Efficient Transitive Closure of Sparse Matrices over Closed Semirings

Gerald Penn
Department of Computer Science
University of Toronto
gpenn@cs.toronto.edu

Keywords: sparse matrix multiplication, semirings, typed feature logic, parsing

1 Background

Let M be an $n \times n$ matrix over the Boolean semiring. We can elect to interpret this matrix as an adjacency representation of a finite directed acyclic graph on n nodes, where $M[i,j] = 1$ iff there is an arc from node i to node j. Under this interpretation, adding in the identity matrix and multiplying the result by itself until it reaches a fixed point, M^*, corresponds to computing the path accessibility relation in the graph. In the category of partial orders, this amounts to computing the reflexive-transitive closure of a cover relation. For brevity, we can refer to M^* as the *transitive closure* of M.

In computational linguistics, this computation is necessary for computing the subtype relation in HPSG type signatures. By generalizing semirings to semi-lattices of more than two elements (1 and 0), we can similarly compute all of the feature appropriateness conditions for defining feature structure unification from HPSG feature declarations (Penn, 2000). Here, we left-multiply a $t \times f$ matrix of declarations by the transitive closure of this generalized subtyping matrix, where t is the number of types in the grammar and f is the number of features.

Let M be an $n \times n$ matrix over a non-commutative semiring in which the carrier is the powerset of some finite set, N, and addition is set-union. We can interpret the elements of N as the set of non-terminals of a context-free grammar plus a distinguished unit element for the "empty category," and multiplication as a powerset extension of the phrase-structure rules of that grammar. Then constructing M^* corresponds to deriving an all-paths parse forest over an $(n-1)$-word input string: $M^*[i,j] = S \subseteq \mathcal{P}(N)$ iff every non-terminal of S can be rewritten to $w_{i+1} \ldots w_j$ (Goodman, 1999). A similar generalization for adjacency representations lets us compute path accessibility in labelled graphs, such as graphical lexical semantic representations like WordNet.

In practical contexts, we expect all of these linguistic applications to begin with a matrix M that is very sparse, and end with a matrix M^* that is still fairly sparse — crucially, there exists a topological ordering of the n vertices implicit in M's graph-theoretic interpretation such that M can be rendered upper-triangular, resulting in a worst-case $(n^2 + n)/2$ non-zero elements.

In this talk, an alternative representation of matrices will be presented that allows for the efficient computation of transitive closure over closed semirings. An empirical evaluation will also be provided.

2 Related Work

Warshall (1962) developed his famous algorithm for transitive closure on the Boolean semiring. Floyd (1962) developed his for transitive closure on the tropical semiring (which computes the solution to the weighted all-pairs-shortest-path problem). Neither of these pays any attention to sparseness. There are, of course, many algorithms that do, cf. Press et al. (1993), including the Yale algorithm, which is used in the empirical comparisons given below. Sparse matrix multiplication algorithms, however, are almost always designed for matrices over integer-or-real-valued rings,[1] and are generally not developed with transitive closure specifically in mind.

Many semirings, including the Boolean semiring, can be embedded into another ring ($\mathbb{R}$, in the case of the Boolean semiring) in which matrix multiplication preserves enough structure for an answer in the original semiring to be recovered. Lee (2002) tacitly relies on this in her more theoretical discussion of Boolean matrix multiplication and parsing. From a practical perspective, this would allow us to use a standard sparse matrix multiplication algorithm.

In the specific case of transitive closure, there is a good reason not to do this, even when the embedding exists. In a closed semiring (Aho et al., 1974):

$$(*) \quad \begin{pmatrix} A & B \\ 0 & C \end{pmatrix}^* = \begin{pmatrix} A^* & A^*BC^* \\ 0 & C^* \end{pmatrix}$$

This reduction does not, by itself, yield a sub-cubic transitive closure algorithm, but it does reduce the number of basic operations by a constant 75%. The present research programme began as an attempt to provide a sparse matrix multiplication algorithm directly for closed semirings that could avail itself of this property.

Note: In the case of rings, Strassen's algorithm and all other non-probabilistic sub-cubic algorithms have very large constant factors that make them realistic only for very dense matrices. In the case of semirings, the Four Russians' algorithm is more co-operative in this respect, but is also less well-suited to large sparse matrices.

