ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 79-84

A Neural Graph Isomorphism Algorithm based on Local Invariants

Brijnesh J. Jain and Fritz Wysotzki

Dept. of Electrical Engineering and Computer Science, TU Berlin
Franklinstr. 28/29, D-10587 Berlin, Germany

Abstract. We present a neural network approach to solve the graph isomorphism problem. In contrast to other neural heuristics or related methods our approach is based on approximating the automorphism partition of a graph to reduce the search space followed by an energy-minimizing matching process. Experiments on random graphs with 100 - 5000 vertices are presented and discussed.

1 Introduction

Given two graphs G and H the *graph isomorphism problem* (GIP) is the problem of deciding whether G and H are structurally equivalent. The problem is of practical as well as theoretical importance. Applications include the identification of isomorphic molecular structures in chemistry, the recognition of protein molecules, the detection of kinematic chains, or optimal routing of messages in multistage interconnecting networks (see [1] and references therein). The theoretical interest in the GIP is based on the persistent difficulty in characterizing its computational complexity. The isomorphism problem is still unsolved in the sense that there is neither an NP-completeness proof, nor an efficient algorithm with polynomial complexity has yet been found.

Despite the practical and theoretical importance of the GIP no neural network approach and related heuristics can be used unmodified in a practical setting. Even the most powerful approaches by Pelillo [6] and Rangarajan et al. [8], [9], require a prohibitive amount of time and are too erroneous on random graphs with only 100 vertices, although the GIP is considered to be trivial for almost all random graphs [3]. The main reason that neural networks or related methods are not competitive with efficient graph isomorphism algorithms is that they solely rely on powerful energy minimization procedures and neglect graph-theoretical properties that are preserved under isomorphism.

In this paper we devise a two stage neural graph isomorphism (NGI) algorithm. In a preprocessing step a neural stabilization procedure partitions the vertex sets of both graphs into equivalence classes. In a second step the information about the vertex partitions is used to match the graphs with a special

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 79-84

Hopfield network. The effectiveness of the proposed NGI approach is tested on random graphs with 100 - 5000 vertices.

2 Terminology and Definitions

Let V be a set. With $[V]^2$ we denote the set of all 2-element subsets $\{i, j\} \subseteq V$. A *partition* $P = \{V_1, \ldots, V_k\}$ of a set V is a set of disjoint non-empty subsets $V_i \subseteq V$ whose union is V. Let $\mathcal{P}(V)$ be the set of all partitions of V. The elements V_i of a partition $P \in \mathcal{P}(V)$ are usually called its *cells*. If P, $P' \in \mathcal{P}(V)$, we say P is *finer* than P', if every cell of P is a subset of some cell P'. In this case we call P' *coarser* than P.

A *graph* G consists of finite sets $V(G) \neq \emptyset$ and $E(G)$, such that $E(G) \subseteq \big[V(G)\big]^2$. The elements of V and E are called *vertices* and *edges*, respectively. With $\mathcal{G}$ we denote the set of all graphs. A subset $C_m \subseteq V(G)$ consisting of m vertices is called *clique* of G if $[C_m]^2 \subseteq E(G)$. A *maximum clique* is a clique with maximum cardinality of vertices. A *maximal clique* is a clique which is not contained in any larger clique. The *clique number* $\omega(G)$ of a graph G is the number of vertices of a maximum clique in G.

Let G and H be two graphs. An *isomorphism* from G to H is a bijective mapping $\phi : V(G) \to V(H)$, $i \mapsto i^\phi$ with $\{i, j\} \in E(G) \Leftrightarrow \{i^\phi, j^\phi\} \in E(H)$ for all $i, j \in V(G)$. If there is an isomorphism between two graphs then the graphs are *isomorphic*. The *graph isomorphism problem* is the problem of deciding whether two graphs are isomorphic.

An *automorphism* of G is an isomorphism of G onto itself. Let Aut_G denote the set of all automorphisms of G. Two vertices i, $j \in V(G)$ are *similar* ($i \sim j$) if there is an automorphism $\phi \in \text{Aut}_G$ with $i^\phi = j$. The *automorphism partition* Π_G of G is the partition of $V(G)$ induced by the equivalence relation $\sim$.

