linear program (BLP) is as follows :

$$\min_{x} \; c^T x \tag{3.1}$$

$$\text{s.t. } Ax \leq b \tag{3.2}$$

$$x \in \{0,1\}^n \tag{3.3}$$

where $c \in \mathbb{R}^n$, $A \in \mathcal{M}_{m,n}(\mathbb{R})$ and $b \in \mathbb{R}^m$ are data of the problem.

A solution is a vector x of n binary variables. Since A and b are used to define linear inequality constraints in (3.2), a feasible solution for the problem is a vector $x \in \{0,1\}^n$ such that constraints (3.2) are respected. The objective function $c^T x$ is a linear combination of the binary variables of x. To find an optimal solution, the objective function (3.1) is minimized over the set of feasible solutions.

There is no known polynomial-time algorithm to solve a BLP, since this task is NP-hard. When n is large, one can not simply explore the entire solution tree, since it takes an exponential time. Such problems are tackled with the help of mathematical solvers that have been designed for solving ILP and BLP as well. They implement a branch-and-cut algorithm, used along with heuristics, to reduce the search space. The lower bound of the optimal objective is computed by solving the continuous relaxation of the program with the interior point method.

3.2 Formulation of the problem

In order to formulate the error-tolerant subgraph matching problem as a BLP, we define four sets of binary variables:

- $\forall i \in \mathcal{V}_1, \forall k \in \mathcal{V}_2, x_{i,k} \in \{0,1\}$ encodes the vertex attribute substitution $(i \to k)$. $x_{i,k}$ is set to 1 if i is substituted with k and 0 otherwise ;
- $\forall i \in \mathcal{V}_1, \alpha_i \in \{0,1\}$ encodes the vertex deletion $(i \to \epsilon)$. α_i is set to 1 if i is deleted from G_1 and 0 otherwise ;
- $\forall ij \in \mathcal{E}_1, \forall kl \in \mathcal{E}_2, y_{ij,kl} \in \{0,1\}$ encodes the edge substitution $(ij \to kl)$. $y_{ij,kl}$ is set to 1 if ij is substituted with kl and 0 otherwise ;
- $\forall ij \in \mathcal{E}_1, \beta_{ij} \in \{0,1\}$ encodes the deletion $(ij \to \epsilon)$. β_{ij} is set to 1 if ij is deleted from G_1 and 0 otherwise.

Let us denote $\mathbf{x} = (x_{i,k})_{i\in\mathcal{V}_1,k\in\mathcal{V}_2}$, $\boldsymbol{\alpha} = (\alpha_i)_{i\in\mathcal{V}_1}$, $\mathbf{y} = (y_{ij,kl})_{ij\in\mathcal{E}_1,kl\in\mathcal{E}_2}$ and $\boldsymbol{\beta} = (\beta_{ij})_{ij\in\mathcal{E}_1}$. Given a cost function $c(.)$ as defined in section 2, the objective function of the BLP can be written as the sum of the costs of the elementary edit operations o_i that are necessary to match G_1 to a subgraph $S \subseteq G_2$:

$$\min_{\mathbf{x},\mathbf{y},\boldsymbol{\alpha},\boldsymbol{\beta}} \Bigg(\sum_{i\in\mathcal{V}_1} \sum_{k\in\mathcal{V}_2} x_{i,k} \cdot c(i \to k) + \sum_{i\in\mathcal{V}_1} \alpha_i \cdot c(i \to \epsilon) \\ + \sum_{ij\in\mathcal{E}_1} \sum_{kl\in\mathcal{E}_2} y_{ij,kl} \cdot c(ij \to kl) + \sum_{ij\in\mathcal{E}_1} \beta_{ij} \cdot c(ij \to \epsilon) \Bigg) \tag{3.4}$$

In order to force the 4-tuple $(\mathbf{x}, \mathbf{y}, \boldsymbol{\alpha}, \boldsymbol{\beta})$ to describe a valid edit path $o \in \mathcal{O}$ which transforms G_1 into $S \subseteq G_2$, the following set of constraints must be defined :

- A vertex of G_1 can be matched to at most one vertex of G_2:

$$\sum_{k \in \mathcal{V}_2} x_{i,k} \leq 1 \quad \forall i \in \mathcal{V}_1 \tag{3.5}$$

  If a vertex of G_1 is not matched to any vertex of G_2, it must be deleted:

$$\alpha_i = 1 - \sum_{k \in \mathcal{V}_2} x_{i,k} \quad \forall i \in \mathcal{V}_1 \tag{3.6}$$

- A vertex of G_2 can be matched to at most one vertex of G_1:

$$\sum_{i \in \mathcal{V}_1} x_{i,k} \leq 1 \quad \forall k \in \mathcal{V}_2 \tag{3.7}$$

