Theoretical and Algorithmic Framework for Hypergraph Matching

Horst Bunke[1], Peter Dickinson[2], and Miro Kraetzl[2]

[1] Institut für Informatik und angewandte Mathematik,
Universität Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland
bunke@iam.unibe.ch
[2] Intelligence Surveillance Reconnaissance Division, Defence Science and
Technology Organisation, Edinburgh SA 5111, Australia
{Peter.Dickinson, Miro.Kraetzl}@dsto.defence.gov.au

Abstract. Graphs have been successfully used in many disciplines of science and engineering. In the field of pattern recognition and image analysis, graph matching has proven to be a powerful tool. In this paper we generalize various matching tasks from graphs to the case of hypergraphs. We also discuss related algorithms for hypergraph matching.

1 Introduction

Graphs have been successfully used in many disciplines of science and engineering. Recent work in structural pattern recognition has resulted in a number of new graph matching algorithms that can be used to compute the similarity or, equivalently, the distance of a given pair of graphs. This led to many interesting applications of graph matching, including document analysis [1], shape recognition [2], biometric identification [3], and other tasks. For a recent survey that covers both methodology and applications of graph matching see [4].

Although many successful applications of graph matching have been reported in the literature, graphs are restricted in the sense that only binary relations between nodes can be represented, through graph edges. An extension is provided by hypergraphs, where each edge is a subset of the set of nodes [5]. Hence higher-order relations between nodes can be directly modeled in a hypergraph, by means of hyperedges. A large body of theoretical work on hypergraphs has been published. However, not many applications in the field of image processing and pattern recognition involving hypergraphs have been reported. Ref. [6] lists a number of hypergraph applications in low level image processing, and [7] describes a 3-D object recognition system using hypergraphs. We notice in particular that there is a lack of hypergraph matching algorithms. It seems that only the problems of maximum common sub-hypergraph [8] and hypergraph monomorphism [7] have been considered in the literature until now.

In this paper, in Section 2, we formally introduce hypergraphs as an extension of ordinary graphs. Then, in Section 3, we generalize a number of matching concepts from ordinary graphs to the case of hypergraphs. Algorithmic procedures

F. Roli and S. Vitulano (Eds.): ICIAP 2005, LNCS 3617, pp. 463–470, 2005.

for actually performing hypergraph matching are discussed in Section 4. Finally, Section 5 concludes this paper and provides suggestions for future research.

2 Preliminaries

Let L_V and L_E be finite sets of labels for the nodes and edges of a graph, or the nodes and hyperedges of a hypergraph, respectively.

Def. 2.1: A graph is a 4-tuple $g = (V, E, \alpha, \beta)$, where V is the finite set of nodes (also called vertices), $E \subseteq V \times V$ is the set of edges, $\alpha : V \rightarrow L_V$ is a function assigning labels to nodes, and $\beta : E \rightarrow L_E$ is a function assigning labels to edges.

The edges of a graph can be interpreted as unary or binary relations. Unary relations correspond to loops, i.e. edges $(x, x) \in E$, while binary relations correspond to directed edges of the form $(x, y) \in E$, $x \neq y$. Hypergraphs are a generalization of ordinary graphs in the sense that higher-order relations between nodes can be modeled.

Def. 2.2: Let $N \geq 1$ be an integer. A *hypergraph* of order N is a 4-tuple $h = (V, \mathcal{E}, \alpha, \mathcal{B})$, where V is the finite set of nodes (also called vertices), $\mathcal{E} = \cup_{i=1}^{N} E_i$ with $E_i \subseteq V^i$ is the set of hyperedges $(i = 1, \ldots, N)$, $\alpha : V \rightarrow L_V$ is a function assigning labels to nodes, and $\mathcal{B} = \{\beta_1, \ldots, \beta_N\}$ is the set of hyperedge labeling functions with $\beta_i : E_i \rightarrow L_E$.

Each E_i is a subset of the set of hyperedges. It consists of i-tuples $(x_1, \ldots, x_i) \in V^i$, where each i-tuple is a hyperedge of hypergraph h. We call i the order of hyperedge $e = (x_1, \ldots, x_i)$. The elements of E_1 are the loops of the hypergraph and the elements of E_2 correspond to the edges of a (normal) graph. A hyperedge of degree higher than two can be used to model higher-order relations among the nodes. Note that graphs according to Def. 2.1 are a special case of Def. 2.2 if $N = 2$.

