Graph-Based Modeling of Alignment Relations and Pattern Detection

Rita-Nathalia Assaf[1 [0009−0001−3059−9953]], Frédéric Lardeux[1 [0000−0001−8636−3870]], and Frédéric Saubion[1 [0000−0001−8227−7174]]

Univ Angers, LERIA, SFR MATHSTIC, F-49000 Angers, France

Abstract. We introduce a unified framework for modeling alignment relations between discrete tokens using position spaces defined by two strict partial orders: horizontal and vertical alignment. These yield two complementary graph representations—the directed position graph and the bipartite row/column graph, on which alignment motifs are formalized as subgraph-isomorphism patterns and detected via a CSP encoding enriched with optional type, order, and cardinality constraints, combined with the ILF filtering algorithm. Experiments on controlled families of position spaces show that ILF substantially reduces search effort on position graphs, while structural constraints are critical for making many row/column cases tractable.

Keywords: Subgraph isomorphism · Modeling · Filtering Algorithms.

1 Introduction

General context Reasoning about spatial organization is an important challenge in artificial intelligence, with applications ranging from image understanding to geographic information systems [3]. Qualitative frameworks such as RCC8 capture topological relations between continuous regions (e.g., disjointness, overlap, containment) without depending on numerical metric coordinates [9]. Despite their success and the availability of efficient reasoning techniques within the constraint programming (CP) community [8], such formalisms remain fundamentally region-centric. In contrast, real situations are driven by the relative alignment of discrete tokens—horizontal and vertical ordering—rather than by metric distances or topological adjacency. This paper addresses that gap by proposing a token-centric formalism in which alignment relations are modeled as strict partial orders, giving rise to rows, columns, and patterns that are not naturally handled by metric or topological approaches.

A real application comes from our previous work on automatically extracting information (e.g, tables) from structured documents, particularly invoices [11]. Figure 1 illustrates how a document can be modeled as a graph, once its content has been processed using standard tools (OCR, tokenization, etc.), which notably allows the extraction of positional information for the document's elements.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

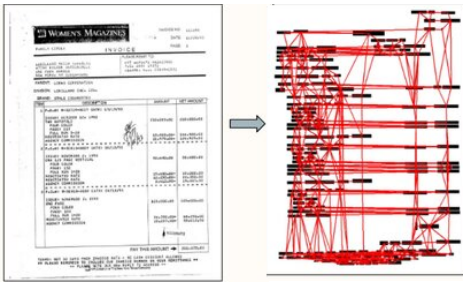


Fig. 1: Example of transforming an invoice into a graph after analyzing the initial input image of the invoice (e.g., a scanned document). The red edges represent alignment relations between the extracted elementary entities. Note that this invoice contains a table (illustration from [11]).)

Motivations The purpose of this work is more modeling-oriented rather than application-driven. As mentioned above, many spatial reasoning frameworks emphasize metric or topological properties, yet they do not explicitly capture the combinatorial structures induced by alignment relations. This gap motivates the need for a formal representation of alignment-based position spaces. In this context, with appropriate modeling, the search for a given pattern within a position space can be reduced to the search for a subgraph within a graph. The subgraph isomorphism problem (SIP) is known to be NP-complete in the general case [2]. Dedicated algorithms for solving SIP have been proposed for decades [13]. More recently, efficient exact algorithms based on filtering techniques have become available [14, 4, 7]. These approaches are grounded in constraint programming principles, notably through the use of specialized filtering algorithms tailored to SIP [15], where the problem is treated as a global constraint. Indeed, SIP can be naturally expressed as a constraint satisfaction problem (CSP), particularly by leveraging global constraints such as the Alldiff constraint [10]. Hence, we intend to propose a framework that can easily be embedded in a constraint programming solving process.

Contributions Our contribution is primarily conceptual: we propose a unified formalism, based on graphs, for modeling alignment-based position spaces and describe how structural motifs can be expressed within this formalism. The solving techniques are issued from constraint programming, including ILF [15], and used only to expose and analyze the structural properties of the proposed models. The paper does not aim to advance the state of the art in subgraph isomorphism solving but rather focuses on understanding the modeling landscape and the interactions between different abstractions of alignment relations. Our experiments reveal weaknesses in the existing filtering techniques developed for handling general subgraph isomorphism problems in constraint solvers. Instead, our dual modeling approach enhances the solver's performance by adding specific constraints to the basic model.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

2 Position Graphs

2.1 Spaces and Order Relations

To provide a general framework, we rely on partial order relations that allow us to define the relative positioning of objects/items/tokens.

Definition 1 (Alignment Relation). *Given a set of tokens $\mathcal{T}$, an alignment relation $<\subseteq \mathcal{T} \times \mathcal{T}$ is a partial order such that $\forall t, t', t'' \in \mathcal{T}, (t < t' \wedge t < t'') \Rightarrow (t' < t'' \vee t'' < t')$ and $(t' < t \wedge t'' < t) \Rightarrow (t' < t'' \vee t'' < t')$.*

The previous definition means that any subset of three tokens must either not be comparable or totally ordered. The condition introduced in this definition prevents the presence of inconsistent alignments from our point of view, meaning alignments that correspond to an incongruent division of the same line. We consider here a strict partial order relation.

Definition 2 (Position Space). *A position space is a triplet $(\mathcal{T}, <_h, <_v)$ such that $<_h$ (horizontal alignment) and $<_v$ (vertical alignment) are two alignements relations defined on $\mathcal{T}$.*

A position space defines the organization of tokens according to their vertical and horizontal alignments. We assume that every element of $\mathcal{T}$ must have a relative positioning with respect to at least one other element (either vertical or horizontal).

In the following we consider only position spaces that are consistent, i.e. where tokens can only be ordered according to one of the alignment relations. Also if two tokens are ordered with a single alignment relation, then there cannot exist an ordering sequence of tokens between these two tokens that contains a different alignment relation.