- An edge of G_1 can be matched to at most one edge of G_2, provided that their head vertices on the one hand, and their tail vertices on the other hand, are respectively matched:

$$\sum_{l \in \mathcal{V}_2, kl \in \mathcal{E}_2} y_{ij,kl} \leq x_{i,k} \quad \forall ij \in \mathcal{E}_1, \forall k \in \mathcal{V}_2 \tag{3.8}$$

$$\sum_{k \in \mathcal{V}_2, kl \in \mathcal{E}_2} y_{ij,kl} \leq x_{j,l} \quad \forall ij \in \mathcal{E}_1, \forall l \in \mathcal{V}_2 \tag{3.9}$$

  If an edge of G_1 is not matched to any edge of G_2, it must be deleted:

$$\beta_{ij} = 1 - \sum_{kl \in \mathcal{E}_2} y_{ij,kl} \quad \forall ij \in \mathcal{E}_1 \tag{3.10}$$

Equations (3.6) and (3.10) are not needed as constraints of the BLP, since they are implicitly respected, provided that constraints (3.5), (3.8) and (3.9) are satisfied. In order to reduce the size of the search space (i.e. the number of variables), we replace the deletion variables in the objective function by their expressions.

We finally get the following BLP:

$$\min_{\mathbf{x},\mathbf{y}} \Bigg(\sum_{i \in \mathcal{V}_1} \sum_{k \in \mathcal{V}_2} x_{i,k} \cdot (c(i \to k) - c(i \to \epsilon)) + \sum_{i \in \mathcal{V}_1} c(i \to \epsilon) \\ + \sum_{ij \in \mathcal{E}_1} \sum_{kl \in \mathcal{E}_2} y_{ij,kl} \cdot (c(ij \to kl) - c(ij \to \epsilon)) + \sum_{ij \in \mathcal{E}_1} c(ij \to \epsilon) \Bigg) \tag{3.11a}$$

subject to :

$$\sum_{k \in \mathcal{V}_2} x_{i,k} \leq 1 \qquad \forall i \in \mathcal{V}_1 \tag{3.11b}$$

$$\sum_{i \in \mathcal{V}_1} x_{i,k} \leq 1 \qquad \forall k \in \mathcal{V}_2 \tag{3.11c}$$

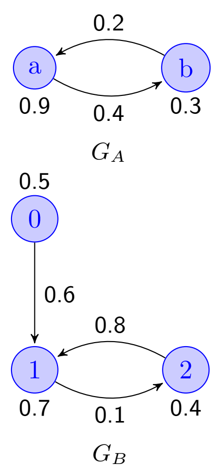
$$\sum_{l \in \mathcal{V}_2, kl \in \mathcal{E}_2} y_{ij,kl} \leq x_{i,k} \quad \forall ij \in \mathcal{E}_1, \forall k \in \mathcal{V}_2 \tag{3.11d}$$

$$\sum_{k \in \mathcal{V}_2, kl \in \mathcal{E}_2} y_{ij,kl} \leq x_{j,l} \quad \forall ij \in \mathcal{E}_1, \forall l \in \mathcal{V}_2 \tag{3.11e}$$

$$x_{i,k} \in \{0,1\} \quad \forall i \in \mathcal{V}_1, \forall k \in \mathcal{V}_2 \tag{3.11f}$$

$$y_{ij,kl} \in \{0,1\} \quad \forall ij \in \mathcal{E}_1, \forall kl \in \mathcal{E}_2 \tag{3.11g}$$

Figure 1 gives a didactic example showing variable values and the corresponding objective function for a toy problem. In this problem, deletion costs are set to 1 for both vertices and edges.



pattern : G_A target : G_B	pattern : G_B target : G_A
$x_{a,0} = 0$	$x_{0,a} = 0$
$x_{a,1} = 1$	$x_{0,b} = 0$
$x_{a,2} = 0$	$x_{1,a} = 1$
$x_{b,0} = 0$	$x_{1,b} = 0$
$x_{b,1} = 0$	$x_{2,a} = 0$
$x_{b,2} = 1$	$x_{2,b} = 1$
$y_{ab,01} = 0$	$y_{01,ab} = 0$
$y_{ab,12} = 1$	$y_{01,ba} = 0$
$y_{ab,21} = 0$	$y_{12,ab} = 1$
$y_{ba,01} = 0$	$y_{12,ba} = 0$
$y_{ba,12} = 0$	$y_{21,ab} = 0$
$y_{ba,21} = 1$	$y_{21,ba} = 1$
$d = 1.2$	$d = 3.2$

Figure 1: Example of variable values and objective function obtained by MCSM for a toy-problem

Let us emphasize that by modifying the notations, the formulation given by equations (3.11a) to (3.11g) is also valid for multi-graphs.

