Edge List Homomorphisms

Tomás Feder
268 Waverley St., Palo Alto, CA 94301, USA
tomas@theory.stanford.edu
http://theory.stanford.edu/~tomas,
and
Pavol Hell
School of Computing Science
Simon Fraser University
Burnaby, B.C., Canada V5A 1S6
pavol@cs.sfu.ca

1 Introduction

2 Edge List Homomorphisms

An *edge list homomorphism* problem is defined for any fixed undirected graph H. An instance is an undirected graph G with lists of edges from H associated to the edges from G. The aim is to map the vertices of G to vertices of H so that edges in G map to edges from the corresponding edge lists. We also add vertex lists from H to the vertices in G.

A *caterpillar* is a tree consisting of a path $a_0 \ldots a_k$ for some $k \geq 0$ and additional vertices grouped into sets $B_0, \ldots, B_k$, where the vertices in B_i are adjacent to a_i for each $0 \leq i \leq k$. A *caterpillar with loop* is a caterpillar plus a single loop at the vertex a_0.

Theorem 2.1 *For any fixed graph H, the edge list homomorphism problem for H is NP-complete except for the four polynomial cases listed below:*
(1) H has at most two vertices;
(2) H is an irreflexive four-cycle;
(3) H is a caterpillar;
(4) H is a caterpillar with loop.

Proof. We first prove (1). If H has at most two vertices $0, 1$, then every vertex of an instance G can be viewed as a Boolean variable (ranging over $0, 1$), and every edge e of G with an edge list from edges of H can be viewed as a Boolean constraint involving the two variables for the endpoints of e. Such Boolean constraints can be described as conjunctions of clauses on two Boolean variables, so the problem can be solved as a 2-satisfiability instance. This proves (1).

We next prove (2). If H is the irreflexive four-cycle $00'11'$, then we may first choose which side of a bipartite instance G maps to which side of the bipartite H. The vertices in one side of G map to $0, 1$, and may thus be viewed as Boolean variables, and the vertices in the other side of G map to $0', 1'$ and may thus also be viewed as Boolean variables. Thus the problem reduces again to a 2-satisfiability instance. This proves (2).

We next prove (3) and (4) together. Let H be a caterpillar possibly with loop. Order the vertices within each B_i arbitrarily, and extend these orders to a total order of the vertices

$a_0, B_0, \alpha_1, B_1, \ldots, a_k, B_k$. Since H is bipartite (if we ignore the possible loop at a_0), we may label the two sides of the bipartition as U and V. We define a majority operation g on H by letting $g(x,y,z) = u$ if x,y,z are all three in U or all three in V and u is chosen from x,y,z to be the vertex out of these three that occurs between the other two in the ordering; if two out of x,y,z are in the same set U or V and the third vertex out of x,y,z is in the other set, V or U respectively, we let u be the earliest occurring vertex in the ordering out of the two vertices from x,y,z that are in the same set U or V. One may verify that g is a majority operation in every subgraph H' of H (not necessarily an induced subgraph of H), that is, if xx', yy', zz' are edges of H', then $g(x,y,z)g(x',y',z')$ is an edge of H'. This proves polynomiality in (3) and (4) by the algorithm in the case of majority of Feder and Vardi [7].

