Succinct Data-Structures for Labeled Graphs

Jérémy Barbay

David R. Cheriton School of Computer Science
University of Waterloo, Canada.

Technical Report CS-2006-48

Abstract. Succinct data-structures support efficient search queries while using space asymptotically optimal. Such data-structures are known for structures such as binary strings, ordinal and cardinal trees, graphs, binary relations, labeled and multi-labeled trees. We consider graphs where each node is associated to an arbitrary number of labels. We show that some categories of such graphs have a succinct encoding which supports efficiently label-based navigation and search operators.

Keywords: Labeled Graphs, Succinct Data-Structures.

1 Introduction

Succinct data-structures support efficient search queries while using space asymptotically optimal. Such data-structures are known for structures such as binary strings, ordinal [13] and cardinal trees, unlabeled graphs [13], and binary relations [1]. Those structures are not only asymptotically smaller and faster than the pointer based data-structures, they were also shown to be more efficient in practice [6, 8, 15].

Succinct data-structures can be divided in two categories: *succinct integrated encodings*, which encode the whole data itself in such a way to support operators efficently [1, 11]; and *succinct indexes*, which describe only the additional information required in order to support efficiently some operators [2, 5]. Succinct indexes and succinct integrated encodings are closely related, but they are different concepts: succinct indexes make assumptions only on the ADT through which the given data is accessed, while succinct encodings represent data in specific formats. The consequences of this difference of concept are that succinct indexes are more difficult to design (one can design a succinct encoding from a succinct index, but the converse is not true); succinct indexes support operators in sligthly worse times than succinct integrated encodings [2], lower bounds are easier to define for succinct indexes [10]; and succinct indexes can be freely combined whereas in general two succinct integrated encodings cannot be combined in a single one.

Combining the succinct data-structure (integrated encoding or index) for binary relations with another basic data-structure yields *labeled* versions of those structures. For instance, a multi-labeled tree is the combination of an ordinal tree with a binary relation which associates each node of the tree to a set of labels [1]. It is sufficient to combine the succinct encodings of the binary relation and the basic structure to encode the labeled structure, but supporting interesting operators requires extra work. For instance, in the case of multi-labeled trees, the support of the search for descendants of a node x with a given label α is reduced to a search in the binary relation which associates nodes to label, provided that the nodes are ordered in an order where all the descendants of x are consecutive [1].

[1] Contact the Corresponding Author at jbarbay@uwaterloo.ca

Jacobson, who introduced the concept of succinct encodings, defined an encoding for undirected planar graphs, and more generally for graphs of bounded book-thickness [13]. This encoding is based on the decomposition of the graph in a finite number of "pages", where each page can be encoded as a binary string of opening and closing parenthesis: the navigation in the graph is reduced to operators on these strings. This encoding uses space within a constant factor of the lower bound suggested by information theory.

We consider *multi-labeled graphs*, defined as graphs where each node is associated to a set of labels from a finite alphabet. Our main results are twofold:

- Combining the succinct index for binary relations with Jacobson's encoding for graphs of bounded book-thickness, we define an encoding for *multi-labeled graphs of bounded book depth*, which supports efficiently label-based navigation operators such as finding the nodes labeled α connected (resp. directly connected) to a node x.
- Observing that a bipartite graph is exactly a binary relation between two sets of nodes, we apply a scheme similar to the one described above to define a succinct encoding for *multi-labeled bipartite graphs*, which supports the same label-based operators. Similarly, observing that an oriented graph on n nodes is a $n \times n$ binary relation, we define a succinct encoding for *multi-labeled orirented graphs*, which supports the same label-based operators.

This paper is organized as follows: we outline the design of succinct data structures for several data types that we either use or relate to in Section 2. We describe the general scheme to combine a graph representation with a binary relation representaion in order to support label-based navigation operators on labeled graphs in Section 3, applying it to Jacobson's basic structure for graphs of bounded book-thickness. We conclude with some research perspective in Section 4.

All our results are in the Random Access Memory (RAM) model where words are of size $\Theta(\lg n)$, where n denotes the number of nodes in the graph. We assume that the nodes are all distinguishable, and in particular our use of the term "labeled graph" should not be confused with the definition of "unlabeled graphs" from Naor [17].

