An Efficient Representation for Sparse Graphs

Gabriel Valiente

Department of Software, Technical University of Catalonia, E-08034 Barcelona, Spain, Fax 00 34 934 017 014, valiente@lsi.upc.es

Abstract. The design of efficient graph algorithms usually precludes the test of edge existence, because an efficient support of that operation already requires time $\Omega(n^2)$ for the initialization of an adjacency-matrix representation. We describe an alternative representation of static directed graphs taking $\Theta(n+m)$ initialization time and using $\Theta(n^2)$ space, which supports the efficient implementation of all usual operations on static graphs. The sparse graph representation allows the design of efficient graph algorithms using both iteration over all vertices adjacent with a given vertex and edge-existence operations, although at the expense of additional (uninitialized) space which may, nevertheless, be used for other purposes. To the best of our knowledge, the representation leads to the first graph algorithms with the disconcerting property that the time complexity is better than the space complexity.

1 Introduction

In the asymptotic analysis of algorithms, efficiency is measured as a function of input size. In the case of graph algorithms, graphs are usually assumed to be represented either by adjacency matrices or by adjacency lists. A graph algorithm runs in linear time if it can be implemented to run in time linear in the size of the representation of the graph, and it takes quadratic $\Theta(n^2)$ time and linear $\Theta(n+m)$ time, respectively, to initialize an adjacency-matrix and an adjacency-list representation for a graph with n vertices and m edges. Therefore, a graph algorithm cannot be implemented to run in linear time using an adjacency-matrix representation.

Adjacency lists are the representation of choice for the implementation of most graph algorithms, because they only take $\Theta(n+m)$ space and support iteration over all vertices adjacent with a given vertex in time linear in the degree of the vertex. Adjacency matrices, on the other hand, take $\Theta(n^2)$ space and support iteration over all vertices adjacent with a given vertex in $\Theta(n)$ time, but support edge-existence test in $\Theta(1)$ time. These iterators over adjacent vertices are fundamental to the edge-based programming of graph algorithms [4]. Notice that neither bit-vector representations of adjacency matrices nor implicit representations of graphs [1, 5, 6, 9–11] do improve these asymptotic time bounds, as long as both iteration over all vertices adjacent with a given vertex and edge-existence test need to be supported.

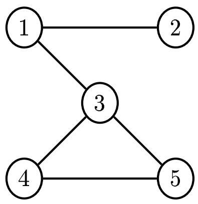
In this paper, we combine the best of both worlds by showing how a simple programming technique, known as *key indexing* and used in [2] for representing sparse sets, allows the representation of static graphs in $\Theta(n+m)$ time and $\Theta(n^2)$ space while supporting iteration over all vertices adjacent with a given vertex in time linear in the degree of the vertex, and also supporting edge-existence test in $\Theta(1)$ time.

The main interest of the sparse graph representation lies in the design of efficient graph algorithms without the need of avoiding edge-existence tests, although at the expense of additional (uninitialized) space which can, nevertheless, be used for other purposes. The sparse graph representation is, to the best of our knowledge, the first technique leading to graph algorithms with the disconcerting property that the time complexity is better than the space complexity.

2 The Graph Representation

A (directed) graph $G = (V, E)$ consists of a finite nonempty set V of vertices and a finite set $E \subseteq V \times V$ of edges. A graph $G = (V, E)$ with n vertices and m edges is called *sparse* if $m \in O(n)$, and it is called *dense* if $m \in \Omega(n^2)$. Let vertices be named, without loss of generality, by the ordered set $V = [1, \ldots, n]$ for a graph $G = (V, E)$ with n vertices, inducing thus an order on the set of edges incident with each vertex, and let $rank(i, j)$ denote the rank of vertex j in the ordered set of edges incident with vertex i. Let also $\deg(i)$ denote the out-degree of vertex i, that is, $\deg(i) = |\{j \mid (i, j) \in E\}|$, for $1 \leqslant i \leqslant n$.

The sparse representation of a graph $G = (V, E)$ with n vertices and m edges consists of two $n \times n$ integer matrices *sparse* and *dense*, together with a vector *deg* of n integer entries. Both matrices only have m initialized entries. Initialized entries in *dense* point to entries in *sparse*, which point back into *dense*. The values of other entries in *dense* and *sparse* are unimportant and, as a matter of fact, they are never initialized. Fig. 1 illustrates the sparse representation of a sample graph.



	1	2	3	4	5
1	0	1	1	0	0
2	1	0	0	0	0
3	1	0	0	1	1
4	0	0	1	0	1
5	0	0	1	1	0

	1	2	3	4	5
1		1	2		
2	1				
3	1			2	3
4			1		2
5			1	2	

sparse

	1	2	3	4	5
1	2	3			
2	1				
3	1	4	5		
4	3	5			
5	3	4			

dense

1	2
2	1
3	3
4	2
5	2

deg

Fig. 1. Adjacency matrix (top) and sparse (bottom) representation of a sample graph.