We now construct four particular NP-complete graphs H. The first one consists of a four-cycle $00'11'$ plus an additional edge $0'2$. We let K be the complete reflexive graph on $0,1,2$, and let K_i be K without the loop at i for $0 \leq i \leq 2$. From H we shall construct a gadget for each K_i. Superposing the three K_i gives the triangle K' without loops, and the homomorphism problem for K' is 3-colorability, proving NP-completeness. To construct K_1, we consider vertices x,z,t,u,y, with x,y,t ranging over $0,1,2$ and z,u ranging over $0',1'$. A subgraph of H can be used to restrict xz to $00'$, $20'$, $11'$; another subgraph of H can be used to restrict zt to $0'0$, $1'1$; another subgraph of H can be used to restrict tu to $00'$, $01'$, $10'$; and a last subgraph of H can be used to restrict uy to $0'0$, $0'2$, $1'1$. The condition thus defined on x and y is K_1. By symmetry between 0 and 1 we may similarly construct K_0. To construct K_2, we consider vertices x,z,t,u,y, with x,y,t ranging over $0,1,2$ and z,u ranging over $0',1'$. A subgraph of H can be used to restrict xz to $01'$, $11'$, $20'$; another subgraph of H can be used to restrict zt to $0'0, 1'1$; another sugraph of H can be used to restrict tu to $01', 10'$; and a last subgraph of H can be used to restruct uy to $0'0$, $0'1$, $0'2$, $1'0$, $1'1$. The condition thus defined on x and y is K_2. This completes the construction of the triangle K' without loops as the intersection of K_0, K_1, K_2 for NP-completeness.

Suppose H is not a tree with loops and not a four-cycle. Then H contains as a subgraph H' an irreflexive cycle that is not a four-cycle, and the problem is NP-complete, see T. Feder, P. Hell, and J. Huang [4].

In the remaining case H is a tree. Let H be the triad with edges $00', 0'3, 11', 1'3, 22', 2'3$. We construct K_0, K_1, K_2 as described above for this triad as well, that is K_i is the complete reflexive graph on $0,1,2$ minus the loop at i. Let x,z,t,u,y be vertices ranging over $0,1,2$. By concatenating four times H minus the edge $2'3$ we may let xz take values $00, 10, 22$. By concatenating four times H minus the edge $0'3$ we may let zt take values $00, 21$. By concatenating two times H minus the edge $1'3$ we may let tu take values $03, 11$. Finally by concatenating H and then three times H minus the edge $2'3$ we may let uy take values $30, 31, 32, 10, 11$. This gives for xy the graph K_2. By symmetry we can obtain K_0, K_1 similarly, and intersecting K_0, K_1, K_2 gives the triangle without loops K'. The NP-completeness for H then follows by 3-colorability.

If a tree T does not contain the triad just constructed, then T is a caterpillar plus loops. If the caterpillar T is irreflexive (has no loops) then the problem is polynomial by case (3). If the caterpillar has one loop, then it may not contain the path 30412 with a loop at 4. Let H be this graph, and construct K_0, K_1, K_2 as described above, where K_i is the complete reflexive graph without a loop at i. To construct K_0, we consider the vertices x,z,t,y. By concatenating two times H minus the edge 04 we may let xz take the values $00, 14, 24$; by using H minus the edges $30, 41, 12$, we may let zt take the values $40, 44, 04$; by concatenating two times H minus the edge 04, we may let zy take the values $41, 42, 00$. The condition thus defined on x and y is K_0. To construct K_2, we consider the vertices x,z,t,y. By using H minus the edge 30 we may let xz take the values $04, 14, 12, 21$; by using H minus the edges $30, 04, 12$ we may let zt take the values $44, 41, 14$; by using H we may let ty take the values $40, 41, 12$. The condition thus defined on x and y is K_2. To construct K_3, we use two vertices x, y. Three concatenations of H minus the edge 41 give us the condition on x and y defined by 00, 12, 21. This exchange of 1 and 2 combined with K_2 gives us K_1. This proves NP-completeness. Thus we may only add a single loop to the caterpillar to give the caterpillar with loop described above with loop at a_0 for the polynomial case (4).

