EuroComb 2005 DMTCS proc. **AE**, 2005, 181–186

Pebble Game Algorithms and (k, l)-Sparse Graphs

Audrey Lee[1 †] and Ileana Streinu[2 ‡]

[1] *Department of Computer Science, University of Massachusetts at Amherst, Amherst, MA, USA. alee@cs.umass.edu*
[2] *Computer Science Department, Smith College, Northampton, MA, USA. streinu@cs.smith.edu*

A multi-graph G on n vertices is (k, l)-sparse if every subset of $n' \leq n$ vertices spans at most $kn' - l$ edges, $0 \leq l < 2k$. G is *tight* if, in addition, it has exactly $kn - l$ edges. We characterize (k, l)-sparse graphs via a family of simple, elegant and efficient algorithms called the (k, l)-pebble games.

As applications, we use the pebble games for computing *components* (maximal tight subgraphs) in sparse graphs, to obtain inductive (Henneberg) constructions, and, when $l = k$, edge-disjoint tree decompositions.

Keywords: sparse graph, pebble game, rigidity, arboricity, graph orientation with bounded degree

1 Introduction

A graph[§] $G = (V, E)$ with $n = |V|$ vertices and $m = |E|$ edges is (k, l)-*sparse* if every subset of $n' \leq n$ vertices spans at most $kn' - l$ edges. If, furthermore, $m = kn - l$, G will be called *tight*. A graph is a (k, a)-arborescence if it can be decomposed into k edge-disjoint spanning trees after the addition of *any* a edges. Here, k, l and a are positive integers satisfying a simple relationship, which is $0 \leq l < 2k$ for sparseness or $0 \leq a < k$ for arboricity.

Sparseness and arboricity are closely related, and have important applications in Rigidity Theory. Classical results of Nash-Williams [10] and Tutte [16] identify the class of graphs decomposable into k edge-disjoint spanning trees with the tight (k, k)-sparse graphs. A theorem of Tay [14, 15] relates them to body-and-bar rigidity in arbitrary dimensions. The tight $(2, 3)$-sparse graphs also play an important role in Rigidity Theory: they are the *generically minimally rigid graphs* (also known as Laman graphs [7]) in dimension 2. Further results of Recski [11, 12] and Lovasz and Yemini [9] relate them to $(2, 1)$-arborescences. Haas [4] proves the equivalence between $(k, l - k)$-arborescences and (k, l)-sparse graphs for $k \leq l < 2k$.

Whiteley [17, 18] identifies rigidity applications for (k, l)-sparse graphs which do not correspond to arborescences, namely for $0 \leq l < k$, and shows that they form the set of bases of a matroid. More

† Supported by an NSF graduate fellowship and NSF grant CCR-0310661.
‡ Supported by NSF grants CCR-0310661 and CCF-0430990.
§ Throughout the paper we will use the abbreviation *graph*, even though they may have multiple (parallel) edges or even loops.

recently, Frank, Szegő and Fekete [3, 13, 2] study inductive constructions for various (sub)classes of sparse graphs, motivated by the so-called Henneberg sequences appearing in Rigidity Theory for Laman graphs and other classes of rigid structures.

From the algorithmic point of view, the literature is equally rich. There exist many algorithms for decomposing a graph into edge-disjoint trees or forests (some generalizing to and presented in the context of matroid unions), or for recognizing Laman graphs. Related topics include packing and covering with trees. However, we have not found algorithms to specifically recognize (k, a)-arborescences or, more generally, the (k, l)-sparse graphs, in an efficient manner.

In this paper, we describe a family of algorithms, called the (k, l)-pebble games, and prove that they recognize *exactly* the (k, l)-sparse graphs, $0 \leq l < 2k$.

Our starting point is the simple and elegant *pebble game algorithm* introduced by Jacobs and Hendrickson [5] for 2-dimensional rigidity and further analyzed by Berg and Jordán [1]. In our terminology, theirs is a $(2, 3)$-pebble game. Its simplicity makes it the tool of choice for practical applications in studies of protein flexibility by Jacobs et al. [6], who also proposed various unproven pebble game heuristics for handling (certain special cases of) three-dimensional rigidity.

