Compact Implicit Representation of Graphs *
(Extended Abstract)

Maurizio Talamo¹ and Paola Vocca²

¹ Dipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza”,
Via Salaria 113, I-00198 Rome, Italy.
talamo@dis.uniroma1.it

² Dipartimento di Matematica, Università di Roma “Tor Vergata”,
Via della Ricerca Scientifica, I-00133 Rome, Italy.
vocca@axp.mat.uniroma2.it

Abstract. How to represent a graph in memory is a fundamental data
structuring problem. In the usual representations, a graph is stored by
representing explicitly all vertices and all edges. The names (labels) as-
signed to vertices are used only to encode the edges and betray nothing
about the structure of the graph itself and hence are a “waste” of space.
In this context, we present a general framework for labeling any
graph so that adjacency between any two given vertices can be tested in constant
time. The labeling schema assigns to each vertex \( x \) of a general graph a
\( O(\delta(x) \log^3 n) \) bit label, where \( n \) is the number of vertices and \( \delta(x) \) is \( x \)'s
degree. The adjacency test can be performed in 5 steps and the schema
can be computed in polynomial time. This representation strictly con-
trasts with usual representations, i.e. adjacency matrix and adjacency
list representations, which require \( O(n \log n) \) bit label per vertex and
constant time adjacency test, and \( O(\delta(x) \log n) \) bit label per vertex and
\( O(\log \delta(x)) \) steps to test adjacency, respectively. Additionally, the labe-
ling schema is implicit, that is: no pointers are used.

1 Introduction

The representation of graphs has received much attention since the very be-
ginning of the study of computer science theory [2,3,18,6,21,11,16,22]. What is
generally required is a “good” and efficient representation: good for the efficiency
of algorithms running on it, and efficient both in terms of the space to store data
and the computational time needed to derive the representation.

In this context, we present a general framework for encoding any graph which
leads to an implicit representation, that is: no pointers are used, allowing to test
adjacency in a constant number of steps. In particular, the problem considered in
this paper can be stated as follows. Given any graph \( G = (V, E) \), with \( n = |N| \),
label the vertices such that, given the names of two vertices, we can determine

* Work partially supported by National Authority for Information Technology for the
Public Administration and the Italian Project MURST-“Algorithms and Data Struc-
true"
adjacency in $O(1)$ steps. The representation proposed assigns to each vertex $x$ a $O(\delta(x) \log^3 n)$ bit label, where $\delta(x)$ is the degree of $x$, allows to test adjacency in 5 steps, and can be computed in $O(n^3)$ time.

Implicit representations are widely accepted as good representations since implicit data structures lend themselves to a sequential storage scheme which requires no pointers, thus providing for a compact way to store data with no waste of space for pointers $[13, 4, 12, 15]$. Moreover, the algorithms are easier to implement and are often more efficient.

Additionally, the representation we proposed satisfies the property of “locality”, that is data are stored on a per-node basis, thus providing an efficient representation for distributed computation $[14]$ and secondary memory storage $[23]$.

Several authors worked on this problem. Breuer $[2]$ and Breuer and Folkman $[3]$ considered the problem of labeling vertices such that adjacency would be determined by the Hamming distance of the labels. Their schema is very restricted and for general graphs the length of the labels can be $O(n \log n)$. Turan $[18]$ and Kannan et al. $[11]$ considered the problem of representing a graph as succinctly as possible. However, they gave an efficient implicit representation for the adjacency list of the graph only for restricted classes of graphs: trees, graphs with bounded arboricity, intersection graphs, and $c$–decomposable graphs $[8]$. Additionally, the class of bounded treewidth can be represented with $O(b \log n)$ bit labels, where $b$ is the treewidth (see $[19]$).

The same problem has been studied in a different context, namely for routing messages in a distributed network. In $[16, 20, 21, 5, 6, 7, 8, 9]$ the problem considered is how to store routing information at the vertices of a distributed network so as to computer near–optimal routes. Again, the problem is optimally solved only for restricted class of graphs, as trees, rings, complete graphs, planar $st$–graphs, interval graphs, or for specific network topologies as hypercube, meshes.

In this paper, we propose a $k$–step labeling schema based on a set of $k$ mutually composable labeling functions, each one evaluable in 1 step. In order to test adjacency, the $k$ functions are evaluated sequentially, thus adjacency can be tested in $k$ steps.

In this scenario we obtain the following results:

1. a 3–step labeling schema for regular bipartite graph of degree $\delta$. The schema assigns $O(\delta \log^2 n)$ bit label to each vertex $x$;
2. a 5–step labeling schema for general graph. The schema assigns $O(\delta(x) \log^3 n)$ bit label to each vertex $x$, where $\delta(x)$ is the degree of $x$.

It is worth noting that if we compare our results with usual representation strategies, the amount of space on a per–node basis is increased of a $\log n$ factor for the former and a $\log^2 n$ factor for the latter. On the counterpart, the adjacency test can be performed in only 3 steps and 5 steps, respectively.

The proof strategy proceeds as follows: $i)$ the adjacency test problem for a general graph is reduced to an equivalent problem on a general bipartite graph (Section 5); $ii)$ the same problem for a general bipartite graph is reduced to an equivalent problem for a collection of regular bipartite graphs (Section 5);