If there are at least two loops added to the caterpillar, every pair of these must be adjacent, otherwise we obtain a connected subgraph whose loops are not connected, which is NP-complete by Feder, Hell, and Huang [4]. Thus we may only have two loops which must be adjacent. Finally we may not have the path 012 with loops at 0 and 1. Let H be this graph, and construct K_0, K_1, K_2 as described above, where K_i is the complete reflexive graph without the loop at i. To construct K_0, we consider vertices x, z, t, y. By using H we let xz take values $00, 11, 21,$; by using H minus the loop at 0 and the edge 12 we let zt take values $01, 10, 11$; by using H minus the edge 01 we let ty take values $00, 11, 12$. The condition thus defined on x and y is K_0. To construct K_2, we consider vertices x, z, t, y. By using H minus the loop at one we let xz take values $00, 10, 21$; by using H minus 12 and the loop at 1 we let zt take values $00, 01, 10$; by using H we let ty take values $00, 01, 10, 11, 12$. The condition thus defined on x and y is K_2. To construct K_1 we use two vertices x, y. By using H minus the edge 01 and the loop at 1 we get the condition on x and y defined by $00, 12, 21$. This exchange of 1 and 2 combined with K_2 gives us K_1. This proves NP-completeness. Thus we may only have 01 plus the two loops at 0 and 1 which is polynomial by case (1) above. □

3 Constraint List Constraint Satisfaction and Edge List Digraph Homomorphism

The *constraint list constraint satisfaction problem* is a constraint satisfaction problem where every constraint, say of arity p containing r p-tuples, may be restricted to any subset of size $1 \leq s \leq r$ of the p-tuples. We may add if desired vertex lists from the template to the vertices of the template, as vertex lists can easily be represented by constraint lists of corresponding incident constraints.

Feder, Madelaine, and Stewart [6] defined an N-free binary relation to be a binary relation B that does not contain elements $a \neq b$ and $c \neq d$ such that ac, bc, bd are in B. They showed the following.

Theorem 3.1 *If a constraint satisfaction problem H contains all possible monadic constraints, plus a collection of N-free binary constraints, then the problem is polynomial if it has a majority function and NP-complete otherwise. The NP-complete cases, without majority function, are those that have a triad $a_0 a_1 \cdots a_k$, $b_0 b_1 \cdots b_\ell$, $c_0 c_1 \cdots c_m$ with $a_k = b_\ell = c_m$ in H^3, and all vertices of this triad are triples (x, y, z) (where $(x, y, z)(x', y, z')$ is an edge of H^3 if and only if xx, yy', zz' are edges of H. Furthermore, the triples (x, y, z) in the triad must have x, y, z distinct, except for a_0 with $x = y \neq z$, for b_0 with $x = z \neq y$, and for c_0 with $y = z \neq x$.*

From this theorem we infer the following.

Theorem 3.2 *The constraint list constraint satisfaction problem can be viewed as a case of the above list problem with N-free binary relations, and is thus polynomial when the associated N-free problem has a majority function and NP-complete otherwise.*

Proof. We define a bipartite graph. On the left side we put all the vertices of the constraint list constraint satisfaction problem. On the right side we put for every p-ary relation B of size r a vertex for each of the r p-tuples. For each of such relations B, we define p binary relations B_i,

$1 \leq i \leq p$, and include xy in B_i for x on the left and y on the right if y corresponds to a p-tuple t in B whose entry t_i satisfies $t_i = x$. An instance of this bipartite problem with lists corresponds to an instance of the original problem because if a vertex v must map to the right side, its list must be contained in the vertices y for a single relation B, as the B_i where v occurs determine the corresponding B. Lists allow us to restrict the constraints to subsets of such B. Since the B_i are N-free, the result follows. $\square$

The *edge list digraph homomorphism problem* is the special case of the preceeding theorem where the constraint satisfaction problem has a single relation, and this relation is binary. It follows that this problem is either polynomial with a corresponding majoriy function from the N-free problem or NP-complete. The case where the digraph is symmetric leads to the edge list graph homomorphism problem studied in the previous section only if we additionally require that if a list contains the edge ab, then it also contains the edge ba. It remains open to consider such symmetry conditions for constraint list constraint satisfaction problems with more than one binary symmetric constraint or a single symmetric constraint of higher arity.