3.3 Extension to induced subgraphs

An induced subgraph isomorphism is a more stringent problem, defined by :

Definition 6. Let $G_1 = (\mathcal{V}_1, \mathcal{E}_1, \mu_1, \nu_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2, \mu_2, \nu_2)$ be two graphs. An injective function $f : \mathcal{V}_1 \to \mathcal{V}_2$ is an induced subgraph isomorphism from G_1 to G_2 if and only if $\forall (u,v) \in \mathcal{V}_1 \times \mathcal{V}_1, (u,v) \in \mathcal{E}_1 \Leftrightarrow (f(u), f(v)) \in \mathcal{E}_2$.

The equivalence in definition 6 requires an additional set of constraints in the BLP :

$$\sum_{i \in \mathcal{V}_1} x_{i,k} + \sum_{j \in \mathcal{V}_1} x_{j,l} - \sum_{ij \in \mathcal{E}_1} y_{ij,kl} \leq 1 \quad \forall kl \in \mathcal{E}_2 \tag{3.12}$$

These constraints ensure that all the edges of the matched subgraph of G_2 have a corresponding edge in G_1.

3.4 Extension to undirected graphs

The formulation given by equations (3.11a) to (3.11g) is dedicated to directed graphs, but it can be extended to undirected graphs. In an undirected graph $G = (\mathcal{V}, \mathcal{E}, \mu, \nu)$, we have :

$$ij \in \mathcal{E} \Leftrightarrow ji \in \mathcal{E}, \forall i, j \in \mathcal{V} \times \mathcal{V}$$

Thus, given two undirected edges, there are two ways of matching them. This leads to replace the equations (3.11d) and (3.11e) by the following equation in the BLP :

$$\sum_{l \in \mathcal{V}_2 : kl \in \mathcal{E}_2} y_{ij,kl} \leq x_{i,k} + x_{j,k} \quad \forall ij \in \mathcal{E}_1, \forall k \in \mathcal{V}_2 \tag{3.13}$$

Please take note that the variables $x_{i,k}$ and $x_{j,k}$ are mutually exclusive (i.e. they cannot both take the value 1), because of constraints (3.7). Therefore, constraint (3.13) still guarantees that an edge of G_1 is matched to at most on edge of G_2.

3.5 Implementation issues : the case of multiple instances

Once the MCSM formulation implemented in a mathematical ILP solver, solving an instance for a given couple (pattern,target) with a given set of edit costs results in the best one-to-one mapping (and its cost) between the vertices and edges of both graphs, possibly with some deletions of vertices and edges in the pattern graph. As defined by equations (3.11a) to (3.11g), the BLP model is only capable of finding the optimal solution. Depending on the application context, it may be the case that the pattern graph that is searched for has many instances in the target graph. There are multiple ways to manage such an issue [5]. In the context of our study, we have chosen to call iteratively the model and to discard the successive optimal solutions after each call. Such a solution is linear in the number of instances. There are multiple ways to discard an optimal solution $(\bar{x}\ \bar{y})^T$. The general idea is that a new constraint cutting the current solution is added to the model. Hence, the current optimal solution becomes infeasible for the next run. The solver can be called again and will be able to find another optimal solution. In the sake of our study, the following constraint is added to the formulation :

$$\sum_{i \in \mathcal{V}_1, k \in \mathcal{V}_2} \left(\sum_{j \in \mathcal{V}_1} \bar{x}_{j,k} \right) * x_{i,k} = 0$$

It discards every vertex of $\mathcal{V}_2$ that has been used in the current optimal solution $(\bar{x}\ \bar{y})^T$. It means that for every vertex k of $\mathcal{V}_2$, if there exists a vertex j of $\mathcal{V}_1$ matched to k, then $x_{i,k}$ equals 0 for every vertex i of $\mathcal{V}_1$.

4 Experiments and results

This section aims at showing the efficiency of the MCSM approach proposed in this article for detecting pattern graphs as sub-graphs of a target graph when both structural and attribute distortions occur. To the best of our knowledge, no reference dataset exists for evaluating such a task. Therefore, we consider in this article an application-oriented problem related to the spotting of distorted symbol in graphical document images. To ensure the possibility of comparing our results with future contributions on this subject, the graph dataset is available at `http://litis-ilpiso.univ-rouen.fr/ILPIso/`.

4.1 Graph Dataset

Symbol spotting is a problem related to document image analysis. Its objective is to detect the occurrences of some pattern symbols in a target document. For this problem, structural representation are known to be useful since (i) symbols can usually be defined as a composition of parts and (ii) graph tools are well suited to consider the segmentation/recognition problems.