Here we exhibit for the first time the full extent to which the basic pebble game of Jacobs and Hendrickson can be generalized. The range of parameter values $0 \leq l < 2k$ appears naturally as a constraint for the pebble games on (multi)-graphs. Our result relates them directly to those (k, l)-sparseness conditions which are *matroidal*, as observed by White and Whiteley in the appendix of [18].

2 (k, l)-Pebble Games

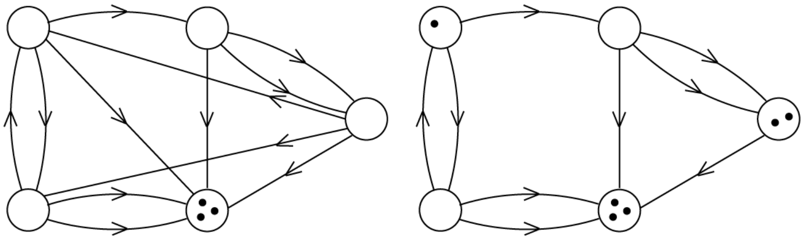
A pebble game algorithm depends on two constants: k, the *initial number of pebbles* on each vertex, and l, the *acceptance condition* (the number of pebbles on the endpoints of an edge after it has been accepted).

The algorithm takes a graph as input and classifies it according to three categories. The first two, *Well-constrained* and *Under-constrained*, correspond to *success* in the game, and the third one to *Failure* to recognize the input graph.

Algorithm 1 The (k, l)-Pebble Game.
Input: *A graph $G = (V, E)$, possibly with loops and multiple edges.*
Output: Well-constrained*,* Under-constrained *or* Failure*.*

Setup: *Maintain, as an additional data structure, a directed graph G', on which the game is played. Initialize G' to be the empty graph on V, and place k pebbles on each vertex.*
Rules: *Throughout the algorithm, no more than k pebbles may be present on a vertex.*
Allowable moves: *An additional pebble can be collected on a vertex v by searching the partially constructed graph G', e.g., via depth-first search. If a pebble is found, the directed path leading to it is reversed and the pebble is moved along the path until it reaches v.*
Algorithm: *Consider the edges of E in an arbitrary order; each edge will either be discarded or accepted according to the following* acceptance condition: *the presence of* at least $l + 1$ *pebbles on its endpoints. Every accepted edge is subsequently inserted into G' as a directed edge; a pebble is consumed from one of its endpoints, and the edge is oriented away from that endpoint.*

The game ends when all edges have been processed. If any edge was rejected, the output is **Failure***. Otherwise, at least l pebbles always remain at the end. The output is* **Well-constrained** *if exactly l pebbles remain, or* **Under-constrained** *otherwise.* □



(a) A well-constrained $(3, 3)$-pebble game output, with the final orientation and distribution of pebbles on the input graph.

(b) An under-constrained $(3, 3)$-pebble game output: the input graph lacks three edges to be a complete $(3, 0)$-arborescence.

Fig. 1: Successful final state of the $(3, 3)$-pebble game on two graphs.

We will prove that *well-constrained* pebble-game graphs coincide with *tight* (k, l)*-sparse*, while *under-constrained* ones are sparse, but not tight. The failure of the pebble game indicates that the graph does not satisfy the (k, l)-sparseness condition. See Figure 1 for an example.

3 Blocks and Components in (k, l)-sparse graphs

In a (k, l)-sparse graph, a subset of vertices $V' \subset V$ may span *exactly* $kn' - l$ edges. In this case, the induced subgraph is called a *block*. A maximal block is called a *component*. The blocks represent the obstructions to adding further edges in a sparse graph. We first characterize the way they interact with one another. Blocks may be vertex-disjoint, intersect in only one vertex or have a larger intersection.

