AKCE J. Graphs. Combin., 5, No. 2 (2008), pp. 117-125

On the Wimer Method for Designing Edge-Based Algorithms

Alan C. Jamieson
Department of Mathematics and Computer Science
St. Mary's College of Maryland
St. Mary's City, MD 20686
e-mail: *acjamieson@smcm.edu*
Wayne Goddard and Stephen T. Hedetniemi
School of Computing
Clemson University
Clemson, SC 29634
e-mail: *goddard@cs.clemson.edu, hedet@cs.clemson.edu*

Communicated by: T.W. Haynes
Received 1 January 2008 ; accepted 24 July 2008

Abstract

The construction of "Wimer"-style table-based algorithms for edge parameters in graphs is discussed. While algorithms for many parameters on graphs classes such as bounded treewidth are commonplace, there are some potential advantages to the table approach, such as automated construction and information gathering.

Keywords: algorithm, table, tree, automated bounds

2000 Mathematics Subject Classification: 05C85

1. Introduction

Two decades ago, Wimer at al. [12] and others proposed a table-based method for constructing dynamic programming algorithms for vertex parameters on trees. As they, and Bern, Lawler, and Wong [2], and others pointed out, this approach can be attempted for many recursive structures. Nowadays, algorithms for parameters on classes such as bounded treewidth are commonplace. However, explicit construction of table-style algorithms for edge parameters seems not to have been considered. In this note, we show how to generate such tables, and discuss what other information might be obtained from them.

To explain the method, we use a simple example parameter α_1^+, the upper edge covering number (definitions later). This parameter is known to equal $n - \gamma$ for all graphs, where n is the order and γ the domination number of the graph. Thus an algorithm for α_1^+

is known for any class of graphs with a domination number algorithm. But the example parameter shows the simple table approach at work in trees. We also discuss how the table might help one discover and prove the lower bound of $\lceil n/2 \rceil$ for this parameter in trees, and how its table and the table for domination might help one discover and prove the connection between it and domination.

For another example, we consider the parameter Γ'_t, the upper total edge domination number. We discuss how the table for Γ'_t can be generated with computer assistance, both by explicitly calculating how the table must look based on all small trees, and by exploiting the original Bern, Lawler, & Wong construction [2] that produces the table for the minimal version of a parameter from the table for the original.

2. The Wimer Edge Variant for Trees

The original Wimer method [13, 12] looked at marked sets of vertices and how they behave under composition. The *edge variant* allows one to compute optimum sets of edges rather than sets of vertices. It is a simple adaptation that allows two kinds of compositions of two rooted trees, one of which takes the edge between the two roots, while the other does not.

Let $T = (V, E)$ be a rooted tree with the root denoted by r. We consider rooted edge-marked trees (T, S, r), hereafter called *rem* trees, some of whose edges S are marked. There are two composition operations:

1. *Not marking the edge*: Given two *rem* trees (T_1, S_1, r_1) and (T_2, S_2, r_2), we form the *rem* tree $(T_1 \diamond T_2, S_1 \cup S_2, r_1)$ by adding an unmarked edge between r_1 and r_2. We denote this operation by $T_1 \diamond T_2$.

2. *Marking the edge*: Given two *rem* trees (T_1, S_1, r_1) and (T_2, S_2, r_2), we form the *rem* tree $(T_1 \blacklozenge T_2, S_1 \cup S_2 \cup \{r_1 r_2\}, r_1)$ by adding a marked edge between r_1 and r_2. We denote this operation by $T_1 \blacklozenge T_2$.

A *Wimer edge congruence* $\equiv$ for a given edge-set property P is a partition of the set of all *rem* trees into a finite number of classes such that:

1. The equivalence relation is invariant under the two composition operations. That is, if *rem* trees (T_1, S_1, r_1) and (T_2, S_2, r_2) are equivalent, and (T, S, r) is any *rem* tree, then $T_1 \diamond T \equiv T_2 \diamond T$, $T \diamond T_1 \equiv T \diamond T_2$, $T_1 \blacklozenge T \equiv T_2 \blacklozenge T$, and $T \blacklozenge T_1 \equiv T \blacklozenge T_2$.

2. The set of all *rem* trees whose set of marked edges has the given property P is the union of some of the equivalence classes.

Given a Wimer edge congruence with k classes, one can construct a $k \times k$ table whose rows and columns are labeled with the equivalence classes, numbered $1, 2, \ldots, k$, and whose (i, j)-entry is the pair of congruence classes $i \diamond j : i \blacklozenge j$.