4 Line graph list partition

In the edge list graph homomorphism problem edges had lists, and the homomorphism constraint was on the incidence of vertices and edges. We now consider a problem where the constraints are among edges, depending on whether two edges share an endpoint or not. The *M-partition problem* is given by a symmetric matrix M with $0, 1, *$ entries. The problem is to partition an input graph G into parts corresponding to rows (and corresponding columns) of M, so that two vertices going to parts i, j can only be adjacent if the ij entry of M is $*$ or 1 and can only be nonadjacent if the ij entry of M is $*$ or 0. In the *list M-partition problem* vertices are in addition equipped with lists of allowed parts. To make this a problem on edges, we restrict our attention to line graphs. The *line graph* of a graph G is a graph G' whose vertices are the edges of G, and two vertices in G' are adjacent if and only if the corresponding edges of G share an endpoint.

Our main result of this section is the following.

Theorem 4.1 *The list M-partition problem on matrices M without $*$ on the diagonal for line graphs is polynomial unless M contains three parts i, j, k with 0 on the diagonal and $*$ off diagonal ij, jk, ik, in which case it is NP-complete by edge 3-coloring [9].*

We distinguish first the cases where the diagonal is all 0 or all 1.

Lemma 4.2 *The case of Theorem 4.1 with 0 on the diagonal holds.*

Proof. Suppose parts i, j have entry ij of M equal to 1. Then as a line graph contains no $K_{2,3}$, one of i, j will be assigned at most two edges of the line graph. We select these two edges for say i in all possible ways and simplify the matrix by removing part i, after adjusting the lists of all other edges to take into account these two assigned edges, which are removed from the graph.

We may thus assume that M has no 1s. If G has a vertex of degree at least 3, then the three incident edges must go to three 0 parts joined by $*$s off diagonals. This is the NP-complete case mentioned in Theorem 4.1. We may thus assume that all vertices of G have degree at most 2, so G decomposes into a union of paths and cycles.

We may solve the problem for these paths and cycles in polynomial time by considering each such path or cycle in turn and traversing it, keeping track by dynamic programming of the first assigned edge in the case of the cycle, and keeping track of all parts used so far. $\square$

Lemma 4.3 *The case of Theorem 4.1 with* 1 *on the diagonal holds.*

Proof. The cliques corresponding to the 1 parts are either triangles or edges incident to a single vertex v. Consider all possible ways of choosing triangles for parts, and of choosing vertices for parts. It remains to determine in the case where the edges incident to a vertex v go to a set I of parts i which part i from I will be chosen for each edge. This can be done by considering the vertex w at the other end, which was not chosen as a part. For w, select among candidate parts for edges joining to various v with sets I consistently for M the various i out of the various I. □

We now prove Theorem 4.1

Proof. By Feder, Hell, Klein, and Motwani [5], the number of possible partitions of which edges of G will go to the k parts with the 0 diagonal and which edges of G will go to the ℓ parts with the 1 diagonal, in about $n^{k\ell}$ possible ways. Once this is done, find all possible feasible solutions for the 1 parts as in Lemma 4.3, leaving open when there are multiple parts i from I for edges incident to v which combination will be chosen at the other end w. Then solve the 0 parts as in Lemma 4.2 by dynamic programming, making the decisions for the 1 parts as vertices are encountered in the dynamic programming, and keeping track of which sets of parts have been assigned to the edges going to the 1 parts to verify the part of M joining the 0 parts and the 1 parts as well. □

The *restriction M-partition* problem is the list M-partition problem where all vertices are restricted to have the same list.

Corollary 4.4 *The restriction M-partition problem on matrices M (possibly with $*$ on the diagonal) for line graphs is polynomial unless M contains three parts i, j, k with* 0 *on the diagonal and $*$ off diagonal ij, jk, ik, in which case it is NP-complete by edge 3-coloring [9].*

Proof. If the chosen list has an entry that has a $*$ on the diagonal, then the answer is trivially yes, so we may assume that the chosen list has entries with no $*$ on the diagonal. The polynomiality then follows from Theorem 4.1 theorem in the polynomial cases, and the NP-completeness from the NP-completeness of edge 3-coloring. □

Now we consider M-partition without lists. The polynomial cases of Corollary 4.4 remain polynomial. For the NP-complete cases, we may restrict an instance G' that is the line graph of G to map to parts that contain 0 on the diagonal and only $0, *$ off the diagonal by making multiple disjoint copies of G to obtain F and let F' be the instance line graph of F. Such a problem without 1s in the matrix can be viewed as the H-coloring problem, where H is the graph that has edges joining parts connected by $*$.