Using the alignment relations $<_h$ (resp. $<_v$), we naturally define the implicit notion of row (resp. column).

Definition 3 (Rows and Columns). *Given a position space $\Pi = (\mathcal{T}, <_h, <_v)$, a row $L \subseteq \mathcal{T}$ (resp. a column $C \subseteq \mathcal{T}$) is a totally ordered subset $(L, <_h)$ (resp. $(C, <_v)$). $\mathcal{L}_\Pi \subseteq 2^\mathcal{T}$ (resp. $\mathcal{C}^\Pi \subseteq 2^\mathcal{T}$) is the set of rows (resp. columns) associated with Π.*

Note that $(\mathcal{L}^\Pi, \subseteq)$ and $(\mathcal{C}^\Pi, \subseteq)$ are two partially ordered sets under the inclusion relation. A complete row (resp. column) is a maximal element of $(\mathcal{L}_\Pi, \subseteq)$ (resp. $(\mathcal{C}_\Pi, \subseteq)$). Note that the set of complete rows (resp. complete columns) $\mathcal{L}_\Pi^{comp}$ (resp. $\mathcal{C}_\Pi^{comp}$) forms a partition of $\mathcal{L}_\Pi$ (resp. $\mathcal{C}_\Pi$).

2.2 Graph Representation

Our aim is now to introduce a graph-based representation of the position space. As mentioned in the Introduction, this representation is motivated by real cases,

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

where graph structures allow to naturally vizualize situtation and to easily define sub-structures (patterns) that will be searched for. Moreover, representing partial ordering relations by means of graphs is rather natural (e.g., Hasse diagrams).

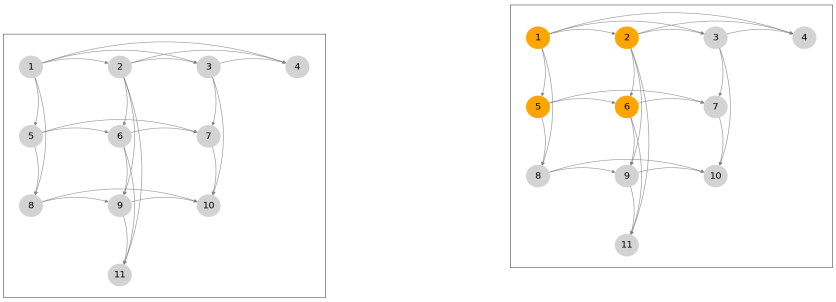
Definition 4 (Position Graph). *Given a position space* $\Pi = (\mathcal{T}, <_h, <_v)$*, the induced position graph* GP_Π *is defined as a labeled directed graph* $(\mathcal{T}, \mathcal{A}, \mathcal{E})$ *where* $\mathcal{A} \subseteq \mathcal{T} \times \mathcal{T}$ *and* $\mathcal{E} : \mathcal{T} \times \mathcal{T} \to \{h, v\}$ *such that:*

- $\forall t, t' \in \mathcal{T}, t <_h t' \Leftrightarrow (t, t') \in \mathcal{A} \wedge \mathcal{E}((t, t')) = h$
- $\forall t, t' \in \mathcal{T}, t <_v t' \Leftrightarrow (t, t') \in \mathcal{A} \wedge \mathcal{E}((t, t')) = v$

Example 1. Let $\mathcal{T} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. We define our position space by setting: $1 <_h 2, 2 <_h 3, 3 <_h 4, 5 <_h 6, 6 <_h 7, 8 <_h 9, 9 <_h 10$, $1 <_v 5, 5 <_v 8, 2 <_v 6, 6 <_v 9, 9 <_v 11, 3 <_v 7, 7 <_v 10$.

In this example, we can observe that element 4 is not vertically aligned with any other element, which will result in a column containing only one element. The same applies to element 11 with respect to $<_h$.

In example 1, we obtain the graph shown in Figure 2a. In Figure 2a, we omit the labels of the arcs because those labeled 'h' will be drawn horizontally and those labeled 'v' vertically.



(a) Position graph GP

(b) Pattern M embedded in GP (in orange)

Fig. 2: Unified illustration of the models: the row/column graph (top) and the corresponding position graph and embedded pattern (bottom).

Property 1 (Consistent Position Graph). A position graph $GP_\Pi = (\mathcal{T}, \mathcal{A}, \mathcal{E})$ is consistent if and only if : $\forall (t, t') \in \mathcal{A}$ there does not exist another (directed) path from t to t' in GP_Π that contains an arc $a \in \mathcal{A}$ such that $\mathcal{E}((t, t')) \neq \mathcal{E}(a)$.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

The proof is based on the intuition that starting from t using the other alignment relation necessarily to reach a vertex t'', thanks to a given correct directed arc, requires coming back to t' using reverse arcs (hence not authorized).

According to the Definition 3, columns and rows can also be used to describe the position space. Hence, we define an alternative graph representation based on this concept. As we will see later on, these graphs will be useful from a computational point of view.

Definition 5 (Row/column Graph). *Given a position space* $\Pi = (\mathcal{T}, <_h, <_v)$*, the row/column graph* $GRC_\Pi = (\mathcal{L}_\Pi \cup \mathcal{C}_\Pi, \mathcal{I})$ *is the undirected graph*[1] *of row/column associated with* Π*, such that* $\mathcal{I} \subseteq \mathcal{L}_\Pi \times \mathcal{C}_\Pi$ *satisfies* $\forall (l, c) \in \mathcal{I}, l \cap c \neq \emptyset$*.* GRC_Π *is a bipartite graph.*

Note that there is an edge in the undirected graph GRC_Π, denoted (l, c), corresponding to each token $t \in \mathcal{T}$ such that $t \in c \wedge t \in l$.