In our experiments, symbol and document images are represented using Region Adjacency Graphs (RAG) $G = (\mathcal{V}, \mathcal{E}, \mu, \nu)$ where $\mathcal{V}$ denotes the regions (the loops) of the image and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ stands for adjacency relations between the regions. $\mu : \mathcal{V} \rightarrow \mathbb{R}^{26}$ describes the morphology of a region with its area and 25 Zernike moments computed using the approach described in [19]. $\nu : \mathcal{E} \rightarrow \mathbb{R}^2$ expresses two properties of adjacency relations : the *relative scale* between the two regions given by $\min(A(i), A(j))/\max(A(i), A(j))$ where $A(i)$ is the area of region i, and the *relative euclidean distance* between the gravity centers of the two regions, with respect to their overall area given by $d_e(g_i, g_j)/\sqrt{A(i) + A(j)}$ where $d_e(g_i, g_j)$ is the Euclidean distance between the gravity centers of regions i and j.

The document images used for building the graph dataset are the 100 documents of the 5^{th} template in the `floorplan` dataset[2] [6]. These images represent several symbol arrangements on empty architectural plan templates. Figure 2 shows an example of a floorplan, with the corresponding extracted graph (without attribute for clarity reasons). Graphs representing documents contain 121 vertices and 525 edges on average. The task associated to this dataset is the localization of the instances of 11 symbol models, examples of which are illustrated in Figure 3. From a numerical point of view, the graphs corresponding to symbol instances contain 4 vertices and 7 edges on average.

In the original floorplan dataset, pattern symbol images and symbol occurrences only differ in size and orientation. Such modifications mainly impact the

[2] `http://mathieu.delalandre.free.fr/projects/sesyd/`

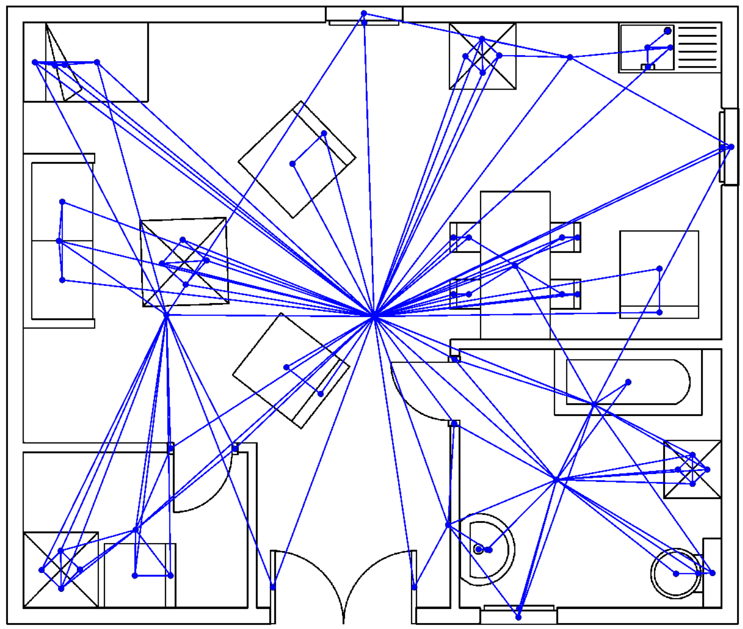


Figure 2: View of the region adjacency graph extracted from an image of the `floorplan-05` dataset

attributes of vertices and edges in the corresponding RAGs, even if skeletonization sometimes produces some artefacts in the edges for touching symbols. In order to better evaluate the MCSM proposed in this paper, we have synthetically deformed the pattern symbols at the image level so that the image modifications impact the graph in its topology. Compared to a modification of the target documents, this choice of distorting pattern images ensures that the ground-truth provided with the floorplan dataset remains valid. For performing the distortions, the image is firstly vectorized using the algorithm described in [2] and [21]. Then a vectorial noise is applied using the approach proposed in [7], with a parameter r that controls the deformation. Finally, the result is rasterized to get the image. Figure 3 illustrates the impact of the value of the noise parameter r.

4.2 Experimental protocol

From the 100 documents of the dataset described in subsection 4.1, 50 documents are used for tuning the system, and 50 are used for testing it. Hence, for the evaluation, we initially obtain for a given level of noise a set of $11 \times 50 = 550$ queries, where a query consists in a couple made of a pattern graph and a target graph. For assuring the robustness of the results, 10 repetitions of the noise generation process are applied, resulting in $10 \times 550 = 5500$ queries.