There are several possibilities to graphically represent a hypergraph. In [5] it was proposed to draw a node as a point, an edge from subset E_1 as a loop, an edge from subset E_2 as a line segment connecting the pair of nodes involved, and edges of higher degree as simple closed curves that enclose the corresponding nodes. In this paper we adopt a different graphical notation, where circles are used to graphically represent nodes, and ellipses are used to represent hyperedges of degree three and higher. If $e = (x_1, \ldots, x_i) \in E_i$, $i \geq 3$, then we draw i line segments connecting the hyperedge symbol and $x_1, \ldots, x_i$, respectively. Labels are written next to nodes or next to hyperedge symbols.

Several hypergraph formalisms have been proposed in the literature. For a more detailed discussion of how these formalisms are related to the current paper we refer the reader to [9].

We conclude this section with a few examples that illustrate how hypergraphs can be used in pattern recognition and image analysis. These examples are also

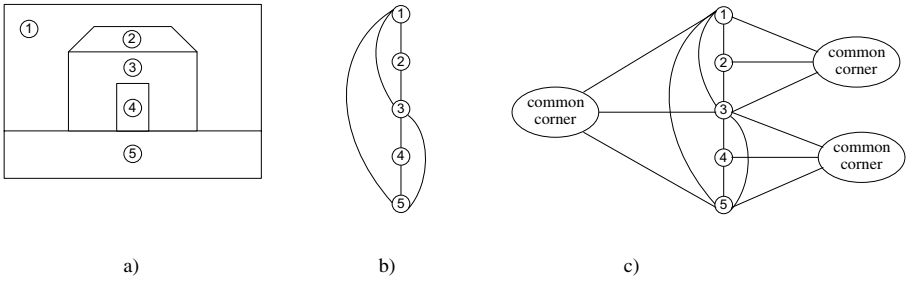

Fig. 1. a) image after segmentation; b) region adjacency graph; c) hypergraph with hyperedges representing region adjacency of order two and three

intended to show that hypergraphs have a higher representational power than normal graphs.

Example 2.1: In pattern recognition and computer vision, *region adjacency graph* (rag) is a popular datastructure to formally represent the contents of an image after it has been segmented into homogeneous regions. In a *rag* nodes represent image regions and edges between nodes indicate whether two regions are adjacent to each other or not. Fig. 1a shows an image that has been segmented into homogeneous regions, and Fig. 1b shows the corresponding *rag*. For certain applications it may be interesting to know all the regions that meet at a common corner in the image. Such a relation among three or more regions can't be directly represented in a normal graph. But in a hypergraph it is straightforward to model relations of this kind by means of hyperedges. Fig. 1c shows a hypergraph that corresponds to Fig. 1a and includes region adjacency of degree three.[1] □

Example 2.2: Wireframe models are a common representation in 3-D object modeling and computer vision. Fig. 2a shows a polyhedral object and Fig. 2b the corresponding wireframe model, given in terms of a graph. In this representation graph nodes represent the vertices of the polyhedron, and graph edges correspond to the edges of the polyhedron. Note that the graph in Fig. 2b only captures the topology of the object, and doesn't include any quantitative, metric information. In a graph, such as the one depicted in Fig. 2b, only binary relations between the vertices of an object can be represented. In a hypergraph it is easy to directly model relations of higher order, such as collinearity and coplanarity, by means of hyperedges. Fig. 2c shows an extended version of Fig. 2b, where the relation of collinearity has been added. □

3 Hypergraph Matching

Graph matching is the task of establishing a correspondence between the nodes and edges of two given graphs such that some constraints are satisfied [4]. Well-

[1] Actually there are two instances of each relation of degree 3 in the image. However, this observation is not modeled in Fig. 1c, i.e. only one instance is included.