It is important to note that this variant is largely inspired by the paper of Bern, Lawler, and Wong [2] where they discuss subgraph computations and building algorithms for recursive structures. Further, while there are several methodologies for developing linear-time algorithms for trees and other families of recursive structures e.g. [3], and several specific algorithms e.g. [1, 10, 14], the table method may provide a simpler algorithm, and, as we show, has other potential benefits.

3. Example 1: Upper Edge Covering Number

As a simple example, we present an algorithm to find the upper edge covering number of a tree. An *edge cover* is a set S of edges such that every vertex in V is incident with an edge in S [11]. Given an edge cover S, if a vertex v is only incident to one edge, say uv, in S, then we say that vertex v is a *private vertex of* uv. An edge cover S is a *minimal edge cover* if every edge $uv \in S$ has at least one private vertex. The *upper edge covering number* $\alpha_1^+(G)$ of a graph G is the maximum cardinality of a minimal edge cover of G. It is known (see for example [9]) that for all graphs that $\alpha_1^+(G) = n - \gamma(G)$, where n is the order and $\gamma(G)$ the domination number of the graph.

3.1. The Table for a Tree

To determine the Wimer table for the upper edge covering number of a tree T, one has to determine the equivalence classes that can be created by a minimal edge cover restricted to a rooted subtree. These classes are:

1. All *rem* trees in which the root r is not incident to a marked edge, but the set of marked edges forms a minimal edge cover of $T - r$;
2. All *rem* trees in which the root r is incident to a marked edge, every such edge has a private vertex that is not r, and the set of marked edges forms a minimal edge cover of T; and
3. The class of all *rem* trees in which the root r is incident to a marked edge rv where v is not a private vertex of rv, and the set of marked edges forms a minimal edge cover of T.

To complete the congruence, there is a class 0 for all other *rem* trees. An example of each possible class is given in Figure 1. The marked edges are given by notches on the edges.

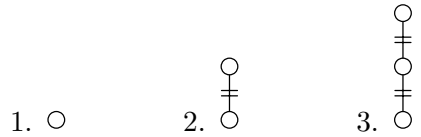

Figure 1: Example trees from each class for Minimal Edge Cover

Next one considers the construction of the tree T from its subtrees. The Wimer table for minimal edge cover is given in Table 1. If a particular composition of two subtrees would create a tree in class 0—meaning it would never be possible to choose marked edges outside the subtree to be a minimal edge cover of the whole tree—the appropriate entry is marked with a hyphen.

	1	2	3
1	-:2	1:3	1:-
2	-:2	2:-	2:-
3	-:-	3:-	3:-

Table 1: Table for Minimal Edge Covers

Now, the algorithm for α_1^+ is obtained by converting the table into a set of recurrence equations, just as in the original method [12]. If an unmarked tree T_a is formed by the composition of unmarked trees T_b and T_c, then

$$[1]_a = \max\{\,[1]_b + [2]_c,\ [1]_b + [3]_c\,\}$$
$$[2]_a = \max\{\,[1]_b + [1]_c + 1,\ [2]_b + [1]_c + 1,\ [2]_b + [2]_c,\ [2]_b + [3]_c\,\}$$
$$[3]_a = \max\{\,[1]_b + [2]_c + 1,\ [3]_b + [2]_c,\ [3]_b + [3]_c\,\}$$

where $[i]_j$ denotes the maximum cardinality of a set of marked edges in a marking of T_j in class i. The algorithm calculates the vector $[[1],[2],[3]]$ using essentially a postorder traversal of the tree. As the singleton tree is in class 1, the starting vector for it is $[0,-\infty,-\infty]$. The set of minimal edge covers is the union of classes 2 and 3; thus, α_1^+ is the maximum of $[2]$ and $[3]$ at the root.

3.2. Verifying the Table

Of course, one should provide some proof for the table. One approach is to justify each entry and thereby also establish that no class needs to be refined. We will not discuss all the entries in Table 1; rather we will discuss the first two entries in column 2 to give the flavor.

Consider the composition of a subtree of class 1 and a subtree of class 2. If one does not mark the new edge, the root is still not incident to any marked edge, which means the tree is in class 1. If one marks the new edge, the new edge has only the root as a private vertex and the resulting tree is in class 3.

Consider the composition of two subtrees of class 2. If one does not mark the new edge, the tree remains in class 2. If one marks the new edge, there is a path of three consecutive marked edges. This cannot form a minimal edge cover because one would still have an edge cover if the new edge were removed; thus this case is invalid.

In a large table it is not easy to verify that all the classes have been found and that the table entries are as given. The latter verification can be automated, *provided* the number

of classes is known to be correct and one knows a representative tree from each class. See [6] for how to do this with the original Wimer tables; the extension to edge tables is straight-forward.

