Embedding trees in Cartesian products by an edge.

Tomás Feder[*] Carlos Subi[†]

February 19, 2009

1 Introduction

In this paper we consider the problem of embedding trees in Cartesian products by an edge. We define three equivalent versions of the problem Let $G = (U, V, E)$ be a bipartite graph with $|U| = k$, $|V| = \ell$, $k \geq \ell$. The *bicoloring problem* is the problem of assigning to the vertices of G k colors A and ℓ colors B such that the edges resulting edges AB form a matching. The *factoring by an edge problem* is the problem of describing G as a subgraph of the Cartesian product of an edge and a clique with k vertices, in such a way that one side of the edge has k vertices and the other side of the edge has ℓ vertices. The *bipartite factoring by an edge problem* is the problem of describing G as a subgraph of the Cartesian product of an edge and a bipartite graph with k vertices, in such a way that one side of the edge has k vertices and the other side of the edge has ℓ vertices.

Theorem 1 *The bicoloring problem, the factoring by an edge problem, and the bipartite factoring by an edge problem are equivalent problems for all graphs G.*

PROOF: Suppose we can solve the bicoloring problem. Then we can put the vertices labeled A in one side and the vertices labeled B in the other side in such a way that the matched vertices AB coincide with the edge of the factorization, thus solving the factoring by an edge problem. Furthermore, if k vertices go to the A side and ℓ vertices go to the B side, then we have some $k - r$ vertices of U going to A and r vertices of U going to B, while r vertices of V are going to A and $\ell - r$ vertices of V are going to A. Matching the two r size parts and the $k - r, \ell - r$ size parts respectively, we may observe that the bipartite graph that we multiply by an edge can be taken to be a complete bipartite graph $K_{k-r,r}$ or some subgraph of it. This solves the bipartite factoring by an edge problem. Finally, given a solution to the bipartite factoring by an edge problem, we get the A side and the B side solving the bicoloring problem. Thus all three problem are equivalent. ∎

[*]268 Waverley St., Palo Alto, CA 94301. E-mail: tomas@theory.stanford.edu.
[†]E-mail: carlos subi@hotmail.com.

2 NP-Completeness in the Balanced Case and Easiness for Trees

The *balanced* version of the three equivalent problems is the case where $k = \ell$.

Theorem 2 *The balanced version of the three equivalent problems is NP-complete even if the bipartite graph is connected and all degrees are at most 5.*

PROOF: The reduction is from a Boolean constraint satisfaction problem where Boolean variables x_i are restricted by constraints that can set them to 0 or 1, or forbid the two patterns $(0, 1, 0), (1, 0, 1)$ for three distinct variables. This problem is NP-complete from Schaefer's classification, since it not Horn, dual Horn, 2SAT, or linear.

The values 0 and 1 are represented by the two sides A and B. The two constants A and B are represented by two large N by N bipartite cliques $K_{N,N}$ that will have to go to the two respective sides A and B so that the two sides end up with the same number of vertices. All variables are represented by smaller n by n bipartite cliques $K_{n,n}$, with a path (x, y, z) forbidding (A, B, A) and (B, A, B), where x, y, z belong to different cliques $K_{n,n}$. If a solution exists, we guarantee that the two sides have the same size k in the solution by including a second copy of the instance connected to the $K_{N,N}$ for B, A instead of A, B which can be solved symmetrically. Notice that each vertex of $K_{N,N}$ or $K_{n,n}$ need only be used once to guarantee that the paths (x, y, z) do not interfere. Finally, the degree is reduced to 5 by using instead of $K_{N,N}$ and $K_{n,n}$ a set of smaller bipartite cliques $K_{2,3}$ and $K_{3,2}$ so that combining them both creates degree 5 and leaves vertices of degrees 2 and 3 which can be used for the paths (x, y, z). ■

We now proceed to examine what happens if the graph G is a tree.

Theorem 3 *If G is a tree with vertices of degrees at most 3, the answer the all three equivalent versions of the general problem is YES.*

PROOF: Suppose G is a tree T. If T has just one vertex, it can be labeled A or B correctly, while if T is two adjacent vertices, then we may match them with the edge joining A and B, as one vertex is in side 1 and one vertex in side 2.

Suppose T has a leaf x with a parent y of degree 2. Then we may match x to y and remove both x and y, proceeding by induction. As x and y will go to different sides 1 or 2, it follows that they will be correctly labeled A and B or viceversa.