Solving the formulations of MCSM proposed in this article requires to define

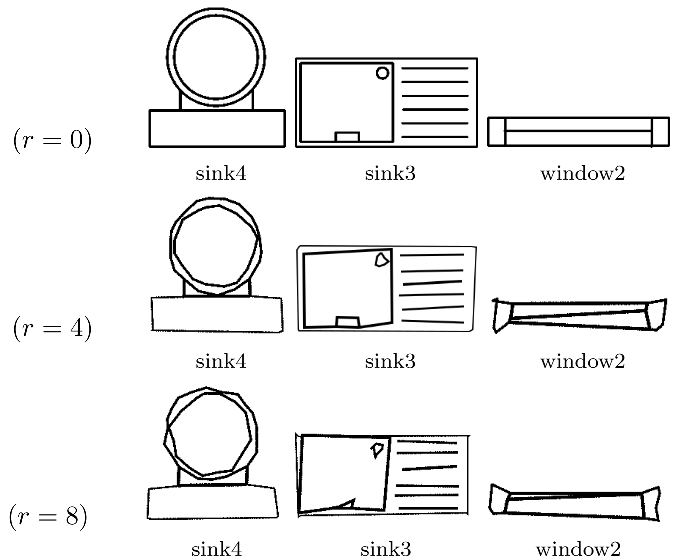


Figure 3: Examples of symbols for $r = 0$ (without degradations) $r = 4$ and $r = 8$

some edit costs. For substitution costs $c(i \rightarrow k)$ and $c(ij \rightarrow kl)$, we defined a weighted euclidean distance on vertices and edges attributes as follows:

- $c(i \rightarrow k) = \sqrt{\sum_{n=1}^{26} {w_n}^2(\mu(i)_n - \mu(j)_n)^2}$
- $c(ij \rightarrow kl) = \sqrt{\sum_{n=1}^{2} {\alpha_n}^2(\nu(ij)_n - \nu(kl)_n)^2}$

For our experiments, the w_n and α_n values have been set regarding the distributions of the absolute difference of the attributes values between queries and targets, on the training dataset. Once these values set, we have tested different values for deletion costs $C = 5, 10, 20, 40, 80$.

Since a symbol can have several instances on the same document image, the strategy described in subsection 3.5 is used in order to find multiple instances. Such a strategy requires the definition of a stopping criterion in order to avoid a large number of false detections. In our experiments, the strategy consists in learning a threshold value th_i (where i denotes the symbol class) of the BLP objective function for each class of symbol, on the learning dataset. Using these values, the search for the subgraphs is stopped as soon as the objective function value exceeds this threshold.

For the evaluation, the matching result has to be compared with the ground-truth information provided with the floorplan dataset at the image level. For this comparison, a feedback is made into the image in order to check if the vertices that compose the mapping are regions that compose real symbols (which is given as a rectangle in the floorplan ground-truth). In our experiments, we consider that a symbol is detected if more than half of the matched nodes are symbol regions. Using this criterion, the classical information retrieval metrics (precision, recall and F1-score) can be used to characterize the performance of the spotting system.

For all the experiments, we compare the results obtained using the new MCSM formulation with those of the system proposed in [13] which is a Substitution Only Tolerant Subgraph Matching (SOTSM). Both approaches are included in the GEM++ software available at `http://litis-ilpiso.univ-rouen.fr/ILPIso/` and described in [14, 11]. The same setting was used for both approaches in this evaluation.

4.3 Obtained results

Table 1 presents the results obtained on the test dataset using both MCSM and SOTSM for $r = 0, 4, 8$. In the case of MCSM, five configurations of C values are compared.

Table 1: Mean F1-score on nodes matching rate for the 5500 queries of the test dataset

Method	MCSM					SOTSM
r \ C	5	10	20	40	80	-
0	0.95	0.99	1.00	1.00	1.00	0.99
4	0.75	0.92	0.94	0.94	0.94	0.93
8	0.58	0.80	0.84	0.85	0.84	-

As expected, F1-scores decrease with the increasing of r for both MCSM and SOTSM. However, if both approaches get comparable spotting performance for $r = 0$ and $r = 4$, SOTSM reaches its limits for $r = 8$. At this level, some queries do not provide any results, what explains the lack of mean value in Table 1. For a finer analysis, Table 2 details the precision/recall results per class, for $C = 40$ (which is the best configuration according to Table 1) and $r = 8$. The results obtained for the class `sink3` illustrate the strength of the MCSM with respect to SOTSM since vertices and edges are frequently generated by the noise on such symbols. This aspect is illustrated by Figure 4 that shows our GUI in the case of a `sink3` query with both approaches. The noise generates spurious vertices and edges in the pattern graph. This results in a wrong matching (with a `table2`) when using SOTSM, whereas the matching proposed by MCSM tolerates these distortions by allowing the deletion of such extra objects. Figure 5 illustrates another search where a `window2` symbol is given as query. Here, two vertices are deleted because of a low deletion cost, what results in an error in the matching.