3.3. Using the Table

The information in a table can help one to discover and prove theorems. We consider two examples: (a) the lower bound of $\lceil n/2 \rceil$ for α_1^+, and (b) its Gallai-type relationship with domination.

For (a), one can use the techniques of [5]. Define for a rooted unmarked tree T the vector $\mathcal{E}(T) = [[1], [2], [3]]$ calculated above. That is, it is the vector giving the maximum number of marked edges in each class. Then, for $n \geq 1$, define the set $\mathcal{E}(n)$ as the set of all vectors of trees of order n. As a first approach, one can simply calculate this set for increasing n, and for each n determine the minimum value of α_1^+. Note that the set has $O(n^3)$ vectors, and can be calculated recursively from $\mathcal{E}(i)$ for $i < n$; so this process is polynomial-time and one can readily determine $\mathcal{E}(n)$ for n up to some reasonable value. In this case, the pattern of $\lceil n/2 \rceil$ jumps out. One can speed up the calculation by retaining only the "minimal" vectors for each n, since we are only interested in the minimum value of α_1^+ for each n. Further, as explained in [5], one can use pattern recognition techniques to automate the discovery of the pattern, and to automate production of an actual inductive proof of the result.

For (b), one can conceptually use the same approach. The Wimer table for γ has three classes [12]. Thus one can construct a 6-tuple for a rooted tree T that is the concatenation of the vectors for α_1^+ and γ. Again one can generate the set $\mathcal{E}(n)$ for successive n, and calculate that α_1^+ and γ always sum to n. In theory, the automated techniques generalize to this case, though they appear to need more complex programming than the simplistic assumptions used in [5].

4. Example 2: Minimal Total Edge Domination

The idea of *total edge domination* was introduced in [10]. A set S of edges is a *total edge dominating* set if every edge in E is adjacent to an edge in S. The *upper total edge domination* number $\Gamma_t'(G)$ is the maximum cardinality of a minimal total edge dominating set of a graph G. In this section, we utilize the Wimer edge variant to present the first algorithm for finding the upper total edge domination number in a tree.

4.1. Finding the Table

The table for minimal total edge dominating sets in a tree has 10 classes. One approach to finding the table is to try to determine the partition and Wimer edge congruence by hand. This is fraught with difficulties, and it is easy to make mistakes (as previous

versions of this manuscript attest). Another approach, at least in theory, is to proceed via an expression in logic, as given for example in [4]. It is unclear how difficult it is to actually follow that path in practice; but we consider here an alternative approach using a computer.

One way to proceed is to consider how the table must look to handle all small trees. Initially one has a partition of the set of all *rem* trees into two classes: those whose marked sets are valid minimal total edge dominating sets and those that are not. Then repeatedly one takes all small trees, calculates their class and the apparent class of their composition. If one obtains inconsistent answers for the composition of class i and class j trees, then that shows that at least one of the classes needs to be refined. And then one can systematically iterate this approach, or use recursion. This process was described in [5]. A drawback is that the computer still does not prove that the partition does not need to be refined, only that the smallest tree in any new class is large.

For the case of Γ'_t, we actually chose an easier approach. This approach is to first create the table for plain total edge domination. This table can be generated by hand, or by computer as we did, and it is relatively easy to verify. It is given in Table 2.

	1	2	3	4
1	2:3	-:3	-:4	1:4
2	2:3	-:3	-:4	2:4
3	3:4	-:4	-:4	3:4
4	4:4	-:4	-:4	4:4

Table 2: Table for Total Edge Dominating Sets

After that, we apply the original Bern, Lawler, and Wong [2] idea of converting the normal table into the table for the minimal version. (They did it for the vertex parameter called irredundance.) The idea is that a subset of edges is 1-minimal if (a) the set has the property, and (b) if one unmarks any one edge, then the set does not have the property. We already know how to determine if a *rem* tree satisfies (a). To determine if it satisfies (b), it suffices to know all possible classes that can arise when one unmarks a marked edge. Thus, one associates with each *rem* tree the pair (a, B) where a is the class of that tree,

and B is the set of all classes that are reachable by unmarking one edge. The table for the pairs can be constructed directly from the original table. To abuse notation:

$$(a, B) \diamond (c, D) = (a \diamond c, E) \text{ where } E = [a \diamond D] \cup [B \diamond c] \cup [a \blacklozenge b], \text{ and}$$
$$(a, B) \blacklozenge (c, D) = (a \blacklozenge c, F) \text{ where } F = [a \blacklozenge D] \cup [B \blacklozenge c].$$

If the original had m classes, then the new table has up to $m2^m$ classes. But, many of these classes cannot occur, while a few can never occur as subtrees in a valid tree, and a few others are equivalent.