3 Zero-Counting by Quadrants

For simplicity, we will consider only the Boolean semiring in the development here. For any i and n such that $1 \leq i \leq n$, let $d_n(i)$ be defined such that:

$$d_n(i) = \begin{cases} 0 & i = 1 \\ d_{\lceil n/2 \rceil}(i) + 1 & 1 < i \leq \lceil n/2 \rceil \\ d_{\lfloor n/2 \rfloor}(i - \lceil n/2 \rceil) + 1 & i > \lceil n/2 \rceil \end{cases}$$

In addition, for any $d \geq d_n(i)$, let $q_n^d(i)$ be defined such that:

$$q_n^d(i) = \begin{cases} n & d = 0 \text{ (thus } i = 1) \\ q_{\lceil n/2 \rceil}^{d-1}(i) & d > 0, i \leq \lceil n/2 \rceil \\ q_{\lfloor n/2 \rfloor}^{d-1}(i - \lceil n/2 \rceil) & d > 0, i > \lceil n/2 \rceil \end{cases}$$

We will use these functions to recursively divide a matrix evenly into quadrants. One way of looking at them is as measures defined on a balanced binary tree with n leaves. Given the ith leaf from the left, there will be some subtrees for which i is the leftmost leaf. In that case, $d_n(i)$ is the least depth of such a subtree, and $q_n^d(i)$ is the total number of leaves that such a subtree at depth d has.

Given a matrix M, we shall say that a submatrix is *rooted* at $M[i,j]$ iff $[i,j]$ is its leftmost, uppermost coordinate in M.

Given i, j, m and n such that $1 \leq i \leq m$, and $1 \leq j \leq n$, let $v_{\langle m,n \rangle}(i,j) = \max(d_m(i), d_n(j))$. When we divide an $m \times n$ matrix M evenly into quadrants, then the largest quadrant rooted at $M[i,j]$ will be $q_m^{v_{\langle m,n \rangle}(i,j)}(i) \times q_n^{v_{\langle m,n \rangle}(i,j)}(j)$ in size. It can be proven that if m and n differ by no more than 1, then these two dimensions will differ by no more than 1.

Now, given a matrix M over the Boolean semiring, there is a unique matrix $Z(M)$ over the non-negative integers such that the value of $Z(M)[i,j]$ is the size of the largest zero-quadrant rooted at $M[i,j]$ in the largest quadrant rooted at $M[i,j]$. If this largest zero-quadrant is not square, then we use the larger of its dimensions as the value. As a simple example, consider the

[1] The present author has yet to find even one exception to this statement.

4×4 identity matrix as M:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad Z(M) = \begin{pmatrix} 0 & 1 & 2 & 1 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Note that the 1s in M are replaced by 0s in $Z(M)$ — there are no zero-quadrants rooted at those coordinates. Also notice that many values of $Z(M)$ can be inferred from other values. The fact that $Z(M)[1,3]$ is 2, for example, tells us that $Z(M)[1,4]$, $Z(M)[2,3]$, and $Z(M)[2,4]$ must be 1, and *vice versa*. It is perhaps useful to conventionally write $Z(M)$ with as few values as can be used to infer the rest of the matrix:

$$Z(M) = \begin{pmatrix} 0 & 1 & 2 & - \\ 1 & 0 & - & - \\ 2 & - & 0 & 1 \\ - & - & 1 & 0 \end{pmatrix}$$

This convention accentuates the sparseness of the original matrix M.

4 Transitive Closure with ZCQ

To transitively close a matrix in its ZCQ-representation, we first recurse on its diagonal quadrants, as suggested by (*), to obtain A^* and C^*. We then compute A^*BC^* with two matrix multiplications. Matrix multiplication in ZCQ is given by the quadrant-based recursive formulation:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{pmatrix}$$

Summation in ZCQ is given by a coordinate-wise *min* operation. In addition to the size-one base cases, an efficient implementation of ZCQ would include in both summation and multiplication a *sparse case*, in which the value of $Z(M)$ is checked first against the dimensions of the submatrices being multiplied. In this context, the base case of multiplication thus always returns 0 (indicating a non-zero element).

It is, in fact, possible to compute A^*BC^* simultaneously with two recursive functions:

$\mathbf{out(A, B, C) : computes\ B := A^*BC^*}$

$B_{22} := in(B_{21}, C_{11}, C_{12}, B_{22})$

$B_{12} := in(A_{12}, A_{22}, B_{22}, B_{12})$
$B_{12} := in(B_{11}, C_{11}, C_{12}, B_{12})$
$out(A_{11}, B_{12}, C_{22})$