Table 3 compares the computation times of both approaches on the test dataset. It shows that using MCSM with a deletion cost $C = 40$ speeds up the search by a factor of 10 with respect to SOTSM, even for $r = 0$. It can be explained by the fact that, considering the algorithm that is used by BLP solvers

Table 2: Recall and precision on the test dataset detailed per class, for $r = 8$ and $C = 40$

	MCSM ($C = 40$)			SOTSM		
symbol	rec.	prec.	F1	rec.	prec.	F1
bed	0.90	1	0.95	0.81	1	0.89
sink1	0.90	1	0.95	0.90	1	0.95
sink3	0.74	1	0.85	0.10	1	0.18
sink4	0.10	0.02	0.03	0	—	—
sofa1	0.98	0.99	0.98	0.98	0.99	0.98
sofa2	0.99	1	0.99	1	1	1
table1	0.97	1	0.98	0.79	1	0.88
table2	1	1	1	1	1	1
tub	1	1	1	1	1	1
window1	0.50	1	0.67	0.50	1	0.67
window2	0.90	1	0.95	0.90	1	0.95
overall	0.82	0.91	0.85	0.72	—	—

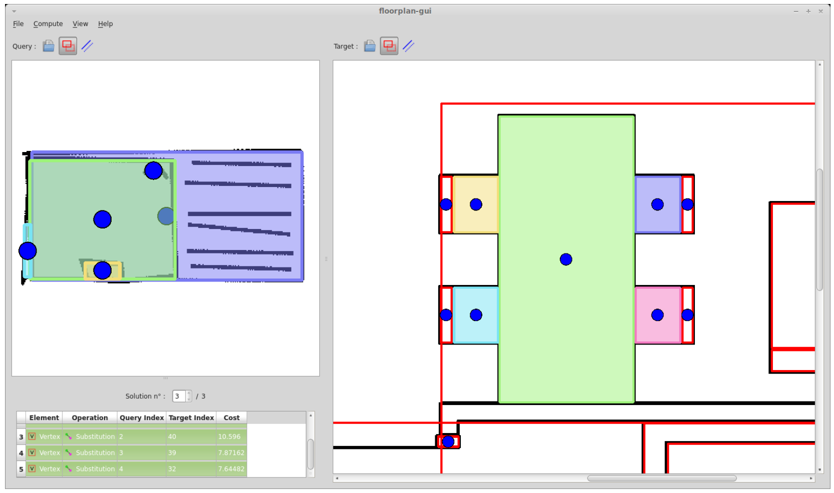
(branch-and-cut), and particularly the lower bound they use (LP continuous relaxation), the initialization of the solving gives a better approximation in the case of MCSM than in the case of SOTSM, and thus reduces the solving time of MCSM.

Table 3: Mean elapsed time by correctly found instance on the test dataset, in seconds

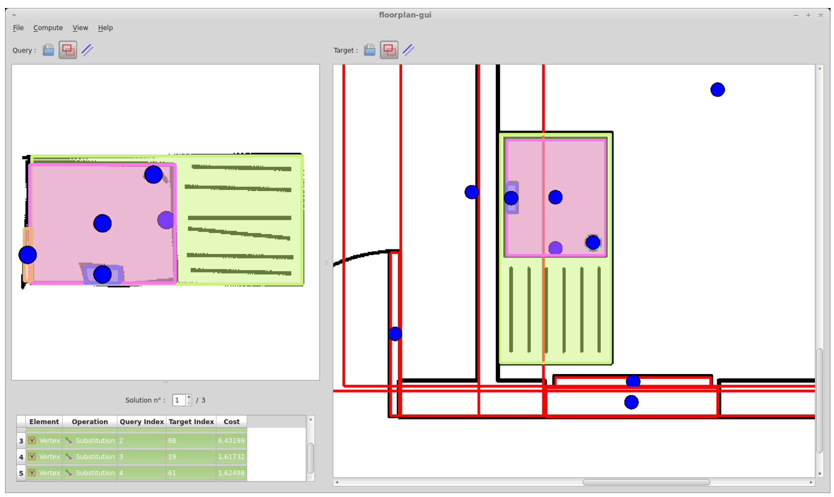
Method	MCSM					SOTSM
r \ C	5	10	20	40	80	-
0	0.11	0.13	0.17	0.41	3.21	3.09
4	0.16	0.13	0.19	0.52	4.72	4.92
8	0.22	0.18	0.27	1.14	10.05	9.26