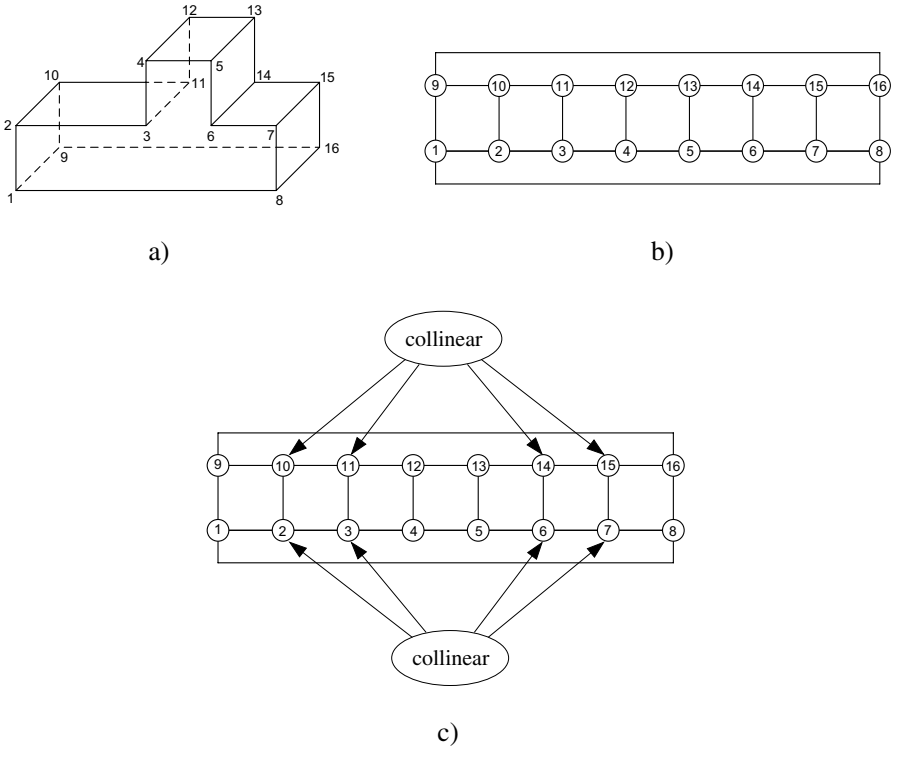

Fig. 2. a) a polyhedral object; b) graph representation; c) hypergraph showing collinear vertices

known instances of the graph matching problem include graph and subgraph isomorphism [10], maximum common subgraph computation [11] and graph edit distance [12]. In this section we'll extend these matching problems from graphs to the case of hypergraphs.

Def. 3.1: Let $h = (V, \alpha, \mathcal{E}, \mathcal{B})$ and $h' = (V', \alpha', \mathcal{E}', \mathcal{B}')$ be two hypergraphs of order N and N', respectively. We call h a *sub-hypergraph* of h' if $V \subseteq V'$; $E_i \subseteq E'_i$ for $i = 1, \ldots, N$; $\alpha(x) = \alpha'(x)$ for all $x \in V$; $\beta_i(e) = \beta'_i(e)$ for all $e \in E_i$ and $i = 1, \ldots, N$.

According to Def. 3.1 the inclusion of a hyperedge in a sub-hypergraph requires all its nodes being present as well. Otherwise the hyperedge will be dropped from the sub-hypergraph.

Def. 3.2: Let h and h' be two hypergraphs with $N = N'$. A *hypergraph isomorphism* between h and h' is a bijective mapping $f : V \to V'$ such that $\alpha(x) = \alpha'(f(x))$ for all $x \in V$; for $i = 1, \ldots, N$ and any hyperedge $e = (x_1, \ldots, x_i) \in E_i$ there exists a hyperedge $e' = (f(x_1), \ldots, f(x_i)) \in E'_i$ such that $\beta_i(e) = \beta'_i(e')$ and for any hyperedge $e' = (f(x_1), \ldots, f(x_i)) \in E'_i$ there exists a hyperedge $e = (f^{-1}(x_1), \ldots, f^{-1}(x_i)) \in E_i$ such that $\beta'_i(e') = \beta_i(e)$.

If $f : V \rightarrow V'$ is a hypergraph isomorphism between two hypergraphs, h and h', and h' is a sub-hypergraph of another hypergraph, h'', then f is called a *sub-hypergraph isomorphism* from h to h''.

Def. 3.3: Let h and h' be two hypergraphs. A *common sub-hypergraph* of h and h' is a hypergraph h'' such that there exist sub-hypergraph isomorphisms from h'' to h and from h'' to h'. We call h'' a *maximum common sub-hypergraph* of h and h', if there exists no other common sub-hypergraph of h and h' that has more nodes and, for a fixed number of nodes, more hyperedges than h''. □

Next we like to mention that it is straightforward to extend edit distance from the case of normal graphs to hypergraphs. Because of limited space we give only an informal description here. For a complete and formal treatment see [9]. Given two hypergraphs, h and h', and a set of edit operations with associated costs, one can transform h into h' by application of a suitable sequence of edit operations. The edit distance of hypergraphs h and h' is defined as the cost of the cheapest sequence of edit operations that transform h into h', i.e. $d(h, h') = min_S\{c(S) \mid S \text{ is a sequence of edit operations transforming } h \text{ into } h'\}$.

The edit distance, $d(h, h')$, measures the difference, or distance, of a pair of hypergraphs, h and h'. Clearly, if h and h' are isomorphic then $d(h, h') = 0$. In general, the greater the dissimilarity between h and h' is, the more edit operations are needed to transform h into h' and the larger the edit distance becomes.