Definition 1 *The sparse representation of a graph $G = (V, E)$ with n vertices is a triple $(sparse, dense, deg)$, where sparse and dense are $n \times n$ integer matrices and deg is a vector of n integer entries, with $sparse[i, j] = rank(i, j)$ and $dense[i, sparse[i, j]] = j$ for all edges $(i, j) \in E$, and with $deg[i] = deg(i)$ for all vertices $i \in V$.*

Correctness of the sparse graph representation can be established in a straightforward way.

Proposition 2 *Let $(sparse, dense, deg)$ be the sparse representation of a graph $G = (V, E)$ with n vertices. Then, $(i, j) \in E$ if and only if $1 \leqslant sparse[i, j] \leqslant deg[i]$ and, moreover, $dense[i, sparse[i, j]] = j$, for all $1 \leqslant i, j \leqslant n$.*

Proof (Proof Sketch). Suppose that, due to the random contents of memory, it happens that $1 \leqslant sparse[i, j] \leqslant deg[i]$ for some $(i, j) \notin E$.

Then, it must be $dense[i, sparse[i,j]] \neq j$, for otherwise it would be $(i,j) \in E$ by Def. 1, contradicting the hypothesis.

The sparse graph representation allows the implementation of edge-existence test in $\Theta(1)$ time. Roughly stated, for an absent edge $(i,j) \notin E$ first the uninitialized element $sparse[i,j]$ is accessed. The value found there can be smaller that 1 or larger than $deg[i]$, in which case the absence of (i,j) is established, or be between 1 and $deg[i]$ inclusive, in which case $dense[i, sparse[i,j]]$ is consulted, but since $(i,j) \notin E$, $dense[i, sparse[i,j]]$ cannot be equal to j and again, the absence of (i,j) is established.

Furthermore, iteration over all vertices adjacent with a given vertex i can be implemented in $\Theta(\deg(i))$ time by just accessing elements $dense[i,1], \ldots, dense[i, deg[i]]$.

	1	2	3	4	5
1		1	2		
2	3				
3	4			5	6
4			7		8
5			9	10	

sparse

1	1
2	3
3	4
4	7
5	9

low

1	2
2	3
3	6
4	8
5	10

high

1	2	3	4	5	6	7	8	9	10
2	3	1	1	4	5	3	5	3	4

compact

Fig. 2. Compact sparse representation of the sample graph in Fig. 1.

The sparse graph representation also allows the implementation of edge insertion in $\Theta(1)$ time. A more compact representation can be used, though, when edge insertion is not needed, consisting in replacing matrix *dense* by a vector *compact* of m integer entries, and vector *deg* by two vectors *low* and *high* of n integer entries. Fig. 2 illustrates the compact sparse representation.

Lemma 3 *Let $(sparse, compact, low, high)$ be the compact sparse representation of a graph $G = (V,E)$ with n vertices. Then, $(i,j) \in E$ if and only if $low[i] \leqslant sparse[i,j] \leqslant high[i]$ and $compact[sparse[i,j]] = j$, for all $1 \leqslant i,j \leqslant n$.*

In the compact sparse representation, iteration over all vertices adjacent with a given vertex i can also be implemented in $\Theta(\deg(i))$ time by just accessing elements $compact[low[i]], \dots, compact[high[i]]$.

Notice that vectors *compact*, *low*, and *high* together are, as a matter of fact, a representation of static directed graphs similar to the one proposed in [8, Ex. 6.14-2].

The additional, uninitialized space allocated by the sparse graph representation can be used for other purposes. For instance, in the context of an application involving a large, but constant, number of graphs, the *sparse* and *dense* matrices of the smaller graphs can be stored in the uninitialized entries of the *sparse* and *dense* matrices of the larger graphs, leading thus to a significant reduction in the additional space required by the sparse graph representation.

3 Applications

The usefulness of the sparse graph representation is illustrated in this section by way of example. The *celebrity problem* [7, Sect. 5.5] consists in determining whether there is a *celebrity* (someone who is known by everyone else, and does not know anyone else) among a group of n people by asking questions of the form: "excuse me, do you know the person over there?"

In graph-theoretical terms, the celebrity problem can be modeled as a directed graph $G = (V, E)$, where V corresponds to the group of n people and there is an edge $(i, j) \in E$ if and only if person i knows person j. A celebrity corresponds to a *universal sink* (a vertex with in-degree $n - 1$ and out-degree zero). Notice that a directed graph can have at most one universal sink.

The celebrity problem was also posed in [3, Ex. 23.1-6] as an example of a graph algorithm running in $\Theta(n)$ time even if an adjacency-matrix representation is used. Given a problem instance in the adjacency-list representation, it is clear that the solution can be found in $\Theta(n + m)$ time and $\Theta(n)$ additional space, by computing the in-degree and out-degree of all vertices, taking thus $\Theta(n^2)$ time for dense graphs. Given a problem instance in the sparse graph representation, though, the solution can be found in $\Theta(n)$ time, as follows.