Let $\mathcal{G}_\mathcal{V} = \big\{(G, i) : G \in \mathcal{G},\, i \in V(G)\big\}$. A function $f : \mathcal{G}_\mathcal{V} \to \mathbb{C}^n$ is a (*local*) *vertex invariant*, if $f(G, i) = f(H, i^\phi)$ for any isomorphism $\phi : G \to H$. The best known and most frequently used vertex invariant is the degree of a vertex. Further examples of local invariants assign a vertex i the number of vertices reachable along a path of length k, or the number of different cliques of size k.

3 A Neural Graph Isomorphism Algorithm

In practice, most algorithms adopt the same basic approach to the GIP, though the details may vary considerably. To reduce the search space this approach first approximates the automorphism partition of each graph using a vertex-classification procedure which is based on a set of selected vertex invariants. In a second step an isomorphism is constructed or non-isomorphism is established by applying a breadth-first search, depth-first search, or a mixture of both methods. The NGI algorithm as outlined in Table 1 follows a similar approach.

Before describing the essential parts (Step 1-3) of the NGI algorithm, we introduce a technical definition for convenience of presentation: All neural net-

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 79-84

Let G and H be graphs with n vertices.

1. Classify vertices of G and H by using a neural stabilization procedure. The outputs are vertex partitions of G and H which are coarser than or equal to their automorphism partitions.
2. Use the vertex partitions obtained in Step 1 to construct an association graph $G \diamond H$ of G and H. This maps the GIP to the problem of finding a maximum clique in $G \diamond H$.
3. Search for a maximum clique C_m in $G \diamond H$ by using a special Hopfield network.
4. If $m = n$ then output $G \simeq H$ otherwise $G \not\simeq H$.

Table 1: OUTLINE OF THE NGI ALGORITHM

works involved in the NGI algorithm are associated with a specific graph. Networks for approximating the automorphism partition are associated with the given graphs to test for isomorphism and the network for solving the maximum clique problem is associated with their association graph. A neural network N_G associated with a graph G consists of $\left|V(G)\right|$ fully connected units. For any vertex $i \in V(G)$ let $N(i)$ denote the set of all vertices j adjacent to i. Then the dynamics of the network is given by

$$x_i(t+1) = x_i(t) + \, w_E \cdot \sum_{j \in N(i)} o_j(t) + \, w_I \cdot \sum_{j \notin N(i)} o_j(t) \tag{1}$$

where $x_i(t)$ denotes the activity of unit i. The synaptic weight between unit i and unit j is $w_E > 0$, if the corresponding vertices are connected by an edge $\{i, j\} \in E$ and $w_I \leq 0$, if $\{i, j\} \notin E$. The output function $o_i(t)$ of unit i is a non-decreasing function applied on its activation $x_i(t)$.

Step 1 - Approximating the Automorphism Partition: For any graph G, a stabilization procedure starts with an initial partition P of $V(G)$, which is coarser than the automorphism partition Π_G of G. The partition P is iteratively refined using a set of vertex invariants. If no further refinement is possible, then the current partition is said to be stabilized. A stabilization procedure always stabilizes in a partition which is coarser than or equal to Π_G.

Let N_G be a neural network associated with G. Suppose the initial activation is identical for all units. Then N_G together with the update rule (1) is a stabilization procedure, which approximates the automorphism partition of G: At each time instance t the activation vector $\boldsymbol{x}(t)$ of N_G induces a partition $P_t(G)$ of the vertex set $V(G)$. Two vertices i and j are members of the same cell $V_k(t) \in P_G(t)$ if and only if $x_i(t) = x_j(t)$. Since the initial activation is identical for all units, the neural stabilizer starts with the coarsest partition $P_0(G)$ consisting of the single cell $V(G)$ and iteratively refines that partition according to its dynamical rule (1). The network has stabilized if the current partition $P_t(G)$ is not finer than $P_{t-1}(G)$. Theorem 3.1 and its implications summarize and prove the statements of this paragraph.