Lemma 4.5 *Consider the H-coloring problem on line graphs. If H has no triangle then the problem is polynomial as before. If H is a clique of size at least 3 the problem is NP-complete [8].*

However, not all cases with triangles are NP-complete. For $k \geq 3$, let the *k-wheel* W_k be the graph containing an induced k-cycle C_k plus one additional vertex v adjacent to all vertices of C_k.

Theorem 4.6 *The H-coloring problem on line graphs is polynomial time solvable for $H = W_5$.*

Proof. Suppose we want to H-color the line graph G' of G. If G has a vertex of degree 4 or more the answer is no since H contains no K_4. If G has vertices of degree 1 or 2 we may connect two copies of G at identical vertices of degree 1 or 2 to make all vertices degree 2 or 3, and do this again to make all vertices degree 3. We may thus assume G is 3-regular. H-coloring G is equivalent to finding a perfect matching in G (to go to the extra vertex v) so that removing the matching the resulting union of cycles has no triangles (and can thus go to the 5-cycle C). If two triangles in G share an edge e, then any matching in G will necessarily cover both triangles. If a triangle T is isolated, then T is covered precisely when exactly one of the three edges coming into T is chosen by the matching (disallowing choosing all three edges which would leave T uncovered). So we may shrink T to a single vertex to be matched once. After shrinking all such isolated T, solve the graph matching problem on the resulting cubic graph. □

Theorem 4.7 *The H-coloring problem on line graphs is NP-complete for $H = W_k$, $k \neq 5$, $k \geq 3$.*

Proof. The case $k = 3$ is 4-edge-colorability and is NP-complete. The case $k \geq 4$, k even has a core that is a triangle, and is thus 3-edge-colorability which is NP-complete.

The remaining case is $k \geq 7$, k odd. We consider instances G' that are line graphs of cubig graphs G. The problem is to find in G a perfect matching M to be assigned to the special vertex v, so that the union of cycles $G - M$ contains no odd cycle of length strictly smaller than k. In other words, we are looking for a perfect matching M that hits all odd cycles of length at most $k-2$. In particular, M must hit all 5-cycles.

If we consider a 5-cycle $a_1a_2a_3a_4a_5$ and the five edges e_i coming out of a_i respectively, the matching M must contain either exactly one of the five e_i, or exactly three consecutive of the five e_i. This can be viewed as a Boolean constraint $C = \{10000, 01000, 00100, 00010, 00001, 11100, 01110, 00111, 10011, 11001\}$, where a 1 in the ith position, $1 \leq i \leq 5$, represents the case where e_i is chosen for M. The Boolean constraint satisfaction problem with two occurrences per variable with constraint C is NP-complete (see Feder [1]), as C is neither 2-satisfiability, nor Horn, nor dual-Horn, nor linear, nor delta-matroid. To represent this Boolean problem, we join the two occurrences of five Boolean variables by attaching both sides of an e_i to 5-cycles.

Next we ensure that there are no other odd cycles of length strictly less than k as follows. Define a gadget eGf for two edges $e = (x, x_0), f = (y, x_1)$ by introducing two additional vertices x_2, x_3 and edges (x_2, x_3), $(x_0, x_2), (x_0, x_3), (x_1, x_2), (x_1, x_3)$. Clearly e is in M if and only if f is in M. We now replace each edge $f_i = (x, y)$ joining two 5-cycles with a path $P = e_0Ge_1, e_1Ge_2, \ldots, e_{k-1}Ge_k$ with the endpoints x, y attached at e_0, e_k respectively. This guarantees that with the exception of the 5-cycles at either end of such paths and the two triangles sharing an edge in each eGf there are no odd cycles of length strictly less than k.