A simple and efficient method for solving the celebrity problem consists of discarding people who cannot be celebrities. If the answer to question "excuse me x, do you know y?" is *yes*, then x cannot be a celebrity, and if the answer is *no*, then it is y the one who cannot be a celebrity. Then, the answer to each such question allows discarding one person. Assume that x is eliminated and find, by induction, a celebrity among the remaining $n-1$ people. If there is no celebrity among the remaining $n-1$ people, then there is no celebrity among the n people. Otherwise, it must be checked that x knows the celebrity and that the celebrity does not know x.

Algorithm 1 Determining whether a directed graph $G = (V, E)$ with $n \geqslant 2$ vertices contains a universal sink.

```
1: i := 1
2: for all j from 2 to n do
3:   if (i, j) ∈ E then
4:     i := j
5:   end if
6: end for
7: for all j from 1 to n do
8:   if (i, j) ∈ E or ((j, i) ∉ E and i ≠ j) then
9:     return false (there is no universal sink)
10:  end if
11: end for
12: return true (vertex i is the universal sink)
```

Alg. 1 is an efficient encoding of the previous method. The algorithm is divided in two phases. In the first phase (lines 1–6), all but one candidate are discarded, and in the second phase (lines 7–12), it is checked whether the remaining candidate is indeed the celebrity. (Notice that in lines 3–5, either i or j is discarded. If $(i, j) \in E$, that is, if i knows j, then i is discarded. Otherwise, j is discarded.) It is clear that the algorithm can be implemented to run in $\Theta(n)$ time and $\Theta(n^2)$ space, on graphs given in the sparse graph representation.

Notice that the problem instance is assumed to be given in the most appropriate representation. Otherwise, if the problem instance is given in adjacency-list representation, the trivial solution would take $\Theta(n+m)$ time and $\Theta(n)$ additional space while the previous solution would take $\Theta(n+m)$ time and $\Theta(n^2)$ space. If the problem instance is given in adjacency-matrix representation, the trivial solution would take $\Theta(n^2)$ time and $\Theta(n)$ additional space while the previous solution would still take $\Theta(n+m)$ time and $\Theta(n^2)$ space.

4 Conclusions

The sparse graph representation combines the best of adjacency lists and adjacency matrices, and allows to represent static graphs in $\Theta(n+m)$ time and $\Theta(n^2)$ space while supporting iteration over all vertices adjacent with a given vertex in time linear in the degree of the vertex, and also supporting edge-existence test in constant time. The representation can be easily extended to associate additional information with vertices and edges.

The main interest of the sparse graph representation lies in the fact that it allows the design of efficient graph algorithms without the need of avoiding edge-existence tests, although at the expense of additional (uninitialized) space. The sparse graph representation is, to the best of our knowledge, the first technique leading to graph algorithms with the disconcerting property that the time complexity is better than the space complexity.

The additional, uninitialized space allocated by the sparse graph representation can be used for other purposes. For instance, in the context of an application involving a large, but constant, number of graphs, the matrices of the smaller graphs can be stored in the uninitialized entries of the matrices of the larger graphs, leading thus to a significant reduction in the additional space required by the sparse graph representation.

Acknowledgments

The author would like to acknowledge detailed comments by C. Martínez, R. Nieuwenhuis, and M. J. Serna on a preliminary version of this paper.

References

1. J. L. Balcázar and A. Lozano. The complexity of graph problems for succintly represented graphs. In *Proc. 15th Int. Workshop Graph-Theoretic Concepts in*

Computer Science, volume 411 of *Lecture Notes in Comput. Sci.*, pages 277–285. Springer-Verlag, 1989.
2. P. Briggs and L. Torczon. An efficient representation for sparse sets. *ACM Lett. Programming Lang. Syst.*, 2(1–4):59–69, 1993.
3. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.
4. J. Ebert. A versatile data structure for edge-oriented graph algorithms. *Commun. ACM*, 30(6):513–519, 1987.
5. J. Feigenbaum, S. Kannan, M. Y. Vardi, and M. Viswanathan. Complexity of problems on graphs represented as ordered binary decision diagrams. In *Proc. 15th Annual Symp. Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Comput. Sci.*, pages 216–226. Springer-Verlag, 1998.
6. H. Galperin and A. Wigderson. Succint representations of graphs. *Inform. Contr.*, 56(3):183–198, 1983.
7. U. Manber. *Introduction to Algorithms: A Creative Approach.* Addison-Wesley, 1989.
8. K. Mehlhorn and S. Näher. *The LEDA Platform of Combinatorial and Geometric Computing.* Cambridge University Press, 1999.
9. C. H. Papadimitriou and M. Yannakakis. A note on succint representations of graphs. *Inform. Contr.*, 71(3):181–185, 1987.
10. M. Talamo and P. Vocca. Representing graphs implicitly using almost optimal space. *Discrete Applied Mathematics*, 108(1–2):193–210, 2001.
11. J. van Leeuwen. Graph algorithms. In *Handbook of Theoretical Computer Science*, volume A, chapter 10, pages 525–631. Elsevier, 1990.