Property 2 (Equivalence between GP_Π *and* GRC_Π*).* Let a position space $\Pi = (\mathcal{T}, <_h, <_v)$ and $GP_\Pi = (\mathcal{T}, \mathcal{A}, \mathcal{E})$, and $GRC_\Pi = (\mathcal{L}_\Pi \cup \mathcal{C}_\Pi, \mathcal{I})$ their associated representations. The following statements are equivalent for any pair $(t, t') \in \mathcal{T} \times \mathcal{T}$:

1. $t <_h t'$ (resp. $t <_v t'$)
2. $(t, t') \in \mathcal{A}$ and $\mathcal{E}((t, t')) = h$ (resp. $\mathcal{E}((t, t')) = v$)
3. $\exists l \in \mathcal{L}_\Pi, t \in l \wedge t' \in l$ (resp. $\exists c \in \mathcal{C}_\Pi, t \in c \wedge t' \in c$)

Equivalence relations between representations are defined as $\sim_\Pi$, i.e., $GP_\Pi \sim_\Pi GRC_\Pi$

The equivalence between 1. and 2. follows from Definition 4. To prove the equivalence between 1. and 3., we use Definition 3 of rows and columns. Indeed, $t <_h t'$ (resp. $t <_v t'$) implies that there exists a row (resp. column) l (resp. c) containing both t and t'. The converse is also evident from the definition of rows and columns.

We have two possible graph representations of position spaces, we will now study how they can be used to search for given patterns within these spaces.

2.3 Subgraph Isomorphism

Recall that our main goal is to detect specific patterns in position spaces. Our computational approach is grounded in graph representations. Hence, we aim at computing if a given pattern graph is isomorphic to a subgraph of a given position graph (or its row/column graph representation).

To simplify notation, the position space $\Pi = (\mathcal{T}, <_h, <_v)$ will be assumed known from here on, and we will therefore omit the Π indices.

[1] Since we will be working with both directed and undirected graphs, to avoid overloading vocabulary and notation, we write $(l, c) \in \mathcal{I}$ to denote that the edge (l, c) belongs to $\mathcal{I}$, just as for arcs. The context, usually clear, will make this standard distinction.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

Definition 6 (Subgraph Isomorphism of Position Graphs). *Let $GP = (\mathcal{T}, \mathcal{A}, \mathcal{E})$ be a position graph and $PP = (T, A, E)$ a position graph pattern. PP is isomorphic to a subgraph of GP if and only if there exists an injection $f : T \to \mathcal{T}$ such that:*

- $(t, t') \in A \Rightarrow (f(t), f(t')) \in \mathcal{A}$
- $(t, t') \in A \Rightarrow E((t, t')) = \mathcal{E}((f(t), f(t')))$.

Each injection f thus defines a subgraph $f(PP) = (f(T), \mathcal{A}_{f(T)}, \mathcal{E}_{f(T)})$ of GP, where $\mathcal{A}_{f(T)} = \{(t, t') \in \mathcal{A} | t, t' \in f(T)\}$ and $\mathcal{E}_{f(T)}$ is the restriction of $\mathcal{E}$ to $f(t)$. We then write $GP \triangleright_f PP$.

In Example 1, we can search in the graph GP for the pattern $M = (T = \{a, b, c, d\}, A = \{(a, b), (c, d), (a, c), (b, d)\}, E)$ with $E((a, b)) = E((c, d)) = h$ and $E((a, c)) = E((b, d)) = v$. A solution $\{a \mapsto 1, b \mapsto 2, c \mapsto 5, d \mapsto 6\}$ is illustrated in Figure 2b.

Since our representations are equivalent, we will define the subgraph isomorphism on row/column graphs.

Definition 7 (Subgraph Isomorphism of Row/Column Graphs). *Let $GRC = (\mathcal{L} \cup \mathcal{C}, \mathcal{I})$ be a row/column graph and $RCP = (RL \cup RC, I)$ a row/column graph pattern. RCP is isomorphic to a subgraph of GRC if and only if there exists an injection $f : RL \cup RC \to \mathcal{L} \cup \mathcal{C}$ such that:*

- $\forall l \in RL, f(l) \in \mathcal{L}$, $\forall c \in RC, f(c) \in \mathcal{C}$,
- $\forall (l, c) \in I, (f(l), f(c)) \in \mathcal{I}$.

Each injection f thus defines a subgraph $f(RCP) = (f(L) \cup f(C), \mathcal{I}_{|f(L) \cup f(C)})$ of GRC where $\mathcal{I}_{|f(L) \cup f(C)} = \{(f(l), f(c)) \in \mathcal{I}\}$. We then write $GRC \triangleright_f RCP$.

In example 1, searching for the pattern M shown in Figure 2b within the position graph corresponds to searching for the row/column graph induced by the M in the position graph modeled as row/column graph. We construct the row/column graph of both the pattern and position graphs according to Definition 5. For the pattern, we obtain sets $RL = \{\{a, b\}, \{c, d\}\}$ and $RC = \{\{a, c\}, \{b, d\}\}$ with $I = \{(\{a, b\}, \{a, c\}), (\{a, b\}, \{b, d\}), (\{c, d\}, \{a, c\}), (\{c, d\}, \{b, d\})\}$. Let the injection $fLC = \{\{a, b\} \mapsto \{1, 2\}, \{c, d\} \mapsto \{5, 6\}, \{a, c\} \mapsto \{1, 5\}, \{b, d\} \mapsto \{2, 6\}\}$. By construction we have $\forall l \in RL, fLC(l) \in \mathcal{L}$ and $\forall c \in RC, fLC(c) \in \mathcal{C}$. One readily verifies that $\forall (l, c) \in I, (f(l), f(c)) \in \mathcal{I}$. Thus fLC defines a subgraph $fLC(M)$ of GRC that is isomorphic to M.