4.4 Experiments on a synthetic dataset

In order to evaluate the scalability of the proposed formulation, some experiments have been led on the synthetic dataset used in [13]. In this dataset, the numbers of vertices in the pattern graph ($n_{\mathcal{S}}$) and in the target graphs ($n_{\mathcal{G}}$) vary, as well as the edge density of the graphs, controlled by a parameter p. For each configuration of $\{n_{\mathcal{S}}, n_{\mathcal{G}}, p\}$, 5 pattern graphs are searched for in 5 target graphs, leading to 25 queries. According to the graph generation protocol (please see [13] for a more detailed description of the dataset), the target graph contains the pattern graph only for 5 queries out of 25. Obtained processing times are given in Table 4. In these experiments, we stop the search when time exceeds 300 seconds.



SOTSM



MCSM

Figure 4: Comparison of the results of a `sink3` query on a given floorplan with both MCSM and SOTSM. The query is on the left, the target is on the right. In the query, the bottom left vertex is an extra one generated by the noise. Similar colors between the pattern and the target correspond to matched nodes.

These results show that larger graphs can be tackled by our formulation and also show its limits, particularly for dense graphs.

In order to compare processing time of MCSM and SOTSM, we use a subset of the dataset described below, keeping only the 5 out of 25 queries for which a subgraph isomorphism exists. We measured the median time taken by both methods to find the unique solution. Obtained results are given in Table 5.

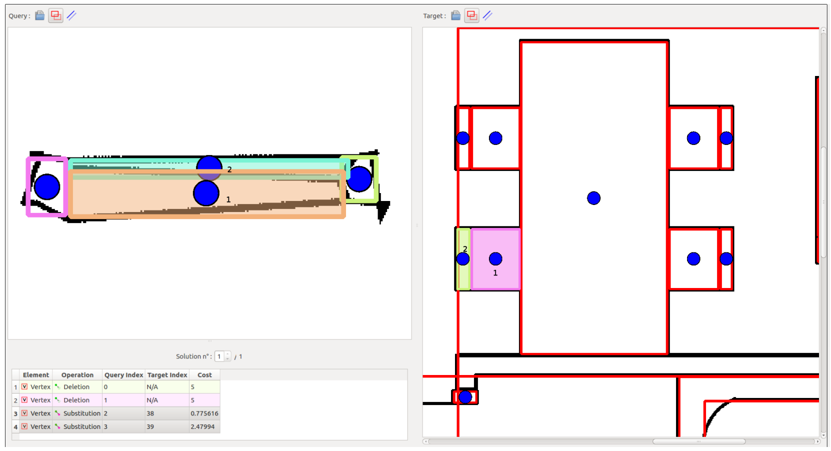


Figure 5: Example of search result of a `window2` where two deletions result in a wrong matching.

Table 4: Median elapsed time for different graph configurations on the synthetic dataset, in seconds

p	$n_{\mathcal{G}} = \|V_{\mathcal{G}}\|$	$n_{\mathcal{S}} = \|V_{\mathcal{S}}\|$		
		10	25	50
0.01	50	0.02	0.04	0.21
	100	0.03	0.15	0.27
	250	0.13	0.31	1.23
	500	0.24	0.76	5.70
0.05	50	0.04	0.95	1.24
	100	0.16	6.25	-
	250	0.33	-	-
	500	3.10	-	-
0.1	50	0.36	-	-
	100	0.39	-	-
	250	10.82	-	-
	500	17.09	-	--

These results confirm those obtained for the symbol spotting application. For the last cell of the table, a memory failure occurs in both cases.

5 Conclusions

This article has proposed a binary linear program that finds the occurrences of a pattern graph in a target graph when both structural and attribute distortions occur. Thanks to this new formulation, better matching performance

Table 5: Median elapsed time for MCSM (left value) and SOTSM (right value) on a subset of the synthetic dataset, in seconds

p	$n_{\mathcal{G}}$	$n_{\mathcal{S}}$ 10	25	50
0.01	50	0.02 / 0.03	0.04 / 0.07	0.17 / 0.19
	100	0.03 / 0.05	0.15 / 0.15	0.25 / 0.39
	250	0.12 / 0.16	0.29 / 0.29	0.93 / 1.03
	500	0.23 / 0.26	0.76 / 0.75	3.61 / 3.08
0.05	50	0.04 / 0.05	0.21 / 0.24	0.84 / 0.99
	100	0.16 / 0.17	0.50 / 0.56	2.88 / 3.37
	250	0.31 / 0.38	3.88 / 5.21	16.69 / 21.54
	500	2.93 / 3.01	24.34 / 46.12	138.1 / 276.4
0.1	50	0.17 / 0.19	0.51 / 0.65	3.43 / 5.11
	100	0.25 / 0.28	1.87 / 2.42	8.96 / 13.10
	250	2.74 / 3.28	15.58 / 25.56	106.8 / 157.6
	500	11.89 / 14.27	171.2 / 320.1	- / -

are achieved on a pattern spotting problem, compared to those provided by an approach which only tolerates attribute substitution. The approach is shown to be faster than an only substitution tolerant approach and can be applied to richly attributed graphs. Our current works concern an embedded learning of the edit costs, in order to get a fully adaptive system.