Otherwise G contains a tree T_1 with a root x and two leaf children y and z, with x having parent t in T. If these four vertices are all of T, then T has three vertices in side 1 and one vertex in side 2, so it suffices to match t and x to have three vertices in A and one in B. Otherwise t may or may not have other children. If t has no other children, then if the parent of t is A we match x and y to get three A and one B, and if the parent of t is B we match t and x to get three A and one B. If t has another child x' which is a leaf, then if the parent of t is B, we may match t and x to get three A and two B, and if the parent of t is A, we may match t and x' and also match x and y to get three A and two B.

In the remaining case, there are two copies T_1 and T_1' with common parent t, and t has parent u. If u has no other children, then either the parent of u is A, in which case we match

u to t, x to y, and x' to y' to get three A and five B, or the parent of u is B, in which caes we match t to x to get three A and five B. If u has onether child t' with two children x'', x''' that are leaves, then either the parent of u is A, in which case we match u to t and also match x to y to get five A and six B, or the parent of u is B, in which case we match u to t, match x to y, and x' to y', to get five A and six B. Finally if x'' has children y'', z'' that are leaves and x''' has children y''', z''' that are leaves, then either the parent of u is A, in which case we match u to t' and match t to x to get five A and ten B, or the parent of u is B, in which case we match u to t, match x to y, and match x' to y' to get five A and ten B. ■

For large enough degrees, the answer can be NO.

Proposition 1 *There exists a balanced tree with 10 vertices and maximum degree 5 such that the answer to the problem is NO.*

PROOF: The tree consists of a path (x, y, z), with x, z each attached to 2 leaves and y attached to 3 leaves. It is easy to check that no matching will divide this tree into five A and five B. ■

However, we obtain an algorithm for general trees.

Theorem 4 *The general problem for trees T can be solved in polynomial time.*

PROOF: Root the tree T. For each subtree separated from the root by removing an edge, determine all possible numbers of A, B decompositions that it can have, in at most n^2 ways, keeping track for each of them whether the root of the subtree whether the root is A or B and whether it need be matched in the subtree. Given a subtree, separate from its root its d children, and consider assigning say A to the root and at most one child B that is not matched in the smaller subtree, to be matched with the root. We can examine the d subtrees in some order, and combine successively the n^2 possible choices for each of them to obtain the n^2 possible combined choices, in polynomial time. ■

3 Embedding Balanced Trees in Products of an Edge by a Tree is Polynomial

Theorem 5 *Let T be a balanced tree with k vertices in each side. Whether T can be embedded in the product of an edge by a tree T' of size k can be solved in polynomial time.*

PROOF: Suppose x and x' are embedded across the edge in the product but not necessarily adjacent in T. Then the path of length $2r-1$ from x to x' in T goes from x to some y of length $r-1$ along a path p in T', then across the edge in the product to some y', then along the path p backwards in T'. If two such paths joining two pairs like x, x' and x'', x''' intersect, then they form two isomorphic subtrees joined across the edge yy'. Thus in general, we will have combining all such paths a decomposition of the tree into components that consist of two isomorphic subtrees joined at the root.

We can inductively assume for subtrees separated from the root by an edge that such a decomposition has been found, and for the last component to be found, choose the edge joining T and T', match at the children of the roots the subtrees in T and T' that can correspond to each other, and then by a bipartite matching algorithm determine whether the roots can be matched to each other as well. ■

Proposition 2 *There exists a balanced tree with 22 vertices and degrees 1 and 3 for which the algorithm gives answer NO.*

PROOF: Let T and T' be two complete binary trees of height 2 with 7 vertices, connected at the roots. Attach to two leafs of the same subtree of size 3 in T two leaf children each, and attach to two leafs of different subtrees of size 3 in T' two leaf children each. This is a balanced three with 11 vertices in each side and vertices of degrees 1 and 3. No edge separates this tree into two balanced trees or two isomorphic trees, so the algorithm above will give answer NO. ■

Corollary 1 *Consider for some constant k a forest F with at most k components. The problem of embedding F into a product of an edge by a tree in a balanced manner can be solved in polynomial time.*

PROOF: The forest F can be turned into a tree by adding $k-1$ edges, in $n^{O(k)}$ possible ways. As k is constant, this is polynomial. ■

Corollary 2 *Consider for some constant k a tree T and the problem of embedding T into a product of an edge by a graph that is a tree T' plus k edges while staying bipartite. This problem can be solved in polynomial time.*

PROOF: It suffices to remove k edges from T to need just the tree T' and which can be obtained by the algorithm for a forest with $k+1$ components, and then add the k removed edges, provided that the added edges join vertices in the same side A or B, which can be guaranteed alongside with the algorithm. ■