Near Graph Isomorphisms

D. V. Chudnovsky, G. V. Chudnovsky
IMAS
NYU Tandon School of Engineering
6 MetroTech Center
Brooklyn, NY 11201

May 24, 2017

Graph Isomorphisms, Automorphisms and Additive Number Theory

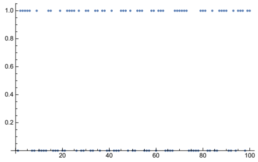
Y. O. Hamidoune. On some graphic aspects of addition theorems. In Topics in Combinatorics and Graph Theory, p.349-355. Physica Verlag, Heidelberg, 1990.

Laszlo Babai. Automorphism groups, isomorphism, reconstruction. Chapter 27 of the Handbook of Combinatorics, 1994

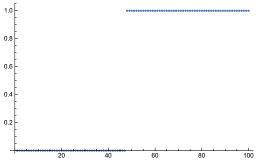
Melvyn B. Nathanson The Caccetta-Haggkvist conjecture and Additive Number Theory, 2006

Entropy 1D

If order of data does not matter, what order minimizes the "complexity" (e.g., just the entropy).
In one-dimensional case one can simply sort the data, creating a histogram (cf. with pdf and pmf for random data).



(a) Random 0/1



(b) Sorted

2D

In the two–dimensional case we have an image of dimension $W \times H$ as a matrix M of gray dots (values from $[0,1]$), or black/white dots (corresponding to $\{0,1\}$).
If one can permute rows and columns separately, what permutation will minimize the entropy (complexity) of the image?
Many definitions of entropy are possible with the simplest as a sum of Manhattan distances between nearest rows and nearest columns.

$$\sum_r \|M(r) - M(r+1)\|_{l_1} + \sum_c \|M^T(c) - M^T(c+1)\|_{l_1}$$

More complex weights (metric or non–metric) are possible (e.g. metrics minimizing both different and the same nearest columns/rows). E.g., for monochrome images:

$$SH(\vec{v_1}, \vec{v_2}) := \min(\mathrm{HammingDistance}(\vec{v_1}, \vec{v_2}), \mathrm{HammingDistance}(1 - \vec{v_1}, \vec{v_2}))$$

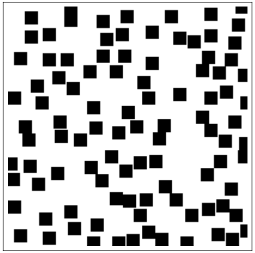
Entropy 2D

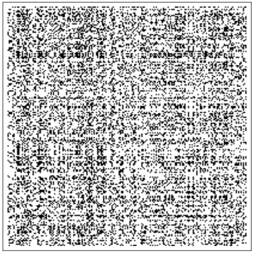
Permutations arising from such an entropy minimization try to create the largest contiguous domains of the same color from the image.
This problem is already *NP* hard, because, in particular, it contains the Traveling Salesman Problem for a general graph.

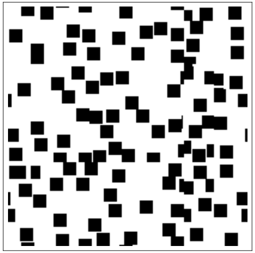
It does not mean that in practical cases of moderate size images this problem is hard to solve. In fact, an optimal or near optimal solution can be found in a reasonable time using MILP on parallel computers combined with additional local optimization.
What is interesting here is that often for images representing natural or synthetic data, the optimal permutations are not unique.

The reason for this is the large automorphism group arising from the original (scrambled) image.

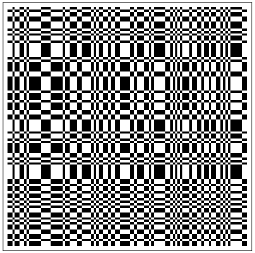
Monochrome Images

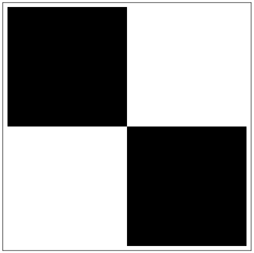

(a) Original


(b) Scrambled


(c) Minimize the Entropy

This is a low entropy image ("checker board like") – randomly permutes in columns/rows and after entropy minimization:


(a) Scrambled


(b) Entropy Minimized

Spectral Invariant

One of simplest invariants associated with graph isomorphism/automorphism problem is the spectrum of the (adjacency matrix or Laplace matrix) of the graph.
High multiplicities of eigenvalues are the obstacles to the isomorphism solutions. L. Babai (1982) showed that for graphs g and h with n vertices and multiplicities of all but one eigenvalues bounded by m, the complexity of isomorphism problem is bounded by

$$O(n^{2m})$$

Graph Isomorphisms

One should think of $W \times H$ image as a bipartite graph, whose vertices are unions of rows and columns ($W \bigcup H$) and an edge between row r and column c vertices exists only when (r, c) element of the matrix (image) is non–zero. The value at (r, c) element is the weight of the edge. For the monochrome image weights are always 0 or 1.