We present now several formal results that establish equivalence, in terms of solutions, between the different patterns and their respective representations to justify the operational method that we will develop in Section 3.

Property 3 (Equivalence of Pattern Matching). Let $GP = (\mathcal{T}, \mathcal{A}, \mathcal{E})$ be a position graph, and $GRC = (\mathcal{L} \cup \mathcal{C}, \mathcal{I})$ a row/column graph, such that $GP \sim GRC$. Let $PP = (T, A, E)$ be a position pattern and $RCP = (RL \cup RC, I)$ a row/column pattern, such that $RL \subseteq 2^T$ and $RC \subseteq 2^T$. There exists an injection fP such that $GP \triangleright_{fP} PP$ if and only if there exists an injection fRC such that $GRC \triangleright_{fRC} RCP$ satisfying:

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

1. $\forall rl \in RL, \forall t \in T, (t \in rl) \Leftrightarrow fP(t) \in fRC(rl)$
2. $\forall rc \in RC, \forall t \in T, (t \in rc) \Leftrightarrow fP(t) \in fRC(rc)$
3. $\forall rl \in RL, \forall rc \in RC, \forall t \in T, rl \cap rc = t \Leftrightarrow fRC(rl) \cap fRC(rc) = fP(t)$

Let us sketch the proof: Given an injection f_P such that $GP \triangleright_{f_P} PP$, Conditions 1–3 ensure that each row or column mapped by f_{RC} contains exactly the images of the tokens mapped by f_P. Since $GP \sim GRC$, these images induce in GRC a subgraph matching the row/column pattern, making f_{RC} injective and yielding $GRC \triangleright_{f_{RC}} RCP$. The reverse direction follows symmetrically.

The following corollary follows from Property 3 and the definition of equivalences between models in Definition 6.

Corollary 1. *Let GP be a position graph and GRC a row/column graph such that $GP \sim GRC$, a position pattern PP and a row/column pattern RCP. If there exist injections fP and fRC such that $GP \triangleright_{fP} PP \wedge GRC \triangleright_{fRC} RCP$ and they satisfy the conditions of Property 3, then $fP(PP) \sim fRC(RCP)$.*

This corollary allows us to establish that searching for a pattern on the position graph GP is equivalent to searching on its associated row/column graph GRC. Property 3 and its corollary ensure consistency between patterns. However, it is important to note that, from an operational point of view, given a position pattern PP, a corresponding row/column pattern RCP must be constructed.

Given that $GP \sim GRC$, we then have a set of solutions $\mathcal{S}^{GP}_{MP} = \{fP \mid GP \triangleright_{fP} MP\}$ and a set of solutions $\mathcal{S}^{GRC}_{MLC} = \{fLC \mid GRC \triangleright_{fLC} MLC\}$ such that $\forall fP \in \mathcal{S}^{GP}_{MP}, \exists fLC \in \mathcal{S}^{GRC}_{MLC}, fP(MP) \sim fLC(MLC)$ and vice versa. However, the sets of solutions are not directly equivalent.

Let us return to our example 1. The pattern illustrated in Figure 2b and defined over $T = \{a, b, c, d\}$ leads us to search for a pattern MLC defined over $TL = \{\{a, b\}, \{c, d\}\}$ and $TC = \{\{a, c\}, \{b, d\}\}$ with $I = \{(\{a, b\}, \{a, c\}), (\{a, b\}, \{b, d\}), (\{c, d\}, \{a, c\}), (\{c, d\}, \{b, d\})\}$.

If we consider the solution $fP = \{a \mapsto 1, b \mapsto 2, c \mapsto 5, d \mapsto 6\}$ shown in Figure 2b, there are several possible solutions for this pattern in the graph GRC. The one we are interested in is $fLC = \{\{a, b\} \mapsto \{1, 2\}, \{c, d\} \mapsto \{5, 6\}, \{a, c\} \mapsto \{1, 5\}, \{b, d\} \mapsto \{2, 6\}\}$, but $fLC' = \{\{a, b\} \mapsto \{5, 6\}, \{c, d\} \mapsto \{1, 2\}, \{a, c\} \mapsto \{2, 6\}, \{b, d\} \mapsto \{1, 5\}\}$ is also a valid solution to the pattern matching problem.

However, this solution fLC' does not satisfy the conditions of Property 3, and therefore we do not have $fP(MP) \sim fLC'(MLC)$. In fact, we have lost in GRC the direct representation of the ordering information between tokens (relative vertical and horizontal positions) by moving from a directed graph to an undirected graph. Nevertheless, we can impose constraints during the pattern search in the row/column graph to extract, from the set of obtained solutions, those that match the original position pattern. We will describe, in operational terms, how these solution refinement constraints are taken into account during our pattern search.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

3 Patterns Search

In this section, we present a modeling of pattern matching in the form of CSPs and also describe a filtering algorithm that specifically addresses the subgraph isomorphism problem. Practical methods for subgraph isomorphism have been proposed in the context of constraint programming, using filter-and-search pipelines and constraint-programming–inspired pruning techniques. They are particularly effective for small pattern queries on large targets [6]. State-of-the-art solvers such as the Glasgow Subgraph Solver [7] integrate sophisticated constraint guided backtracking, support both induced and non-induced matching, and exploit parallelism. In contrast, LAD2025 [12] offers a modern constraint-based approach with strengthened domain filtering and propagation, achieving competitive performance on diverse benchmarks.

3.1 Position Patterns

Let $GP = (\mathcal{T}, \mathcal{A}, \mathcal{E})$ be a graph and $MP = (T, A, E)$ an associated pattern as defined in Definition 6.