2 Previous Work

Succinct data structures were introduced by Jacobson [13], to encode bit vectors, tree structures and planar graphs in space essentially equal to the information-theoretic lower bound, while supporting appropriate operators on them efficiently. For *bit vectors*, Jacobson defined two useful operators: given a bit vector $B[1, \ldots, n]$, a bit $\alpha \in \{0, 1\}$, an object[1] $x \in [n]$ and an integer $r \in [n]$, the operator $\texttt{bin\_rank}_B(\alpha, x)$ returns the number of occurrences of α in $B[1, ..., x]$, and the operator $\texttt{bin\_select}_B(\alpha, r)$ returns the position of the r-th label α in B, or $-\infty$ if it does not exist. For $\alpha \in \{0, 1\}$, the operator $\texttt{bin\_rank}_B(\alpha, x)$ returns the number of occurrences of α in $B[1..x]$, and the operator $\texttt{bin\_select}_B(\alpha, r)$ returns the position of the r-th α in B. For conciseness, we omit the subscript B when it is clear from the context. Encodings for binary strings supporting those operators have been extensively studied, and we consider several complementary solutions depending of the application.

Lemma 1. *A bit vector B of length n with v 1s can be represented using either: (a) $n + o(n)$ bits [5, 13]; or (b) $\lg \binom{n}{v} + O(n \lg \lg n / \lg n)$ bits [18], to support the access to each bit,* `bin_rank` *and* `bin_select` *in $O(1)$ time.*

[1] We use $[i]$ to denote the set $\{1, 2, ..., i\}$.

A less powerful version of $\texttt{bin\_rank}(1, x)$, which we denote $\texttt{bin\_rank}'(1, x)$, returns -1 instead of the the number of 1s in $B[1..x]$ when $B[x] = 0$. It is still useful in some application, and uses less space:

Lemma 2 ([18]). *A bit vector B of length n with v 1s can be represented using $\lg \binom{n}{v} + o(v) + O(\lg \lg n)$ bits to support the access to each bit, $\texttt{bin\_rank}'(1, x)$ and $\texttt{bin\_select}(1, r)$ in $O(1)$ time.*

2.1 Ordinal Trees and Planar Graphs

An *ordinal tree* is a rooted tree in which the children of a node are ordered and specified by their ranks. Preorder and postorder traversals of such trees are well-known. We also use a different order for traversals, namely `DFUDS`, which is the order associated with the *depth first unary degree sequence* [3] representation, where all the children of a node are listed before its other descendants.

Various succinct data structures were designed to represent ordinal trees [3, 9, 13, 14, 16]. Benoit *et al.* [3] proposed the `DFUDS` representation of an ordinal tree using $2n + o(n)$ bits to support various navigational operations, which is close to the lower bound suggested by information theory ($2n - \Theta(\lg n)$ bits). Jansson *et al.* [14] extended this representation to support a richer set of navigational operations. The operations that we consider in this paper are as follows (we refer to each node by its preorder number):

- $\texttt{child}(x, i)$, i-th child of node x for $i \geq 1$;
- $\texttt{child\_rank}(x)$, number of left siblings of node x;
- $\texttt{depth}(x)$, depth of node x, i.e. the number of edges in the rooted path to x;
- $\texttt{level\_anc}(x, i)$, i-th ancestor [2] of node x for $i \geq 0$;
- $\texttt{nbdesc}(x)$, number of descendants of node x;
- $\texttt{degree}(x)$, degree of node x, i.e. the number of its children.

Benhart and Kainen [4] defined book embeddings of undirected graphes, which specify a unique ordering of the vertices of the graph, and partition the edges into pages such that the edges on any given page do not intersect. Using his succinct encoding for binary strings, Jacobson [13] showed how to encode each page of such a decomposition as a binary string of well balances parenthesis. In this encoding, the support for navigation operators in the subgraph corresponding to each page is reduced to operation on parenthesis, which are supported in constant time: As four pages are sufficient to represent planar graphs [19], this encoding supports the navigation operators in constant time.

2.2 Strings and Binary Relations

Grossi *et al.* [12] generalized the operators `bin_rank` and `bin_select` to a string (or a sequence) S of length n over an alphabet of arbitrary size σ, and the operations include: $\texttt{string\_rank}(\alpha, x)$, which returns the number of occurrences of α in $S[1..x]$; $\texttt{string\_select}(\alpha, r)$, which returns the position of the r-th occurrence of α in the string; and $\texttt{string\_access}(x)$, which returns the character at position x in the string. They gave an encoding that takes $nH_0 + o(n \lg \sigma)$ bits to support these three operators in $O(\lg \sigma)$ time, where n is the length of the string. Golynski *et al.* [11] gave an encoding that uses $n(\lg \sigma + o(\lg \sigma))$ bits and supports $\texttt{string\_rank}(\alpha, x)$ and $\texttt{string\_access}(x)$ in $O(\lg \lg \sigma)$ time, and $\texttt{string\_select}(\alpha, r)$ in constant time.