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 79-84

Theorem 3.1 *Let N_G be the neural network associated with G and $x_i(0) = \alpha$ for all $i \in V(G)$. Then for all $i, j \in V(G)$ and $t \geq 0$ we have*

$$i \sim j \quad \Rightarrow \quad x_i(t) = x_j(t).$$

Proof: (Sketch) The assertion holds for $t = 0$, since $x_i(t) = x_j(t) = \alpha$ for all $i, j \in V(G)$ irrespective of the similarity relation $\sim$. Now assume that $i \sim j \Rightarrow x_i(t) = x_j(t)$ holds for some $t > 0$. Since $i \sim j$, there exists an automorphism $\phi \in \text{Aut}_G$ with $i^\phi = j$. By construction of N_G all self-coupling weights w_{ii} are identical. Furthermore from the definition of N_G together with the edge preserving property of an automorphism follows that $w_{ik} = w_{jk^\phi}$ where $w_{ik} \in \{1, w_E, w_I\}$. Then by induction we have

$$x_i(t{+}1) = \sum_k w_{ik} o_k(t) = \sum_k w_{jk^\phi} o_{k^\phi}(t) = x_j(t{+}1)\ .$$

From Theorem 3.1 directly follows, that for all $t \geq 0$ the partitions $P_t(G)$ are coarser than or equal to the automorphism partition $\Pi(G)$ of G. Thus it is left to show, that the neural stabilizer N_G stabilizes within finite time. Since $V(G)$ is finite, there are only finitely many partitions $V(G)$ which are coarser than $\Pi(G)$. Thus any sequence of refinements is finite.

Notice that stabilization and convergence to a stable state are different concepts. The first notion corresponds to stability of a partition, the latter to stability of the state vector. For $w_E = 1$ and $w_I = 0$ we obtain Morgan's procedure [5].

Step 2 - Construction of an Association Graph: Let G and H be graphs with n vertices. We map the `GIP` to the problem of finding a maximum clique in the association graph of G and H. The association graph is a well-known auxiliary structure which goes back to Ambler et al. [2] and has since been employed with varied success not only to the graph isomorphism, but also to the more general graph matching problem [7].

Let $x_i(G)$ and $x_j(H)$ be the activation of unit i in N_G and j in N_H after stabilization. The association graph $G \diamond H$ of G and H is a graph with

$$\begin{aligned} V(G \diamond H) &= \Big\{(i,j) \in V(G) \times V(H) \;\big|\; x_i(G) = x_j(H)\Big\} \\ E(G \diamond H) &= \Big\{\{(i,k),(j,l)\} \in \big[V(G \diamond H)\big]^2 \;\big|\; \{i,j\} \in E(G) \Leftrightarrow \{k,l\} \in E(H)\Big\} \end{aligned}$$

Since G and H are isomorphic if and only if $\omega(G \diamond H) = n$, we can cast the `GIP` to the maximum clique problem in an association graph.

Step 3 - A Neural Maximum Clique Solver: For finding the maximum clique in the association graph $G \diamond H$ of two given graphs G and H we use the winner-takes-all (`WTA`) network as described in [4]. This `WTA` algorithm is extremely fast and outperforms greedy heuristics with respect to both, speed and solution quality. The performance of the `WTA` network is based on an optimal parameter setting which is theoretically justified in [4].

The topology of the `WTA` net is associated with the graph $G \diamond H$, where the connective weights w_E and w_I depend on the structure of $G \diamond H$. The `WTA`

ESANN'2003 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 23-25 April 2003, d-side publi., ISBN 2-930307-03-X, pp. 79-84

size n	edge probability p						
	0.01	0.05	0.1	0.2	0.3	0.4	0.5
100	6	3	2	3	3	3	5
500	72	49	57	55	53	55	58
1000	239	181	207	195	204	217	231
2500	1321	1140	1206	1150	1217	1360	1402
5000	5133	5080	4767	4740	5334	5594	6046

Table 2: Average computational time in *msec* taken by `NGI`.

algorithm operates as follows: An initial activation is imposed on the network. Finding a maximum clique then proceeds in accordance with (1) until the system reaches a stable state. During evolution of the network any unit is excited by all active units with which it can form a clique and inhibits all other units. After convergence the stable state corresponds to a maximal clique of $G \diamond H$. The size of a maximal clique can be read out by counting the units with activation $x_i(t) > 0$.