We introduce, based on this definition, a constraint satisfaction problem $(\mathcal{X}, D, \mathcal{C})_P$ (variables, domains, constraints) that encodes the search for MP in GP. A decision variable is added for each node of MP : $\mathcal{X} = \{u_t \mid t \in T\}$. Their domains are the nodes of GP: $\forall u_t \in \mathcal{X}, D(u_t) = \mathcal{T}$. Next, we link the arcs of GP to those of MP by adding the following constraint in $\mathcal{C}$:

$$\forall u_t, u_{t'} \in \mathcal{X}, (t, t') \in A \Rightarrow (u_t, u_{t'}) \in \mathcal{A} \wedge \mathcal{E}((u_t, u_{t'})) = E((t, t'))$$

Finally, since this CSP encodes the search for the injective function fP from Definition 6, we add the well-known global constraint [10] on the set of decision variables: $Alldiff(\mathcal{X})$.

3.2 Row/column Patterns

We now consider a row/column graph $GRC = (\mathcal{L} \cup \mathcal{C}, \mathcal{I})$ and an associated row/column pattern $MLC = (TL \cup TC, I)$. Similarly to the previous case, we define a CSP $(\mathcal{X}, D, \mathcal{C})_{LC}$ related to the search for MLC in GRC. The basic model is similar to the model presented in Section 3.1.

$$\mathcal{X} = \{u_{lc} \mid lc \in TL \cup TC\}$$
$$\forall u_{lc} \in \mathcal{X}, \quad D(u_{lc}) = \mathcal{L} \cup \mathcal{C} \tag{1}$$
$$\forall u_{lc}, u_{lc'} \in \mathcal{X}, \quad (lc, lc') \in I \Rightarrow (u_{lc}, u_{lc'}) \in \mathcal{I} \tag{2}$$
$$\text{Alldiff}(\mathcal{X}) \tag{3}$$

In connection with the remarks at the end of Section 2.3, we introduce additional constraints to better establish the links between position patterns and row/column patterns.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

Type constraint The fact that GRC is bipartite with respect to the partition $\mathcal{L} \cup \mathcal{C}$ of its vertices allows us to restrict the domain definitions of each variable. Hence, we get the following constraint

$$\forall u_{lc} \in \mathcal{X},\ lc \in TL \ \Rightarrow\ D(u_{lc}) = \mathcal{L} \wedge lc \in TC \ \Rightarrow\ D(u_{lc}) = \mathcal{C} \tag{4}$$

Order constraint We define two order relations on rows and columns, naturally induced by the two alignment relations $<_h$ and $<_v$, first on complete rows and columns: $\forall l, l' \in \mathcal{L}^{comp}, l <_v l' \Leftrightarrow \exists t \in l, t' \in l', t <_v t'$ Similarly: $\forall c, c' \in \mathcal{C}^{comp}, c <_h c' \Leftrightarrow \exists t \in c, t' \in c', t <_h t'$.This order then canonically extends to all rows and columns as follows:

$$\forall l, l' \in \mathcal{L}, l <_v l' \Leftrightarrow \exists l^c, l'^c \in \mathcal{L}^{comp}, l \subseteq l^c \wedge l' \subseteq l'^c \wedge l^c <_v l'^c \tag{5}$$
$$\forall c, c' \in \mathcal{C}, c <_h c' \Leftrightarrow \exists c^c, c'^c \in \mathcal{C}^{comp}, c \subseteq c^c \wedge c' \subseteq c'^c \wedge c^c <_h c'^c \tag{6}$$

We then extend the pattern $MLC = (TL \cup TC, I)$ into an ordered row/column pattern $MLC^O = (TL \cup TC, I, O)$ with $O \subseteq (TL \times TL) \cup (TC \times TC)$, a set of pairs describing the order relations that must be respected between rows and columns. We can then add the following order constraint to our model:

$$\forall u_{lc}, u_{lc'} \in \mathcal{X},\ (lc, lc') \in O \ \Rightarrow\ (u_{lc} <_v u_{lc'} \ \vee\ u_{lc} <_h u_{lc'}) \tag{7}$$

Cardinality constraint The cardinality of a node n in the row/column graph is the number of tokens in the row(or column) that n represents. Thus we introduce the following constraint:

$$\forall u_{lc} \in \mathcal{X}, \quad D(u_{lc}) = \{\, l \in \mathcal{L} \mid |l| = |lc| \,\} \tag{8}$$

Informally, a node of the pattern graph can only be associated with a node of the same cardinality,that is with a node representing a row or column containing the same number of tokens.

4 Experimental results

4.1 Dataset

Our purpose is to generate a set of instances that are sufficiently representative of various position graphs and patterns. We consider 3 classes, each class containing 20 different graph instances. An instance corresponds to a given position graph and its associated row/column graph. A generator creates alignment relations among nodes randomly distributed in an $n \times n$ matrix. We explore many dimensions n of the matrix, yielding different graph densities. Each experiment consists of searching for a given pattern in two different target graphs:

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

– a position target graph GP, described in Definition 4 and thus directed, and
– the corresponding row/column graph GRC based on Definition 5 and thus undirected.

Table 1a summarizes the main characteristics of considered graphs. For position graphs GP: we consider the number of nodes $\#N_p$ and edges $\#E(GP)$, and maximum degree $d_{\max}$. Similarly for GRC.Variations within a class arise from the random generation process. GRC exhibits significantly higher density and degree due to the representation of all possible subrows/subcolumns.

Table 1: Summary of instances used in the experiments and their solutions.

(a) Description of the classes considered for the position and row/column graphs.

Class	Position graph GP			Row/Column graph GRC		
	#Np	#E(GP)	dmax	#Nrc	#E(GRC)	dmax
A	40	131-160	13	392-1078	5576-47652	414
B	200	753-840	18	3018-6275	93102-931394	1300
C	600	1747-1886	18	6163-10330	109710-587342	1302

(b) Number of solutions and standard deviation per class.