In ordinary graph matching, a few other distance measures have been proposed. They are based on the maximum common subgraph of a pair of graphs. One of those measures [13] is defined as

$$d(g, g') = 1 - \frac{|mcs(g,g')|}{max(|g|,|g'|)} \tag{3.1}$$

In this definition, $|g|$ denotes the size of graph g, for example, the number of nodes, and $mcs(g, g')$ is the maximum common subgraph of graphs g and g'. Clearly this definition can be applied to hypergraphs as well if we replace maximum common subgraph by maximum common sub-hypergraph.

There are applications where one needs to represent a set of graphs by a single prototype. In [14] the median graph has been introduced to accomplish this task. Formally, the median of a set of graphs, $G = \{g_1, \ldots, g_k\}$, is a graph, $\overline{g}$, that satisfies

$$\sum_{i=1}^{k} d(\overline{g}, g_i) = \min\{\sum_{i=1}^{k} d(g, g_i) \mid g \in U\} \tag{3.2}$$

In this definition, U is the set of all graphs with node and edge labels from L_V and L_E, respectively. Hence the median is a graph that has, among all graphs with labels from L_V and L_E, the smallest possible average edit distance to the members of the given set, G.

Median graph computation has a very high computational complexity. However, if we restrict set U to be identical to G, we just need to compute all pairwise

distances $d(g_i, g_j)$, where $i, j = 1, \ldots, k$ and $i \neq j$, and select that graph g_i from set G that has the smallest sum of distances. Obviously, this procedure can be directly generalized from graphs to hypergraphs. We only need to replace graph distance by hypergraph distance.

4 Algorithms for Hypergraph Matching

In the previous section, a number of theoretical concepts have been introduced. However, no algorithmic procedures were considered. In the current section we'll discuss possible algorithms for hypergraph matching. For the matching of normal graphs, many algorithms have been proposed in the literature. They are based on various computational paradigms, including combinatorial search, neural networks, genetic algorithms, graph eigenvalue decomposition, and others. For a recent survey we refer to [4].

We start with the problem of extending graph and subgraph isomorphism computation to the case of hypergraphs. One of the best known graph matching algorithms, which can be used for both graph and subgraph isomorphism detection, is the one by Ullman [10]. It is a combinatorial search procedure that explores all possible mappings between the nodes of the two graphs under consideration. In order to avoid the exploration of partial mappings that can't lead to a correct solution, a look-ahead strategy is used. In this section, we'll discuss a generalization to of this algorithm to hypergraph matching. We refer again to [9] for more details.

Given two graphs, g_1 and g_2, that need to be tested for subgraph isomorphism, Ullmann's algorithm sequentially maps each node, x, of g_1 to a node, y, of g_2 and checks a number of constraints. Let $x \in V_1$, $y \in V_2$ and $f : V_1 \rightarrow V_2$ the mapping being constructed. The partial mapping constructed up to a certain point in time is augmented by $f(x) = y$ if the following three constraints are satisfied:

1. There exists no node $x' \in V_1$, $x' \neq x$, with $f(x') = y$, i.e. no other node of g_1 has already been assigned to y
2. Nodes x and y have the same label
3. The assignment $f(x) = y$ is compatible with all previous node assignments under function f, i.e. if the assignment $f(u) = v$ has been made before and there is an edge (x, u) or (u, x) in g_1, there must be an edge $(f(x), f(u))$ or $(f(u), f(x))$ in g_2 with the same label

To extend Ullmann's algorithm to the case of hypergraphs, we adopt constraints 1 and 2 without any changes. Only constraint 3 needs to be generalized in the sense that not only compatibility with respect to all edges, but w.r.t. all hyperedges is checked. Clearly such an extension is straightforward to implement.

A significant speedup in Ullmann's algorithm is achieved through the use of a lookahead technique. The basic idea is to maintain a *future match* table where all possible future node assignments are recorded. Initially, all pairs

$(x, y) \in V_1 \times V_2$ where x and y have identical node labels are possible. During the course of the search, pairs that violate Constraint 3 are eliminated. Consequently the number of assignments to be investigated in the tree search procedure is reduced. Obviously this lookahead procedure can be integrated in our subhypergraph matching schema. Assume that $x_1, \ldots, x_n \in V_1$ have been mapped to $f(x_1), \ldots, f(x_n) \in V_2$. Once a new assignment $f(x) = y$ has been made, we inspect all future (i.e. remaining) nodes $u_1, \ldots, u_m \in V_1$ and $v_1, \ldots, v_k \in V_2$ and delete any pair (u_i, v_j) from the future match table if nodes $x_1, \ldots, x_m, x, u_i \in V_1$ are part of a hyperedge in h_1 but $f(x_1), \ldots, f(x_n), f(x), v_j$ are not part of an hyperedge with the same label and order in h'.