$out(A_{22}, B_{22}, C_{22})$

$B_{11} := in(A_{12}, A_{22}, B_{21}, B_{11})$
$out(A_{11}, B_{11}, C_{11})$

$out(A_{22}, B_{21}, C_{11})$

$$\mathbf{R} := \mathbf{in}(\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{R^0}) : \textbf{computes } \mathbf{R} := \mathbf{AB^*C} + \mathbf{R^0}$$

$$\text{temp} := in(A_{11}, B_{11}, B_{12}, A_{12})$$

$$R_{11} := in(A_{11}, B_{11}, C_{11}, R^0_{11})$$
$$R_{11} := in(\text{temp}, B_{22}, C_{21}, R_{11})$$

$$R_{12} := in(A_{11}, B_{11}, C_{12}, R^0_{12})$$
$$R_{12} := in(\text{temp}, B_{22}, C_{22}, R_{12})$$

$$\text{temp} := in(A_{21}, B_{11}, B_{12}, A_{22})$$

$$R_{21} := in(A_{21}, B_{11}, C_{11}, R^0_{21})$$
$$R_{21} := in(\text{temp}, B_{22}, C_{21}, R_{21})$$

$$R_{22} := in(A_{21}, B_{11}, C_{12}, R^0_{22})$$
$$R_{22} := in(\text{temp}, B_{22}, C_{22}, R_{22})$$

where the transitive closures are again assumed to apply to upper-triangular matrices only. These functions decompose their arguments by quadrant, as does ZCQ. $in/4$ has complexity $\mathcal{O}(n^{\log 10})$ because of the number of recursive calls in its definition. One might expect that sparseness would still favour performing two simultaneous closures and multiplications in this way since even a single zero argument produces a zero result. Experimentally, this has proven not to be the case: this method is over 1000 times slower than performing the left and right multiplications separately, as described earlier, with even modest values of n. It also consumes more memory, although total memory consumption by the temporary buffer, temp, in computing the closure of an $n \times n$ matrix is bounded above by $\frac{1}{3}n(2n+5)$.

5 Evaluation

So far, this representation and algorithm have only been partially evaluated on Boolean semiring closures. For this, the typing partial orders from three HPSG grammars are being used: the one distributed with ALE, and two versions distributed by the English Resource Grammar (ERG) project. Times are given in milliseconds, and were measured as an average of three runs on a

	ALE(162)	Baby-ERG(2763)	ERG(4305)
Naive-BOR	315	3600870	$> 5.8 \times 10^7$
Naive-Z*	7	271488	865742
Naive-BOR*	3	225698	570967
Yale-BOR	<1	2992	6865
Yale-Z	<1	1579	3573
ZCQ	<1	425	547

Table 1: Preliminary evaluation of ZCQ. Times are in milliseconds.

Dell dual-Xeon 2.4GHz server with 2GB of RAM available. The two asterisked methods were obtained by using naive matrix multiplication over the integer ring and the Boolean semiring, respectively, in combination with the reduction given in (*) above. The Yale algorithm is based on a row-indexed representation scheme, and cannot straightforwardly be adapted to use (*). All implementations are those of the present author.

Why is ZCQ so fast? A preliminary analysis of its performance reveals that the step in which zero subquadrants are re-combined to form larger zero quadrants is crucial to efficient performance — without it, ZCQ is roughly as fast as the Yale algorithm. In their lattice-theoretic interpretation, large zero quadrants of size $n \times n$ occur off-diagonally as representatives of anti-chains of n elements. Large $n \times n$ quadrants consisting only of non-zero elements correspond to a number of other structures, ranging from the generalised crown S_n, in which case the matrix has zeros in its remaining positions, to a chain of length $2n$, in which case the quadrant would be contained in a $2n \times 2n$ triangle of exclusively non-zero elements. The ERG contains few if any generalised crowns, and its largest chain is of length 21. We may thus conjecture that the success of ZCQ on these data is a result of HPSG's predilection for short, "bushy," sparsely joined partially ordered sets.

6 Future Work

The above evaluation needs to be extended to a wider range of partial orders, from the domain of computational linguistics and elsewhere, as well as a wide range of synthetically generated cases. In addition, ZCQ should be adapted and tested with other closed semirings, including those that can be used for parsing. These two extensions will permit us to engage in a finer-grained analysis of ZCQ's practical efficiency. It seems rather fortuitous that decomposing matrices into roughly square quadrants would turn out to be the optimal choice of shape into which M could be cut. It may very well be that there is some degree of flexibility there that would allow us to tailor the quadrants of M according to some prior knowledge of the domain from which the interpretation of M arises.

References

Aho, A., Hopcroft, J., and Ullman, J. (1974). *The Design and Analysis of Computer Algorithms*. Addison-Wesley.

Floyd, R. W. (1962). Algorithm 97 (shortest path). *Communications of the ACM*, 5(6):345.

Goodman, J. (1999). Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Lee, L. (2002). Fast context-free grammar parsing requires fast Boolean matrix multiplication. *Journal of the ACM*, 49(1):1–15.

Penn, G. (2000). The algebraic structure of transitive closure and its application to attributed type signatures. *Grammars*, 3(2–3):295–312.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1993). *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, 2nd edition.

Warshall, S. (1962). A theorem on Boolean matrices. *Journal of the ACM*, 9(1):11–12.