Time-Space Lower Bounds for Directed s-t Connectivity on JAG Models

Greg Barnes* Jeff A. Edmonds †

January 8, 1996

1 Stack JAGs and Other JAG models

Although JAGs are a powerful structured model for s-t connectivity, they are not a general model and there are complaints about the model. Hence, a goal is to define a stronger model on which a lower bound can be proved. However, even defining an intermediate model between the NNJAG model and the branching program seems hard. For example, allowing the model to move a pebble to an arbitrary node or to the next node in some fixed ordering would give the power of branching programs. Within a polynomial factor in time, so does allowing it to move pebbles backwards along any directed edge [?]. This section defines a model that addresses some of these issues. It is called a Stack JAG and it was simultaniously defined in [?]. Their bounds were proved on the Stack NNJAG. Here we define a provably stronger version of this stack property than that used by them. The proofs in this paper easily go through with this stronger model.

The standard JAG model is not allowed to move a pebble to an arbitrary node. In fact, it can only access vertices that have been walked to from s or from t and vertices that are disjoint from s and t can never be reached. On the other hand, adding this strength makes the model the same as a branching program simply by the definitions of the models. Alternatively, one could allow the JAG to *strong jump*. Here the vertices are put in some arbitrary order and a pebble is allowed to jump from its current vertex to the next vertex in the order. Clearly this adds at most a factor of n onto the time of a branching program. Moreover, Savitch [?] shows that such a machine is equivalent to a Turing machine with no blow up in time for the purposes of solving s-t connectivity.

The difficulty in proving lower bounds on a general model of computation is that of defining a good measure of progress. When the computation problem requires many outputs,

*Department of Computer Science, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. E-mail address: gsbarnes@plg.uwaterloo.ca Portions of this work were performed while the author was at the Max-Planck-Institut für Informatik, Saarbrücken, Germany.

†International Computer Science Institute, 1947 Center Street, Suite 600, Berkeley, California 94704-1198. E-mail address: edmonds@icsi.berkeley.edu; Home Page: http://www.icsi.berkeley.edu/~edmonds. Portions of this work were performed while the author was at the University of Toronto, Canada.

the measure can be the number of values outputted. However, when the output is a single bit, it is harder to say for each point in the computation, how much "progress" the algorithm has made towards deciding this one bit. The weakness of the JAG of not being able to quickly access all of the nodes provides us with the required measure of progress. Progress is simply the number of "hard" nodes that have been accessed.

Our intuition into why randomly accessing the vertices is not useful is that if the algorithm bounces around the graph and collects local information about the neborhood of each node, then the model will not have enough space to store enough useful information and will be unable to put this information together to decide STCON.

As evidence that having random access to the vertices might be useful, there are two algorithms for undirected s-t connectivity, Barnes and Ruzzo's [**?**] and Nisan *et al.*'s [**?**], that seem to require such power. Both algorithms repeatedly find sets of connected vertices and "coalesce" each set into a single representative. On the other hand, it seems likely that the coalescing scheme used by these two algorithms is only useful in undirected graphs, where "connected to" is an equivalence relation. Therefore, it is not clear that accessing all vertices would be useful when implementing JAG algorithms for (directed) STCON.

The standard JAG model also does not allow a pebble to travel backward along a directed edge. In fact, the JAG has no easy access to the indegree of a node. On the other hand, on a general model, even if the input representation of the directed graph lists for each node the out edges but not the in edges, such operations could be done in linear time. Therefore a reasonable strengthening of the JAG model would be to allow it to specify an incoming edge via a labeling of these edges and then to move backwards along the edge.

Not being able to traverse backwards along an edge provably does weaken the model. The $\Omega(\log^2 n / \log\log n)$ space bound for STCON [**?**,**?**,**?**] exploits just this fact. They show that in their graphs (which are trees) pebbles get "stuck" in dead end paths and hence have to jump a considerable distance back to be able to restart again. The fact that the model is too weak is demonstrated by the fact that a general model can solve STCON on trees with only $O(\log n)$ space.

The stack version of the JAG was defined in response to this complaint by allowing the model to push onto a stack the nodes traversed by a pebble and to back up along this traversed path by popping the stack. Clearly this model can traverse trees with only a constant number of pebbles, i.e. $O(\log n)$ space.

Before discussing the Stack JAG further, let us consider whether the model could be strengthened further. What the Stack JAG is unable to do is to traverse forward down some path and then to traverse backwards along edges up a different path. Our intuition is that such an operation is not useful for determining whether a strictly forward directed path exists between s and t. (See Section **??** for a counter example to this intuition). At first our intuition also told us that proving such a lower bound could on such a model could not be that much harder. However, this second intuition is false. Being able to do this would

give the JAG (within a polynomial factor in time) the power of the branching program [**?**]. This is proved using a variation on Theorem 7 in Beame *et al.* [**?**]. The idea is that by moving backwards along edges, one can treat the graph as an undirected graph and, using a universal traversal sequence [**?**], visit any vertex in polynomial time. Like the strong jump operation, this implies the JAG can then simulate a branching program. Because of the polynomial blow up in time needed by the transformation, a $ST = n^2/\log n$ lower bound in a backward moving JAG would say nothing about a general model of computation. Hence, this is an interesting open problem. The problem is similar to (but maybe slightly easier than) proving this trade off for undirected STCON. (Again see Section **??**).