[2] Given a node x at depth d, its i-th ancestor is the ancestor of x at depth $d - i$.

Barbay *et al.* [1] extended the problem to the encoding of sequences of n objects where each object can be associated with a subset of labels from $[\sigma]$, this association being defined by a binary relation of t pairs from $[n]\times[\sigma]$. The operations include: `label_rank`(α, x), which returns the number of objects labeled α up to (and including) x; `label_select`(α, r), which returns the position of the r-th object labeled α; and `label_access`(x, α), which checks whether object x is associated with label α. Their representation supports `label_rank` and `label_access` in $O(\lg\lg\sigma)$ time, and `label_select` in constant time using $t(\lg\sigma + o(\lg\sigma))$ bits.

Barbay *et al.* [2] defined a succinct index which does not constraint how the string or binary relation is encoded, and which supports the same operators in time only slightly worse. This is particularly important when using binary relations to code the association between the labels and the nodes of a tree or graph, as various operators require the support for operators in various orders on the nodes, the binary relation is encoded only once while each order corresponds to a different succinct index.

Barbay *et al.* [2] also showed how to extend the data-structure to support a limited version of "negative" searches, ons string and binary relations:

Definition 1. *An object* $x \in [n]$ matches the literal $\alpha \in [\sigma]$ *if* x *matches the label* α*, and* matches the literal $\bar{\alpha}$ *if it does not match the label* α*. For simplicity, we define* $[\bar{\sigma}]$ *to be the set* $\{\overline{1}, \ldots, \overline{\sigma}\}$.

Their encoding does not support the select operator on literals, but rather the operators `string_succ`(α, x) and `string_pred`(α, x), which return the first object matching the label α respectively after and before x: those operators are sufficient for most applications.

2.3 Labeled and Multi-Labeled Trees

A *labeled tree* is an ordinal tree in which each node is associated with a label from a given alphabet $[\sigma]$; while in a *multi-labeled tree*, each node is associated with at least one label. We use n to denote the number of nodes in a tree, and t to denote the total number of node-label pairs in a multi-labeled tree.

Geary *et al.* [9] defined labeled extensions of the first six operators defined in Section 2.1. Their data structures support those in constant time, but use $2n + n(\lg\sigma + O(\sigma\lg\lg\lg n/\lg\lg n))$ bits, which is much more than the asymptotic lower bound of $n\,(\lg\sigma - o(\lg\sigma))$ suggested by information theory when σ is large. Ferragina *et al.* [7] proposed another structure for labeled trees that supports locating the first child of a given node x labeled α in constant time, and finding all the children of x labeled α in constant time per child. But it does not efficiently support the retrieval of the ancestors or descendants by labels. Also it uses $2n\lg\sigma + O(n)$ bits, which is almost twice the minimum space required to encode the tree. Barbay *et al.* [1] gave an encoding for labeled trees using $n\,(\lg\sigma + o(\lg\sigma))$ bits to support the retrieval of the ancestors or descendants by labels in $O(\lg\lg\sigma)$ time, which is generalized to represent multi-labeled trees in $t\,(\lg\sigma + o(\lg\sigma))$ bits. Barbay *et al.* [2] applied the same scheme using a succinct index for binary relations, and gave an encoding for multi-labeled trees which, in addition to the retrieval of the ancestors or descendants by labels, also supports the retrieval of the children by label, using one succinct index to support the string operators on the preorder traversal of the tree and another succinct index to support the string operators on the `DFUDS` traversal of the tree.

3 Multi-Labeled Graphs

Graphs have too many applications to list them here, and in many of them each node is associated to one or more labels.

Definition 2. *A* multi-labeled graph *is a graph where each node is associated to a set of labels from a finite alphabet.*

We do not consider simple labeled graphs as they are just a particular case of the multi-labeled case, where the binary relation can be encoded as a string. General graphs can have many edges, which makes them difficult to encode in small space, and makes the navigation operators difficult to support in efficient time. We consider instead several restrictions of graphs: graphs of bounded book-thickness, bipartite graphs and a oriented graphs of bounded degree.

3.1 Undirected Graphs of Bounded Book-Thickness

We first consider graphs of bounded book-thickness [4], i.e. which can be represented on k pages. Without loss of generality, we suppose that the ordering on the nodes of the book representation of the graph is the identity $(1, \ldots, n)$.