4 Experimental Results

We tested the `NGI` algorithm on random graphs. The algorithm is implemented in Java using `JDK` 1.2. All experiments were run on a `Sparc SUNW Ultra-4`.

For each isomorphism test we considered pairs of graphs (G, H) where G is a randomly generated graph with n vertices and edge probability p and H is a randomly permutated version of G. The chosen parameters were $n = 100, 500, 1000, 2500, 5000$ and $p = 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5$. We generated 100 examples for each pair (n, p) giving a total of 3500 isomorphism tests. Note, that the `GIP` of graphs with $p > 0.5$ is equivalent to the `GIP` of the complementary graphs. We have chosen random graph to facilitate comparison with the best methods applied to the `GIP` within the neural network community, namely the *Lagrangian Relaxation Network* (`LRN`) by Rangarajan and Mjolsness [9], the *Optimizing Network Architecture* (`ONA`) by Rangarajan, Gold, and Mjolsness [8], and the *Exponential Replicator Equations* (`REP`) by Pelillo [6].

`NGI` significantly outperforms `LRN`, `ONA`, and `REP` with respect to both, accuracy and speed. Due to their high computational effort `LRN`, `ONA`, and `REP` were tested on 100-vertex random graphs only. Accuracy of `LRN`, `ONA`, `REP` degrades for sparse graphs. The `LRN` algorithm terminated with a correct solution for all test runs except for 5% failures at $p = 0.01$. `ONA` and `REP` performed too defective on 100-vertex random graphs with $p < 0.05$. As an example for $p = 0.01$ the percentage of correct solutions is about 0.11 for `REP` and 0.0 for `ONA`. In contrast `NGI` gave exact results on all 3500 trials. But even if we are willing to live with a small degree of uncertainty, `LRN`, `ONA`, and `REP` are prohibitively slow. The average times to match two 100-vertex random graphs were about 600 - 1800 seconds for LRN on a SGI workstation, about 80 seconds for ONA on the same SGI workstation, and about 3-2000 seconds for REP on a SPARC-20 workstation. In contrast, the average time required by NGI is about 0.002 - 0.006 seconds for 100-vertex graphs and 5-6 seconds for 5000-vertex graphs. Table 2 shows the average computational time in milliseconds (*msec*) required by the NGI algorithm for an isomorphism test on random graphs with the specified parameters n and p.

5 Conclusion

We have formulated and tested a neural network approach to solve the GIP based on using local vertex invariants. Experimental results on random graphs yield exact results on all 3500 trials within impressive time limits. The results demonstrate that (1) neural networks are capable to discover structural properties in a preprocessing step, (2) the further neural processing of the discovered structural properties to solve a given problem is of greater impact than sophisticated energy minimization methods. In a forthcoming paper we intend (1) to improve NGI by incorporating additional local graph invariants, and (2) extend NGI to inexact graph isomorphisms of noisy data.

References

[1] M. Abdulrahim and M. Misra. A graph isomorphism algorithm for object recognition. *Pattern Analysis and Application*, 1(3):189–201, 1998.

[2] A.P. Ambler, H.G. Barrow, C.M. Brown, R.M. Burstall, and R. J. Popplestone. A versatile computer-controlled assembly system. In *International Joint Conference on Artificial Intelligence*, pages 298–307. Stanford University, California, 1973.

[3] L. Babai, P. Erdös, and S. Selkow. Random graph isomorphism. *SIAM Journal on Computing*, 1980.

[4] B.J. Jain and F. Wysotzki. Fast winner-takes-all networks for the maximum clique problem. In *25th German Conference on Artificial Intelligence*, pages 163–173. Springer, 2002.

[5] H.L. Morgan. The generation of a unique machine description for chemical structures. *Journal of Chemical Documentation*, 5:107–112, 1965.

[6] M. Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, 11(8):1933–1955, 1999.

[7] M. Pelillo, K. Siddiqi, and S.W. Zucker. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1105–1120, 1999.

[8] A. Rangarajan, S. Gold, and E. Mjolsness. A novel optimizing network architecture with applications. *Neural Computation*, 8(5):1041–1060, 1996.

[9] A. Rangarajan and E. Mjolsness. A Lagrangian relaxation network for graph matching. *IEEE Transactions on Neural Networks*, 7(6):1365–1381, 1996.