Lemma 1 Intersection of blocks. *When two blocks intersect in at least two vertices, or when they intersect in one vertex and* $0 \leq l \leq k$*, the intersection induces a block.*

As a corollary, it follows that the components are *edge-disjoint* (but not always vertex-disjoint). If the edges of the components are removed from a sparse graph, some extra edges (called *free edges*) may remain.

Lemma 2 Decomposition into Components, Free Vertices and Free Edges. *For* $0 \leq l \leq k$*, the components are vertex disjoint, may not be connected, and a single vertex with* $k - l$ *loops is a block. For* $l = k$*, a single vertex is always a block, but not necessarily otherwise. In this case, the vertex set is partitioned into components (possibly connected by free edges). For* $k \leq l < 2k - 1$*, there may be free edges, but for* $l = 2k - 1$*, there are none (the smallest component is a single edge). In this case, the edge set is partitioned into components.*

4 Pebble Game Graphs coincide with (k, l)-Sparse Graphs

The main result can now be formulated:

Theorem 3 *The class of* (k, l)*-sparse graphs coincides with the class of under-constrained* (k, l)*-pebble game graphs, and tight ones correspond to well-constrained.*

The proof follows from the following sequence of Lemmas. Let $peb(v)$ be the number of free pebbles on a vertex v, $span(v)$ be the number of loops and $out(v)$ its out-degree (edges with a second different endpoint than v). Extend to vertex sets in a natural way: for $V' \subset V$, $peb(V') = \sum_{v \in V'} peb(v)$, $span(V')$ is the number of edges spanned by V' (including loops) and $out(V')$ is the number of edges between V' and its complement $V - V'$.

Lemma 4 Invariants of the Pebble Game. *At any time during the pebble game, the following invariants are maintained:*

1. *For every vertex v, $peb(v) + span(v) + out(v) = k$.*
2. *There are at least l free pebbles in the graph.*
3. *For any subset $V' \subset V$ on n' vertices and $m' = span(V')$ edges, $peb(V') + span(V') + out(V') = kn'$. This implies that $m' \leq kn' - l$.*
4. *If V' spans a block, $peb(V') + out(V') = l$. If V' is under-constrained, $peb(V') + out(V') > l$.*

Denote by $Reach(v)$ the *reachability region* of a vertex v: the set of vertices that can be reached via directed paths from v.

Lemma 5 Edge insertion. *An edge (u, v) can be successfully inserted during the pebble game iff its endpoints do not belong to a block of the current graph.*

The proof works by showing inductively that the pebbles can be collected one by one on the endpoints of the vertex. After each collection, the reachability region of a vertex may change. It is instructive to notice that it doesn't suffice to require that l pebbles be present in the reachability regions of u and v, see Fig. 2.

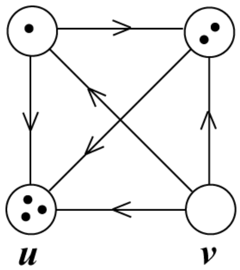


Fig. 2: A $(3, 5)$-pebble game where the reachability region for the pair u and v contains 6 pebbles but they cannot be all collected on the two vertices u and v. No edge parallel to uv can be inserted.

Theorem 3 follows then by induction: in one direction on the number of inserted edges, in the other, on the number of edges in a sparse graph.

5 Further results and Applications

The basic pebble game paradigm presented in this abstract is for the *Recognition Problem* of (k, l)-sparse graphs. The game can be extended to compute *Components* in sparse graphs (which, in rigidity applications, correspond to *rigid components* of under-constrained graphs). We also consider the *Extraction Problem*, where a maximal (k, l)-sparse graph is extracted from a denser graph, and the *Optimization Problem*, where a minimum cost maximal sparse graph is computed.

The basic pebble game presented here for the recognition problem runs in time $O(n^2)$, but when we want to extract a sparse graph from a dense graph (possibly with $m = O(n^2)$ edges), it is $O(nm)$. To obtain an improved $O(n^2)$ running time for extraction and optimization, we maintain *components*. Indeed, this allows for easy rejection of edges when their endpoints fall inside an existing component. To this end, the algorithm relies on additional (efficient but elementary) data structures. This falls outside the scope of what we do here and is described in a companion paper of Lee, Streinu and Theran [8].