Jacobson's encoding supports the enumeration of the neighbors of a node [13, Section 4.2.2]. We consider instead some operators, implictly supported by Jacobson's encoding, which permit to describe at once the enumeration of all the nodes matching a request, and the extraction of the first node matching the request, given the nodes $x, y \in [n]$ and an integer $r \in [n]$:

- `neighbor_nb`(x), the number of neighbors of node x;
- `neighbor_rank`(x, y), the position of node y among the neighbors of node x;
- `neighbor_select`(x, r), the r-th node among the neighbors of node x;

We extend those operators to labeled trees for a given label or literal α to the *label-based operators on undirected graphs*:

- `lab_neighbor_nb`(α, x), the number of neighbors of node x matching literal α;
- `lab_neighbor_rank`(α, x, y), the position of node y among the neighbors of node x matching litteral α;
- `lab_neighbor_select`(α, x, r), the r-th node among the neighbors of node x associated with label α;
- `lab_neighbor_succ`(α, x, y) (resp. `lab_neighbor_pred`(α, x, y)), the first neighbor of node x matching literal α after (resp. before) node y;

For the sake of generality, we further add the following operators, in the context where the graph is not connected:

- `lab_connected_nb`(α, x), the number of nodes matching literal α in the same connected component as node x ;
- `lab_connected_rank`(α, x, y), the position of node y among the nodes matching literal α in the same connected component as node x;
- `lab_connected_select`(α, x, r), the r-th node associated with label α in the same connected component as node x;
- `lab_connected_succ`(α, x, y) (resp. `lab_connected_pred`(α, x, y)), the first node matching literal α in the connected component of x after (resp. before) node y;

All those operators can be supported by an encoding combining Jacobson's encoding for graphs and Barbay *et al.*'s succinct index for binary relations:

Theorem 1. *Consider a labeled graph of book-thickness k over n nodes, associated with σ labels in t pairs ($t > n$), so that it can be accessed in time $f(n, \sigma, t)$. One can build in time $O(ktf(n, \sigma, t))$ a succinct index using $kt \cdot o(\lg \sigma)$ bits which supports each label-based operators in time $O((\lg\lg\lg\sigma)^2 k(f(n, \sigma) + \lg\lg\sigma))$.*

Proof (draft). The proof is based on defining several orders on the nodes of the graph. For each order, we build a succinct index [2, Theorem 2] using $t \cdot o(\lg \sigma)$ bits that supports the operators `label_rank` and `label_access` in $O(\lg\lg\sigma \lg\lg\lg\sigma(f(n, \sigma, t) + \lg\lg\sigma))$ time, and `label_select` in $O(\lg\lg\lg\sigma(f(n, \sigma, t) + \lg\lg\sigma))$ time.

We first show how to support the label-based operators on the connected components. In linear time, assign each node to a connected component, and order the nodes so that all the nodes of the same connected component are consecutive. For each connected component, represent in unary[3] the number of nodes in this connected component minus one (by definition, there is no connected component of size zero). The concatenation of those representations forms a binary string of lenght n. Encoded in $n + o(n)$ bits using the encoding (a) from Lemma 1, it indicates for each node of the graph the range occupied in the order by its connexted component. Supporting the number, rank, select, predecessor and successor labeled operators on the connected component is then reduced to the corresponding operators in the binary relation.

We now show how to support the label-based operators on neighbors. By definition, there is an ordering on the nodes and a partition of the edges in k pages, such that on each page the edges do not cross [4]. As noted by Jacobson, each page can be considered as a forest of ordinal trees or, in combination with a binary relation, as a forest of multi-labeled trees. On each page, traverse each tree in `DFUDS` order, so that its children are consecutive, and build an index for the binary relation relative to this order of the nodes. Note that this order is possibly different from the book embedding and different from page to page. Hence we build one succinct index per page: this is fine for graphs of bounded book-thickness. Encode all the pages using any encoding for ordinal trees which supports the DFUDS rank and select operators [3, 14] and uses at most $2n$ bits. As the neighbors of a node x are its parent followed by its children, the label-based neighbor operators are then reduced to a combination of operators on the binary relation in the order defined on each page.

The data-structure can be computed in time linear in the number of pages and associations between nodes and labels, and the space it requires is dominated by the cost of the succinct indexes for the binary relation, $kt \cdot o(\lg \sigma)$ bits. Each label-based operator is supported by a finite number of operations on binary relations, hence the time of $O((\lg\lg\lg\sigma)^2 k(f(n, \sigma) + \lg\lg\sigma))$. □

As four pages are sufficient to represent planar graphs [19], the encoding is within a constant factor of the optimal for labeled planar graphs:

Corollary 1. *Consider a planar labeled graph over n nodes, associated with σ labels in t pairs ($t > n$), so that it can be accessed in time $f(n, \sigma, t)$. One can build in time $O(ktf(n, \sigma, t))$ a succinct index using $t \cdot o(\lg \sigma)$ bits which supports each label-based operators in time $O((\lg\lg\lg\sigma)^2 k(f(n, \sigma) + \lg\lg\sigma))$.*

[3] The unary encoding of x is the string composed of x zeroes followed by a one.