In this formulation permutations of rows and columns give rise to isomorphic graphs. Our discussion of graph isomorphism/automorphism uses images as adjacency matrices of bipartite graphs.

While graph isomorphism/automorphism problem has at most quasi–polynomial complexity – L.Babai, 2015-2017:

$$O(e^{\log n^{O(1)}})$$

a subgraph isomorphism problem is NP–hard. In this problem one has two graphs g and h and one has to find a subgraph of g isomorphic to graph h.

Induced Subgraph Isomorphisms

Moreover, one usually wants to find an induced subgraph of g isomorphic to h. This, induced subgraph isomorphism problem, as well as the original one, is NP complete. One can see this by taking h as a line/circle graph (Hamiltonian path/tour) or a clique.

Nevertheless, subgraph isomorphism problems are often solvable for medium/large graphs using a variety of optimization techniques such as MILP (or convex programming) together with new methods of Boolean Matrix Factorization.

Really hard problems occur when the exact isomorphism (or automorphism) does not exist, because the original graph data are incomplete or noisy. In this case we are looking at a vaguely formulated problem of nearly isomorphic graphs. This problem is actually hard to solve in practice even for moderate graph sizes.

It is exactly this problem that is important in biological and medical data analysis, where bipartite graphs represent interactions between various parts of biological systems.

Subgraph Isomorphism and Near Isomorphism

Let G and H be biadjacency matrices of bipartite graphs g and h respectively. To study induced subgraph isomorphism of h to g we look at one–to–one mapping between rows and columns of G and H. If G is a $N \times M$ matrix and H is a $K \times L$ we are looking at matrices P of size $N \times K$ and Q of size $M \times L$ such that

$$P \cdot H = G \cdot Q$$

Here P and Q are $\{0, 1\}$ matrices representing one–to–one mappings of $K \to N$ and $L \to M$. For example, when $K = N$, P is a permutation matrix (0/1 doubly stochastic matrix).

This representation fits directly into MILP formalism.
If there is no solution to the subgraph isomorphism problem one asks the question of the best near isomorphism. This is also known as an "graph edit distance problem".

MILP Formulation

This is the problem of finding P and Q representing one–to–one mappings that minimize l_1 norm of the following matrix

$$\min_{P,Q} \|P \cdot H - G \cdot Q\|_{l_1}$$

This problem is NP complete even when the dimensions of G and H are the same, i.e. when P and Q are simply permutation matrices.

Such a problem cannot be solved even approximately (within a constant factor) in a polynomial time, unless $P = NP$.

Most of the interesting problems of interpretation of realistic biological data fall into this category.

For moderate size matrices (hundreds of vertices) parallel MILP runs provide reasonably tight bounds on the near isomorphism metric.

Experiment – Subgraph Isomorphisms and Scrambled Text Image

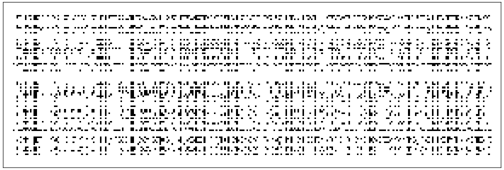


Figure: 6-Letter Text, with randomly permuted Rows/Columns

Running Subgraph Isomorphisms codes we found that there are 7 letters that are subgraphs of this image:
{"E", "H", "L", "M", "N", "V", "Y"}
Nothing in the standard dictionary, but if one thinks that this is a name, then in the Facebook list of names of people in New York area among 23, 392 names there is only a single one:

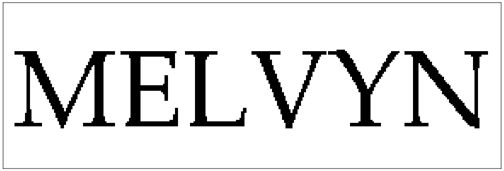


Figure: Original 6-Letter Text

Experiment – Subgraph Isomorphisms through a Sequence of Images

Look at this image:

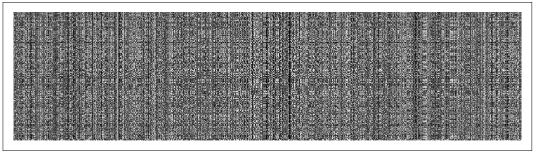


Figure: Set of Grayscale Images with randomly permuted Rows/Columns

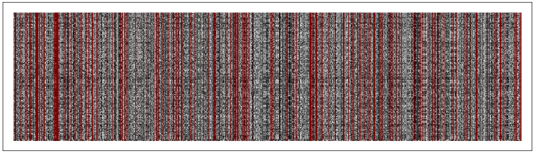
Is the picture below (as a subgraph) present in the image above:

Figure: M. N.

Relatively Simple Answer

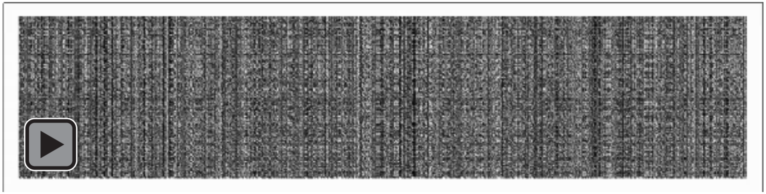
Yes.

Red Lines show the columns mapped from "M.N." picture to the bigger image



So, what was the original big image..

Unscrambling...



Cost of Subgraph Isomorphism vs "Near Isomorphism"

For the scrambled combined image the resolution of subgraph isomorphism comes rather quickly – in a matter of 100 CPU/secs per subgraph despite 100 M non-zero elements (reason: most of equations come from totally unimodular matrices).

The problem of finding the "nearest isomorphism" is much harder. For example even to get the "nearest isomorphism" (edit distance) cost between letters in the preceding "MELVYN" example takes a very long time: between 100 and 300 CPU-days (per item).

Cost. Example

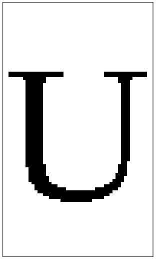
If we add 10% noise to the image (by flipping a pixel intensity w with the probability 0.1 to $1 - w$), we get this image:

Figure: 10% noise

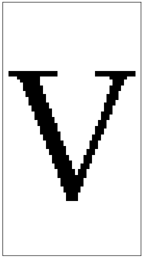
The cost of "nearest isomorphism" between the previous (scrambled) image "MELVYN" and this one is 320.

Cost. Example

The "nearest isomorphism" between these two letters is exactly 180, and to find that took 150 CPU-days:



(a) U



(b) V

But what about the Deep Learning

Deep Learning Networks, particularly Convolution Neural Networks, that are very good in image interpretation, cannot be applied to "scrambled" (isomorphic) images.

Instead that can be turned into optimization problems with additional variables (like in P, Q matrices) specifying permutations. The number of variables becomes proportional to the number of edges, making it even harder to solve in a limited time for all but relatively small graphs.

Circulant Machines – an example from Biology

Imagine "clockworks", operating independently in a large system in a repeatable fashion, with state machines specified by Circulant graphs (i.e. by graph whose weighted Adjacency matrix is a Circulant), but starting at different phases.
Then you sample all the states at one moment, can you reconstruct these "Circulant machines"..
This is how these "machines" look like neatly advancing in time and occupying their own space:

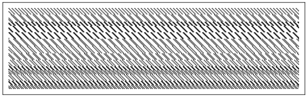


Figure: The Circulant machines with time (c)/machines (r) ordered

But if the order is lost..

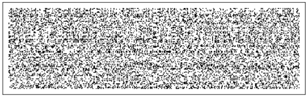


Figure: No order in c (time) nor r (machines)

Reconstruction

In the case when the data are perfect, without the noise, the key to the reconstruction is the Automorphism group of the system built from Circulant machines (and nice number-theoretic properties of Circulant plus the polynomial complexity of Automorphism problem for Circulant graphs).

Automorphisms with large cycles in *r* and *c* allow reconstruction of the time and machine elements in a canonical form (up to a smaller automorphism group):

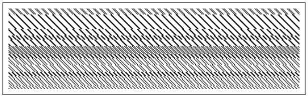


Figure: Reconstructing the order..

Unfortunately, once the noise is added at the level of 10% the reconstruction becomes a very hard problem.