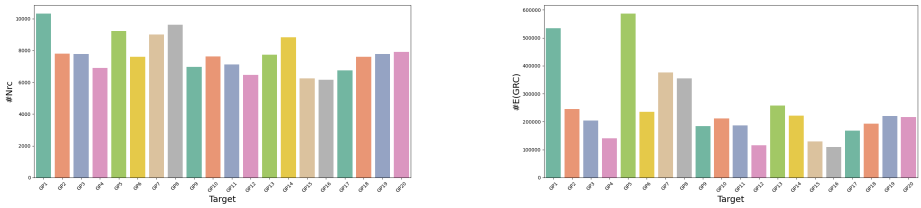
	Sols		
	A	B	C
pan	50.9 ± 24.9	77.0 ± 19.1	17.0 ± 6.9
gap	46.2 ± 10.8	80.5 ± 24.5	17.2 ± 9.3
net	1.9 ± 2.6	0.1 ± 0.2	0.0 ± 0.0
bowtie	16.1 ± 12.7	8.2 ± 4.3	0.2 ± 0.4
ladder	14.7 ± 24.5	6.2 ± 7.5	0.0 ± 0.0

If we consider a position graph GP whose horizontal alignment relation partitions the N tokens into r complete rows of lengths $k_1, k_2, \ldots, k_r$, and whose vertical alignment relation partitions them into c complete columns of lengths $h_1, h_2, \ldots, h_c$. The number of vertices of its corresponding GRC is given $\sum_{i=1}^{r}(2^{k_i}-1) + \sum_{j=1}^{c}(2^{h_j}-1)$. Since any token t belongs to the complete row $i(t)$ of length $k_{i(t)}$ and to the complete column $j(t)$ of length $h_{j(t)}$, the number of edges of GRC is $\sum_{t=1}^{N} 2^{k_{i(t)}+h_{j(t)}-2}$. In the worst case, we have $k_1 = h_1 = N$ and the number of vertices is $2^{N+1}-2$ with $N.2^{2N-2}$ edges. Figure 3 illustrates how the size of the row/column graph changes for different position graphs in class C.

In our experiments, we propose five pattern graphs (they have been inspired by structural configurations encountered in invoice layouts). We recall that each time we search for a position graph we search in parallel for its corresponding row/column representation. Hence each pattern is associated with a row/column pattern used in the problem of subgraph isomorphism of row/column graphs (see Definition 7).

Table 1b gives for each pattern and each class, the average number of solutions as well as its standard deviation among the 20 instances of the class.We remark that the number of solutions may exhibit a rather large standard deviation, since it varies with the chosen pattern, as illustrated in Figure 4. For instance, we see that on Class A, the pattern "pan" is more frequent than the pattern "net", which requires a larger structure involving 9 nodes.

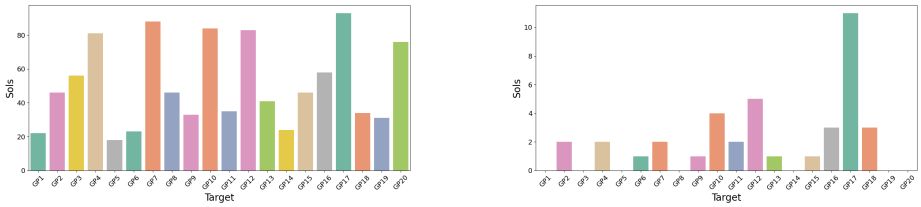
ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24



(a) Variation of the number of nodes.

(b) Variation of the number of edges.

Fig. 3: Variation in the dimensions of the row/column graph with respect to the distribution of node in the position graph in the class C.



(a) Pattern graph pan in 40-node position graphs

(b) Pattern graph net in 40-node position graphs

Fig. 4: Variation in the number of solutions across different instances of 40-node graphs.

4.2 Experimental Setup

All experiments were conducted using PyCSP3 [5] and the solver ACE in a base configuration. We fix a one hour timeout for the solving process; experiments exceeding the timeout are considered failures.

To evaluate the efficiency of the solver we use the following classic measures on the search trees [1]:

- **Effective propagations (Effs):** number of efficient propagator activations that do not lead to an inconsistency.
- **Wrong decisions (Wrgs):** number of incorrect choices made by the solver. In particular, it takes the value 0 when the solver explores the search space without making any wrong decisions.

Filtering Algorithm for Subgraph Isomorphism : To reduce the search effort before CSP solving, we apply the Iterative Labeling Filtering (ILF) algorithm introduced in [15] for subgraph isomorphism. ILF assigns each vertex of both the pattern and the target graph a label based on its in- and out-degrees $(\deg^-, \deg^+)$, defines a partial order over these labels, and prunes any candidate mappings that violate this ordering.

From these initial label constraints, ILF iteratively enforces neighborhood consistency: a candidate vertex remains valid only if its labeled neighbors ad-

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

mit compatible matches respecting the adjacency structure of the pattern. This iteration continues until a fixpoint is reached, at which point the CSP solver proceeds using the remaining domains together with the standard constraints of our encoding, including the global `Alldiff` constraint. Although originally designed for general SIP, ILF naturally adapts to our directed position graphs and bipartite row/column graphs by leveraging their structural properties.

Solving Configuration : We use two possible configurations of the solving process. ILF can be used as a pre-processing filtering algorithm that reduces the domains before running the solver. On the other side, we may introduce additional constraints to the basic model to enhance the solver's performance.

4.3 Analysis of the Results

We first study the impact of ILF on the position graphs GP (Table 2).

Table 2: Results on position graphs for classes A, B and C: effectives (Effs), wrong decisions (Wrgs), and runtime per pattern.