A simple extension leads to an $O(n^2)$ time algorithm for producing a Henneberg sequence for Laman graphs, and all the (k, l)-sparse graphs with $0 \leq l < \frac{3}{2}k$.

The pebble game algorithm works without producing or maintaining any tree/forest decompositions along the way. For the special case of graphs which are the edge-disjoint union of k spanning trees ($(k, 0)$-arborescences), we further produce the tree decomposition efficiently from the Henneberg sequence.

Finally, we note that the outcome of the pebble game algorithm is an orientation of the input graph with certain very uniform out-degree constraints (in particular, the out-degree of every vertex is at most k). By moving the pebbles around we can generate all such orientations compatible with the given sparse graph.

References

[1] A. Berg and T. Jordán. Algorithms for graph rigidity and scene analysis. In G. D. Battista and U. Zwick, editors, *Proc. 11th Annual European Symposium on Algorithms (ESA)*, volume 2832 of *Lecture Notes in Computer Science*, pages 78–89. Springer, 2003.

[2] Z. Fekete and L. Szegő. A note on $[k, l]$-sparse graphs. TR 2005-05, Egerváry Research Group, Eötvös University, Budapest, 2005.

[3] A. Frank and L. Szegő. Constructive characterizations on packing and covering by trees. *Discrete Applied Mathematics*, 131:347–371, 2003.

[4] R. Haas. Characterizations of arboricity of graphs. *Ars Combinatorica*, 63:129–137, 2002.

[5] D. J. Jacobs and B. Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. *Journal of Computational Physics*, 137:346 – 365, November 1997.

[6] D. J. Jacobs, A. Rader, M. Thorpe, and L. A. Kuhn. Protein flexibilty predictions using graph theory. *Proteins 44*, pages 150 – 165, 2001.

[7] G. Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:331–340, 1970.

[8] A. Lee, I. Streinu, and L. Theran. Finding and maintaining rigid components. *Manuscript, submitted*, 2005.

[9] L. Lovasz and Y. Yemini. On generic rigidity in the plane. *SIAM J. Algebraic and Discrete Methods*, 3(1):91–98, 1982.

[10] C. S. A. Nash-Williams. Edge-disjoint spanning trees of finite graphs. *Journal London Math. Soc.*, 36:445–450, 1961.

[11] A. Recski. A network theory approach to the rigidity of skeletal structures I. Modelling and interconnection. *Discrete Applied Mathematics*, 7:313–324, 1984.

[12] A. Recski. A network theory approach to the rigidity of skeletal structures II. Laman's theorem and topological formulae. *Discrete Applied Mathematics*, 8:63–68, 1984.

[13] L. Szegő. On constructive characterizations of (k, l)-sparse graphs. TR 2003-10, Egerváry Research Group, Eötvös University, Budapest, 2003. EuroComb 2003, Prague.

[14] T.-S. Tay. Rigidity of multi-graphs i. linking rigid bodies in n-space. *Journal of Combinatorial Theory Series B*, 36:95–112, 1984.

[15] T.-S. Tay. Linking $(n-2)-$dimensional panels in n-space ii: $(n-2,2)-$franeworks and body and hinge structures. *Graphs and Combinatorics*, 5:245–273, 1989.

[16] W. T. Tutte. On the problem of decomposing a graph into n connected factors. *Journal London Math. Soc.*, 142:221–230, 1961.

[17] W. Whiteley. The union of matroids and the rigidity of frameworks. *SIAM Journal Discrete Mathematics*, 1(2):237–255, May 1988.

[18] W. Whiteley. Some matroids from discrete applied geometry. In J. O. J. Bonin and B. Servatius, editors, *Matroid Theory*, volume 197 of *Contemporary Mathematics*, pages 171–311. American Mathematical Society, 1996.