References

[1] H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. IEEE Trans. Pattern Anal. Mach. Intell., 15(5):522–525, May 1993.

[2] G. Sanniti Di Baja and E. Thiel. Skeltonization algorithm running on path-based distance maps. Image and Vision Computing, 14:47–57, 1996.

[3] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. Performance evaluation of the VF graph matching algorithm. In Proc. of the Int. Conf. on Image Analysis and Processing, pages 1172–1177, 1999.

[4] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. A (sub)graph isomorphism algorithm for matching large graphs. IEEE Trans. on Patt. Anal. and Mach. Intell., 26(10):1367–1372, 2004.

[5] Emilie Danna, Mary Fenelon, Zonghao Gu, and Roland Wunderling. Generating multiple solutions for mixed integer programming problems. In IPCO '07: Proc. of the 12th int. conf. on Integer Programming and Combinatorial Optimization, pages 280–294, 2007.

[6] Mathieu Delalandre, Ernest Valveny, Tony P. Pridmore, and Dimosthenis Karatzas. Generation of synthetic documents for performance evaluation of symbol recognition & spotting systems. IJDAR, 13(3):187–207, 2010.

[7] Anjan Dutta, Josep Lladós, and Umapada Pal. A symbol spotting approach in graphical documents by hashing serialized graphs. Pattern Recognition, 46(3):752 – 768, 2013.

[8] Andreas Fischer, Ching Y. Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. Approximation of graph edit distance based on hausdorff matching. Pattern Recognition, 48(2):331–343, 2015.

[9] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman & co., 1979.

[10] David E. Ghahraman, Andrew K.C. Wong, and Tung Au. Graph optimal monomorphism algorithms. IEEE Trans. on System, Man and Cybernetics, 10:181–188, 1980.

[11] Maroua Hammami, Pierre Héroux, Sébastien Adam, and Vincent Poulain D'Andecy. One-shot field spotting on colored forms using subgraph isomorphism. In 2015 13th International Conference on Document Analysis and Recognition, Nancy, France, 2015, 2015.

[12] Derek Justice and Alfred Hero. A binary linear programming formulation of the graph edit distance. IEEE Trans. Pattern Anal. Mach. Intell., 28(8):1200–1214, August 2006.

[13] Pierre Le Bodic, Pierre Héroux, Sébastien Adam, and Yves Lecourtier. An integer linear program for substitution-tolerant subgraph isomorphism and its use for symbol spotting in technical drawings. Pattern Recognition, 45(12):4214–4224, 2012.

[14] Julien Lerouge, Pierre Le Bodic, Pierre Héroux, and Sébastien Adam. Gem++: A tool for solving substitution-tolerant subgraph isomorphism. In Cheng-Lin Liu, Bin Luo, Walter G. Kropatsch, and Jian Cheng, editors, Graph-Based Representations in Pattern Recognition, volume 9069 of Lecture Notes in Computer Science, pages 128–137. Springer International Publishing, 2015.

[15] Josep Lladós, Enric Martí, and Juan José Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. IEEE Trans. PAMI, 23(10):1137–1143, 2001.

[16] Bruno T. Messmer and Horst Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. IEEE Trans. Pattern Anal. Mach. Intell., 20(5):493–504, 1998.

[17] Kaspar Riesen and Horst Bunke. Improving bipartite graph edit distance approximation using various search strategies. Pattern Recognition, 48(4):1349–1363, 2015.

[18] Christine Solnon. Alldifferent-based filtering for subgraph isomorphism. Artificial Intelligence, 174(12-13):850 – 864, 2010.

[19] M. Teague. Image analysis via the general theory of moments. Journal of the Optical Society of America, 70(8):920–930, 1980.

[20] Julian R. Ullmann. An algorithm for subgraph isomorphism. Journal of the ACM, 23(1):31–42, 1976.

[21] Karin Wall and Per-Erik Danielsson. A fast sequential method for polygonal approximation of digitized curves. Computer Vision, Graphics, and Image Processing, 28(2):220 – 227, 1984.

[22] Andrew K.C. Wong, Manlai You, and S.C. Chan. An algorithm for graph optimal monomorphism. IEEE Trans. on System, Man and Cybernetics, 20(3):628–638, 1990.