The computational paradigm underlying Ullmann's algorithm is tree search. Also the problem of maximum common subgraph and graph edit distance computation can be solved by means of the tree search [11]. In case of maximum common subgraph computation an isomorphism between the first graph, g_1, and the second graph, g_2, is constructed that has the maximum possible size. This procedure can be extended from graphs to hypergraphs by extending the consistency checks in the common subgraph from edges to hyperedges. In case of graph edit distance computation a mapping of the nodes of g_1 to the nodes of g_2 is constructed such that the cost of the edit operations implied by this mapping is minimized. This procedure can be generalized to hypergraphs by extending edit operations on the edges to hyperedges.

5 Conclusions

Graphs has become a well-established representation formalism in pattern recognition. There is a large number of applications where graphs and graph matching algorithms have been used successfully [4]. Nevertheless, graphs are restricted in the sense that only two-dimensional relations can be modeled. In this paper we have investigated a more general framework that is based on hypergraphs.

Hypergraphs allow us to model not only binary relations, but relations of any finite order, and include graphs as a special case. A fundamental requirement for any graph-based formalism in pattern recognition is the availability of related graph matching algorithms. For the case of normal graphs, such algorithms exist. Examples are isomorphism, subgraph isomorphism, maximum common subgraph, and graph edit distance computation. On top of such algorithms, classification and clustering procedure can be implemented. In this paper we show that similar matching algorithms can be designed for the domain of hypergraphs. This makes the enhanced representational power of hypergraphs available for a potentially large number of practical pattern recognition applications.

The main purpose of this paper was to introduce a theoretical and algorithmic framework for hypergraph matching. Our future work will be concerned with practical implementations of the methods proposed in this paper. Also we plan to conduct practical experiments to study the computational behavior (time and space complexity) of the proposed algorithms and compare them to classical graph matching.

References

1. Llados, J., Marti, E., Villanueva, J.: Symbol recognition by error-tolerant subgraph matching between region adjacency graphs, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 23-10, 2001, 1137-1143
2. Luo, B., Hancock, E.: Structural graph matching using the EM algorithm and singular value decomposition, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 23-10, 2001, 1120-1136
3. Marcialis, G., Roli, F., Serrau, A.: Fusion of statistical and structural fingerprint classifiers. In Kittler, J., Nixon, M., eds.: *4th Int. Conf. Audio- and Video-Based Biometric Person Authentication*, Springer LNCS 2688, 2003, 310-317
4. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition, *Int. Journal of Pattern Recognition and Art. Intelligence*, Vol. 18, No. 3, 2004, 265-298
5. Berge, C.: Hypergraphs, North-Holland, 1989
6. Bretto, A., Cherifi, H., Aboutajdine, D.: Hypergraph imaging: an overview, Pattern Recognition, Vol. 35, No. 3, 2002, 651-658
7. Wong, A.C.K., Lu, S.W., Rioux, M.: Recognition and shape synthesis of 3-D objects based on attributed hypergraphs, *IEEE Trans. PAMI*, Vol. 11, No. 3, 1989, 279-290
8. Demko, D.: Generalization of two hypergraphs. Algorithm of calculation of the greatest sub-hypergraph common to two hypergraphs annotated by semantic information. In Jolion, J.-M., Kropatsch, W. (eds.): *Graph Based Representations in Pattern Recognition, Computing*, Supplement 12, Springer Verlag, 1998, 1-10
9. Bunke, H., Dickinson, P., Kraetzl, M.: Matching Hypergraphs, *Technical Report*, Intelligence Surveillance Reconnaissance Division, Defence Science and Technology Organisation, Edinburgh SA 5111, Australia, 2004
10. Ullman, J.R.: An algorithm for subgraph isomorphism, *Journal of ACM*, Vol. 23, 1976, 31-42
11. McGregor, J.J.: Backtrack search algorithm and the maximal common subgraph problem, *Software – Practice and Experience*, Vol. 12, 1982, 23-34
12. Messmer, B.T., Bunke, H.: A new algorithm for error-tolerant subgraph isomorphism detection, *IEEE Trans. PAMI*, Vol. 20, No. 5, 1998, 493-507
13. Bunke, H., Shearer, K.: A graph distance metric based on the maximal common subgraph, *Pattern Recognition Letters*, Vol. 19, 1998, 255-259
14. Jiang, X., Münger, A. Bunke, H.: On median graphs: properties, algorithms, and applications, *IEEE Trans. PAMI*, Vol. 23, No. 10, 2001, 1144-1151