Assuming our programming is correct (which has been tested with dozens of parameters), then we get the compositions presented in Table 3. The right-hand column gives the (a, B) for each class, and can be used to provide English descriptions of each class (see [8]).

	1	2	3	4	5	6	7	8	9	10	*origin*
1	3:5	3:-	-:4	-:6	-:9	1:7	2:10	1:10	1:-	2:-	(1, {})
2	-:-	-:-	-:-	-:-	-:-	2:-	2:-	2:-	2:-	2:-	(1, {2})
3	3:5	3:-	-:4	-:6	-:9	3:8	-:9	3:9	3:-	-:-	(2, {})
4	4:9	4:-	-:6	-:6	-:-	4:9	-:-	4:-	4:-	-:-	(3, {})
5	5:9	5:-	-:9	-:-	-:-	5:-	-:-	5:-	5:-	-:-	(3, {2})
6	6:-	6:-	-:6	-:6	-:-	6:-	-:-	6:-	6:-	-:-	(4, {})
7	8:-	8:-	-:9	-:-	-:-	7:-	-:-	7:-	7:-	-:-	(4, {1})
8	8:-	8:-	-:9	-:-	-:-	8:-	-:-	8:-	8:-	-:-	(4, {2})
9	9:-	9:-	-:-	-:-	-:-	9:-	-:-	9:-	9:-	-:-	(4, {3})
10	9:-	9:-	-:-	-:-	-:-	10:-	-:-	10:-	10:-	-:-	(4, {1, 3})

Table 3: Table for Minimal Total Edge Dominating Sets

A set of recurrences can be derived from Table 3 as before. For example, once the algorithm completes, one should end in class 1, 2, 6, 8, or 9 in order to have a minimal total edge dominating set.

4.2. Application

As an application, the automated software instantly shows that:

Theorem 1. *For $n \geq 3$, the maximum value of $\Gamma'_t(T)$ over all trees T of order n is exactly $2\lfloor n/3 \rfloor$.*

This result appears to be new, though it is not hard to provide a typical inductive proof as well. The path is an example where the bound is attained.

5. Conclusion

The Wimer edge variant opens the door to the solution of many edge-based problems in trees. Several problems that one could consider include induced matchings, the lower connected matching number, the lower disconnected matching number, and other parameters such as the lower open irredundance number; many of these are listed in [7]. The variant also suggests its possible use to automate conjecture creation and theorem proving. We expect that much of this generalizes to more complex recursive structures.

References

[1] A. Berger and O. Parekh, Linear time algorithms for generalized edge dominating set problems, *Algorithmica*, **50**(2008),244–254.

[2] M. Bern, E. Lawler and A. Wong, Linear-time computation of optimal subgraphs of decomposable graphs, *J. Algorithms*, **8**(2)(1987), 216-235.

[3] R. B. Borie and R. G. Gary and C. A. Tovey, Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families, *Algorithmica*, **7**(1992), 555-581.

[4] B. Courcelle, Recognizable sets of unrooted trees, In *Tree Automata and Languages,* North-Holland, Amsterdam, 141–158, 1992.

[5] W. Goddard, Automated bounds on recursive structures, *Utilitas Math.,* **75**(2008), 193–210.

[6] W. Goddard and S. T. Hedetniemi, Trees, tables and algorithms, *Networks*, (To appear)

[7] S. T. Hedetniemi, Unsolved algorithmic problems on trees, *AKCE J. Graphs. Combin.*, **3**(1)(2006), 1-37.

[8] A. C. Jamieson, *Linear time algorithms on edge-based problems*, Ph.D. thesis, Clemson University, Clemson, SC, 2007.

[9] D. F. Manlove, On the algorithmic complexity of twelve covering and independence parameters of graphs, *Discrete Appl. Math.*, **91**(1999), 155–175.

[10] S. L. Mitchell and S. T. Hedetniemi, Edge domination in trees, *Congr. Numer.*, **19**(1977), 489-509.

[11] D. B. West, *Introduction to Graph Theory*, Prentice Hall, NJ, USA,2nd edition, 2001.

[12] T. V. Wimer, S. T. Hedetniemi and R. Laskar, A methodology for constructing linear graph algorithms, *Congr. Numer.*, **50**(1985), 43-60.

[13] T. V. Wimer, *Linear algorithms on k-terminal graphs*, Ph.D. thesis, Clemson University, Clemson, SC, 1987.

[14] M. Zito, Linear time maximum induced matching algorithm for trees, *Nordic J. Comput.*,**7**(2000), 58-63.