Finally we must be able to set an f_i joining two 5-cycles to either be in M or not be in M. To set to be in M, create the path P as above and join the ends of e_0, e_k to form a cycle, with an additional edge f joined at the common end of e_0, e_k. As e_0, e_k must simultaneously be or not be in M, they must not be in M, so f is forced to be in M. Using this gadget twice instead of the first edge in the path for f_i guarantees f_i is chosen for M. To set f_i not to be in M, join f required to be in M as just defined to two other edges g_1, g_2 that will not be in M. Using this gadget with g_1, g_2 instead of the first edge in the path for f_i guarantees f_i is not chosen for M. This completes the reduction from the Boolean problem. □

Corollary 4.8 *Let H be a graph with at least one triangle T and containing a vertex v such that every triangle T of H contains v. If the neighborhood of v does not contain a 5-cycle, then H-coloring for line graphs is NP-complete. If the neighborhood of v contains a 5-cycle, then H-coloring for line graphs is polynomial time solvable.*

Proof. Suppose the neighbors of v does not contain a 5-cycle. Then either this neighborhood is bipartite or has the shortest odd cycle of odd length $k \geq 7$. Then an instance G' that is the line graph of a cubic graph G must map to v and its neighbors, and here the core is a triangle in the bipartite case, giving NP-completeness by [9]. In the case where the shortest odd cycle has length $k \geq 7$, the edges that do not map to v form cycles which must be of even length or of odd length at least k, that is the problem is NP-complete by Theorem 4.7.

Suppose the neighbors of v contain a 5-cycle. If we consider an instance that is the line graph G' of G, then after removing from G the edges that map to H we are left with paths and cycles, since $H - v$ has no triangles. The mapping is complete if these paths and cycles contain no triangle by mapping to v and the neighboring 5-cycle, otherwise if some cycle is a triangle the mapping is not possible. Thus polynomiality follows from Theorem 4.6. □

Corollary 4.9 *Let H be a graph with at least one triangle T and containing an independent set I such that $H - I$ contains no triangles. Let E be the set of edges in $H - I$ that belong to a triangle with a vertex in I. If $H - I$ does not contain a 5-cycle, and either E is bipartite or H contains some k-wheel W_k with $k \geq 7$ odd, then H-coloring for line graphs is NP-complete. If H contains the 5-wheel W_5, then H-coloring for line graphs is polynomial time solvable.*

Proof. As in the preceeding corollary, the edges of G that go to I form a perfect matching M so that the remaining edges contain no short odd cycle, and NP-completeness follows as in the proof of Theorem 4.7. If H contains the 5-wheel, then it suffices that the complement of the matching M contain no triangle to obtain a polynomial algorithm as in Theorem 4.6. □

References

[1] , T. Feder, Fanout limitations on constraint systems. Theoretical Computer Science 255 (2001) 281-293.

[2] T. Feder and P. Hell, List homomorphisms to reflexive graphs. J. Comb. Theory Series B 72 (1998) 236-250.

[3] T. Feder, P. Hell, and J. Huang, List homomorphisms and circular arc graphs. Combinatorica 19 (1999) 487-505.

[4] T. Feder, P. Hell, and J. Huang, Bi-arc graphs and the complexity of list homomorphisms. J. Graph Theory 42 (1999) 61-80.

[5] T. Feder, P. Hell, S. Klein, and R. Motwani. List partitions. SIAM J. Discrete Mathematics 16 (2003) 449-478.

[6] T. Feder, F. Madelaine, and I.A. Stewart Dichotomies for classes of homomorphism problems involving unary functions. Theoretical Computer Science 314 (2004), 1-43.

[7] T. Feder and M.Y. Vardi, The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. SIAM J. Comput. 28 (1998) 236-250.

[8] Z. Galil and D. Leven, NP-completeness of finding the chromatic index of regular graphs. Journal of Algorithms 4 (1983) 35–44.

[9] I. Holyer, The NP-completeness of edge-coloring, SIAM J. Comput. 10 (1981), 718-720.