4 Conclusion

Barbay *et al.* [1] showed how ordinal trees, combined with binary relations, yields powerful labeled structures such as multi-labeled trees, through an adequately chosen order on the nodes of the tree. Barbay *et al.* [2] showed that one can use more than one orders on the nodes of the tree, which permits to support even more operators on multi-labeled trees. We extend those results to some classes of graphes, by combining the data-structures encoding various types of graphes with succinct indexes of a binary relation associating nodes and labels.

This demonstrates the importance of encodings for binary relations. Basic succinct data-structures such as binary strings or ordinal trees have been implemented and proved to be more efficient than pointer based representations [6, 8, 15]: a practical implementation of binary relations will trigger the use of succinct encodings in many practical applications were labeled structures are required.

Bibliography

[1] J. Barbay, A. Golynski, J. I. Munro, and S. S. Rao. Adaptive searching in succinctly encoded binary relations and tree-structured documents. In *Proceedings of the 17th Annual Symposium on Combinatorial Pattern Matching*, pages 24–35. Springer-Verlag LNCS 4009, 2006.

[2] J. Barbay, M. He, J. I. Munro, and S. S. Rao. Succinct indexes for strings, binary relations and multi-labeled trees. In *Proceedings of the 18th ACM-SIAM Symposium On Discrete Algorithms (to appear)*, 2007.

[3] D. Benoit, E. D. Demaine, J. I. Munro, R. Raman, V. Raman, and S. S. Rao. Representing trees of higher degree. *Algorithmica*, 43(4):275–292, 2005.

[4] F. Bernhart and P. Kainen. The book thickness of a graph. *Journal of Combinatorial Theory*, B(27):320–331, 1979.

[5] D. R. Clark and J. I. Munro. Efficient suffix trees on secondary storage. In *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 383–391, 1996.

[6] O. Delpratt, N. Rahman, and R. Raman. Engineering the louds succinct tree representation. In C. Àlvarez and M. J. Serna, editors, *WEA*, volume 4007 of *Lecture Notes in Computer Science*, pages 134–145. Springer, 2006.

[7] P. Ferragina, F. Luccio, G. Manzini, and S. Muthukrishnan. Structuring labeled trees for optimal succinctness, and beyond. In *Proceedings of the 46th IEEE Symposium on Foundations of Computer Science*, pages 184–196, 2005.

[8] R. F. Geary, N. Rahman, R. Raman, and V. Raman. A simple optimal representation for balanced parentheses. In S. C. Sahinalp, S. Muthukrishnan, and U. Dogrusöz, editors, *CPM*, volume 3109 of *Lecture Notes in Computer Science*, pages 159–172. Springer, 2004.

[9] R. F. Geary, R. Raman, and V. Raman. Succinct ordinal trees with level-ancestor queries. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–10, 2004.

[10] A. Golynski. Optimal lower bounds for rank and select indexes. In *Proceedings of the International Colloquium on Automata, Languages and Programming*, 2006.

[11] A. Golynski, J. I. Munro, and S. S. Rao. Rank/select operations on large alphabets: a tool for text indexing. In *Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 368–373, 2006.

[12] R. Grossi, A. Gupta, and J. S. Vitter. High-order entropy-compressed text indexes. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 841–850, 2003.

[13] G. Jacobson. Space-efficient static trees and graphs. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 549–554, 1989.

[14] J. Jansson, K. Sadakane, and W.-K. Sung. Ultra-succinct representation of ordered trees. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

[15] G. Manzini and P. Ferragina. Engineering a lightweight suffix array construction algorithm. *Algorithmica*, 40(1):33–50, 2004.

[16] J. I. Munro and V. Raman. Succinct representation of balanced parentheses and static trees. *SIAM Journal on Computing*, 31(3):762–776, 2001.

[17] M. Naor. Succinct representation of general unlabeled graphs. *Discrete Appl. Math.*, 28(3):303–307, 1990.

[18] R. Raman, V. Raman, and S. S. Rao. Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 233–242, 2002.

[19] M. Yannakakis. Four pages are necessary and sufficient for planar graphs. In *Proeedings of the 18th ACM Symposium on Theory of Computing*, pages 104–108, 1986.