	ILF	A			B			C		
	Used	Effs ($\times 10^4$)	Wrgs ($\times 10^2$)	Runtime(s)	Effs ($\times 10^7$)	Wrgs ($\times 10^4$)	Runtime(s)	Effs ($\times 10^8$)	Wrgs ($\times 10^4$)	Runtime(s)
pan	yes	4.1 ± 1.5	4.1 ± 1.9	0.6	2.1 ± 0.5	2.0 ± 0.3	21.5	1.2 ± 0.9	5.1 ± 3.5	213.1
	no	6.3 ± 2.5	5.5 ± 2.4	0.6	3.3 ± 1.1	2.7 ± 0.6	37.1	2.2 ± 1.1	7.6 ± 2.9	512.4
gap	yes	6.1 ± 2.4	8.1 ± 3.5	0.7	4.6 ± 2.0	4.8 ± 1.5	40.2	1.9 ± 0.5	10.0 ± 2.0	278.9
	no	8.6 ± 2.2	10.0 ± 4.0	0.7	6.4 ± 2.2	5.9 ± 1.4	59.2	4.3 ± 2.0	18.0 ± 7.0	696.8
net	yes	2.4 ± 1.4	2.0 ± 1.1	0.6	1.6 ± 0.7	1.1 ± 0.5	20.7	0.11 ± 0.09	0.46 ± 0.38	43.4
	no	11.0 ± 8.0	5.3 ± 4.8	0.8	9.8 ± 7.5	4.8 ± 3.6	139.4	3.4 ± 2.3	6.9 ± 4.8	1006.1
bowtie	yes	5.9 ± 3.5	4.8 ± 2.9	0.7	4.6 ± 2.0	3.2 ± 1.4	54.6	1.9 ± 1.4	5.7 ± 3.8	478.1
	no	9.3 ± 4.2	5.3 ± 2.6	0.7	7.3 ± 3.3	4.2 ± 1.7	96.4	3.6 ± 2.7	8.2 ± 5.6	1137.4
ladder	yes	2.7 ± 2.0	2.2 ± 1.6	0.6	1.8 ± 1.2	1.5 ± 0.9	22.5	0.69 ± 0.43	2.4 ± 1.3	173.7
	no	7.0 ± 3.6	3.9 ± 2.2	0.6	9.3 ± 4.9	5.1 ± 2.3	126.4	4.3 ± 2.2	10.0 ± 5.0	1192.5

Across all classes, ILF consistently reduces the search effort: *Effs* decrease by roughly 50% in class A and up to 70% in class C, while *Wrgs* drop by 30–50%. Correspondingly, runtime improves from a marginal gain in class A (0.7 s $\rightarrow$ 0.6 s) to nearly a fourfold reduction in class C (910 s $\rightarrow$ 235 s). This highlights a modeling-dependent interaction: ILF becomes more effective as node degrees diversify and graph density increases. Let us note that the pattern topology plays a significant role: in class B, ILF reduces the runtime of patterns such as *net* and *ladder* by factors of 5–7, while in class A the effect remains mild due to limited degree variability. Figure 5 shows a systematic reduction of Effs when ILF is enabled, confirming that the filtering reduces propagation work on a per-instance basis, not only on average. Moreover, this figure highlights the heterogeneity of instance difficulty: even without ILF, *Effs* vary widely across targets, reflecting the variability in the random generator. ILF compresses this spread and lowers the overall scale of *Effs*, demonstrating its ability to stabilize the search effort by eliminating many incompatible candidate mappings before search.

We now turn to the row/column graphs GRC. Table 3 shows that without structural constraints (typing, ordering, cardinality), no instance is solved within

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

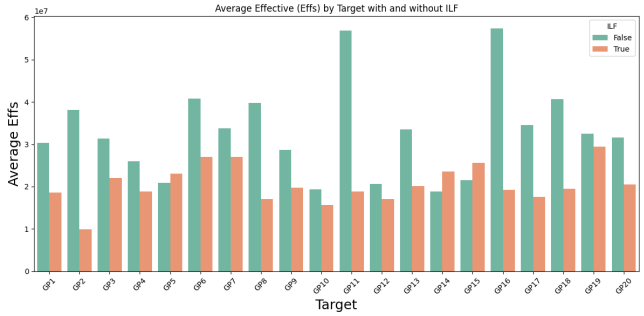


Fig. 5: Effectives per target for the pattern graph Pan with ILF(in orange) and without ILF(in green), the position graphs are of 200 nodes.

the timeout, whereas *all constraints + no ILF* yield a 100% success rate for all classes. The cardinality constraint (Eq. 8) is particularly decisive (e.g., class B: 20% → 100% solved).

Table 3: Evaluation of performance of the solver on row/column graph : Solved instances: study on the role of the model constraints (csts), of cardinality constraint card and of ILF.

	Solved Instances (%)				
	no ILF, no csts	ILF all csts	ILF no card	no ILF no card	no ILF all csts
A	0	100	90	95	100
B	0	0	0	20	100
C	0	0	0	0	100

With all constraints but without ILF (Table 4), solving remains efficient (typically 1–3 s), despite GRC being much larger than GP. In contrast, removing the cardinality constraint inflates both *Effs* and runtime by one to two orders of magnitude.

Comparing both models, we observe that GP is compact and benefits strongly from ILF, especially on medium/large instances. Without ILF, runtimes can reach the hundreds-to-thousand-second range in class C. GRC is larger but, once structural constraints (typing, ordering, cardinality) are enabled, solving becomes surprisingly efficient (often in a few seconds), whereas turning off these constraints renders most instances intractable within the timeout. Hence, GP leverages pre-processing (ILF) for pruning while GRC leverages internal model structure (constraints). The two representations thus offer complementary strengths. We observe large standard deviations for Effs and sometimes for Wrgs, especially in classes B–C , indicating wide instance-to-instance variability from the generator. In spite of this spread, the mean reductions under ILF (for

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

Table 4: Results on row/column graphs when ILF is not used with all constraints (Section 3.2) (first table) and without cardinality (second table).

	A			B			C		
	Effs ($\times 10^3$)	Wrgs ($\times 10^2$)	Runtime(s)	Effs ($\times 10^4$)	Wrgs ($\times 10^3$)	Runtime(s)	Effs ($\times 10^4$)	Wrgs ($\times 10^3$)	Runtime(s)
pan	1.5 ± 0.3	1.2 ± 0.4	0.5	1.5 ± 0.2	1.4 ± 0.3	3.0	2.6 ± 0.2	1.5 ± 0.3	3.1
gap	1.6 ± 0.3	1.2 ± 0.5	0.5	1.5 ± 0.2	1.3 ± 0.2	2.8	2.6 ± 0.2	1.5 ± 0.3	3.0
net	1.3 ± 0.3	0.6 ± 0.3	0.4	1.4 ± 0.2	0.4 ± 0.1	1.9	2.4 ± 0.2	0.4 ± 0.1	2.6
bowtie	1.5 ± 0.4	1.0 ± 0.4	0.5	1.6 ± 0.2	0.8 ± 0.3	2.1	2.4 ± 0.2	0.4 ± 0.1	2.6
ladder	1.2 ± 0.5	0.5 ± 0.5	0.4	0.95 ± 0.2	0.5 ± 0.2	2.2	1.3 ± 0.2	0.4 ± 0.1	3.1

	A			B			C		
	Effs ($\times 10^5$)	Wrgs ($\times 10^3$)	Runtime (s)	Effs ($\times 10^5$)	Wrgs ($\times 10^3$)	Runtime (s)	Effs ($\times 10^5$)	Wrgs ($\times 10^3$)	Runtime (s)
pan	2.2 ± 1.6	5.7 ± 2.7	10.6	13.0 ± 8.0	78.0 ± 66.0	175.1			
gap	2.5 ± 2.5	4.8 ± 5.4	23.0						
net	0.094 ± 0.213	0.61 ± 1.12	1.4						
bowtie	1.3 ± 1.4	5.2 ± 4.9	7.4						
ladder	1.3 ± 1.7	5.6 ± 8.1	15.2						

GP) and under structural constraints (for *GRC*) remain consistent in direction and large in magnitude.

5 Conclusion

We introduced a formal framework for alignment-based position spaces and two complementary graph abstractions: the directed position graph and the bipartite row/column graph. These representations capture the combinatorial structure of horizontal and vertical alignments and provide a unified basis for defining and detecting patterns through subgraph isomorphism. Using CSP encodings and an existing filtering technique, we analyzed how modeling choices and structural constraints influence propagation and search. Experiments show that ILF significantly reduces search effort on position graphs, whereas for row/column graphs, type, order, and cardinality constraints are essential for tractable solving. The complementary behaviors of the two models suggest promising hybrid approaches, especially via cross-model propagation.

As a future work we plan to explore a CSP model that aims to solve the isomorphism problem on the position graph and its corresponding row/column graph simultaneously with cross-model constraint propagation. Propagating the filtering at each node from one model to the other may accelerate convergence to a fixpoint.

References

1. Christian Bessiere, Bruno Zanuttini, and Cèsar Fernàndez. Measuring Search Trees. *Workshop on Modelling and Solving Problems with Constraints - ECAI'2004*, pages 31–40, Valencia, Spain, 2004.
2. M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24

3. Bo Huang, Thomas Cova, Ming-Hsiang Tsou, Georg Bareth, Chunqiao Song, Yan Song, Kai Cao, and Elisabete Silva. *Comprehensive Geographic Inf. Sys.*, 2017.
4. Lars Kotthoff, Ciaran McCreesh, and Christine Solnon. Portfolios of subgraph isomorphism algorithms. In Paola Festa, Meinolf Sellmann, and Joaquin Vanschoren, editors, *Learning and Intelligent Optimization*, volume 10079 of *Lecture Notes in Computer Science*, pages 107–122. Springer, 2016.
5. Christophe Lecoutre and Nicolas Szczepanski. Pycsp3: Modeling combinatorial constrained problems in python, 2024.
6. Ciaran McCreesh, Patrick Prosser, Christine Solnon, and James Trimble. When subgraph isomorphism is really hard, and why this matters for graph databases. *Journal of Artificial Intelligence Research*, 61:723–759, 2018.
7. Ciaran McCreesh, Patrick Prosser, and James Trimble. The glasgow subgraph solver: Using constraint programming to tackle hard subgraph isomorphism problem variants. In *13th International Conference, ICGT 2020*, volume 12150 of *LNCS*, pp 316–324. Springer, 2020.
8. Charalampos Nikolaou and Manolis Koubarakis. Fast consistency checking of very large real-world rcc-8 constraint networks using graph partitioning. *Proceedings of the National Conference on Artificial Intelligence*, 4:2724–2730, 06 2014.
9. David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In *Principles of Knowledge Representation and Reasoning*, KR, page 165–176, 1992.
10. Jean-Charles Régin. A filtering algorithm for constraints of difference in csps. In Barbara Hayes-Roth and Richard E. Korf, editors, *Proceedings of the 12th National Conference on Artificial Intelligence, Volume 1*, pages 362–367, 1994.
11. Thomas Saout, Frédéric Lardeux, and Frédéric Saubion. An overview of data extraction from invoices. *IEEE Access*, 12:19872–19886, 2024.
12. Christine Solnon. Lad2025, A constraint-based solver for the subgraph isomorphism problem. *Artif. Intell.*, 352:104474, 2026.
13. Julian R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976.
14. Yiyuan Wang, Chenghou Jin, and Shaowei Cai. Pathlad+: Towards effective exact methods for subgraph isomorphism problem. *Artif. Intell.*, 337:104219, 2024.
15. Stéphane Zampelli, Yves Deville, and Christine Solnon. Solving subgraph isomorphism problems with constraint programming. *Constraints*, 15(3):327–353, 2010.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29921-5_24