Now let us consider the **Stack JAG**. In [**?**], it was define as follows. The Stack NNJAG is given the additional power of a number of **stack pebbles**. Such a pebble is defined to have a **stack** of nodes associated with it. Whenever the pebble walks along a directed edge, the node is *pushed* onto its stack. The model is also allowed to *pop* the pebble's stack, which has the effect of moving the pebble backward along the path that it took getting there. (This does not allow a pebble to move backwards along arbitrary directed edges but it is a good start.) When a stack pebble jumps to the location of another stack pebble his stack is replaced with a copy of the other pebble's stack. When it jumps to a non-stack pebble, its stack is emptied. The model is not charged for the space used by the stacks. The paper [**?**] considers both the case in which there are only a constant number of stack pebbles and the case in which all the pebbles are. They prove their bounds on such a probabilistic Stack NNJAG.

Let us consider why the model does not charged for the space used by the stacks. For upper bounds, it would be fair to charge $\log n$ bits of space for each node on a stack. However, if this is done the Stack JAG is no stronger than a regular JAG. The regular JAG can simulate the stacks by dropping, jumping to, and picking up pebbles. Hence, the Stack model only becomes interesting when no space is charged for the Stacks. Since we are doing lower bounds, defining such a powerful model only makes the bounds stronger.

This paper generalizes the Stack model only slightly. However, this difference turns out to be significant. Let us call it a **Stack$'$ JAG**. In this model, a stack pebble is not required to push a node when it walks and edge nor is it required to copy the other pebbles stack when it jumps. Instead, it does these things only when it likes.

It is not difficult to see that the proofs of both Theorems **??** and **??** carry over easily to Stack$'$ JAGs. The proofs only charge a time step each time a pebble enters a tree (or tooth) and for reaching the bottom of a path. They do not charge for the pebbles getting back up. They also give away for free the information as to whether or not two pebbles are contained in the same tree (tooth). Hence, the following two Corollaries are true.

Corollary 1 *Any Stack$'$ JAG that solves* STCON *on graphs with n vertices using p pebbles requires time* $\Omega\left(\frac{n^2}{p\log^2(n/p)}\right)$.

Corollary 2 *Any Stack′ JAG that solves* STCON *on graphs with n vertices and m edges using p pebbles requires time $\Omega(mn^{\frac{1}{2}}/p^{\frac{1}{2}})$.*

Let us compare the Stack′ JAG and the Stack NNJAG and the corresponding lower bound techniques. The Stack NNJAG bound [**?**] points out that the maximum number of nodes that a stack pebble can push onto its stack is the length of the longest directed path in the input graph. Hence they bound this length and in some cases limit the number of stack pebbles to be a constant, there by bounding the amount of space that is given away in the stacks for free to either $O(S)$ or $O(S^2)$. Then they observe that increasing the space in their NNJAG lower bound by this extra amount does not change the asymptotics of the result.

In contrast, a Stack′ NNJAG is much too powerful of a model. In fact, it can solve (directed) STCON with $O(n^2)$ time and $O(\log n)$ space. The algorithm used is depth first search. A constant number of pebbles are able to traverse a spanning tree of the graph in linear time in the usual way by pushing and popping onto the stack the vertices along the path to the root. All that is required is to prevent the algorithm from cycling by remembering which nodes have been seen before. However, this requires n bits of "space".

The Stack′ NNJAG can "store" a vector $\vec{l} \in \{0,1\}^*$ by choosing two nodes v_0 and v_1 and pushing the sequence of these two nodes onto the stack in the order specified by the bits of $\vec{l}$. The ith bit of $\vec{l}$ can be recalled and modified by popping the nodes from one stack while pushing them onto another, stopping at the required bit. This requires only $O(\log n)$ states to count the number of nodes pushed and popped and linear time per memory access.

The bounds in this paper hold for the Stack′ JAG, even when there is no limit on the number of nodes pushed onto the Stack. The main reason that the lower bound adversary was able to not charge space for the stacks is that she already only charges for the pebbles. Recall that the states are free. The reason that she can do this is that the proofs assume that the JAG remembers everything that it has ever known. The key is that a JAG only can learn for each time step which pebbles are on the same node. In Theorem **??**, the JAG learns nothing since pebbles NEVER meet unless they have traversed exactly the same path. In Theorem **??**, the JAG learns as little as possible. This is done by the adversary playing the partition game with the JAG to decide when the pebbles should meet and when they should not.

2 Undirected S-T Graph Connectivity

Like (directed) STCON, undirected graph s-t connectivity (USTCON) is of interest. Time space tradeoffs as high as those found in [**?**] cannot be obtained here, because it can be solved probabilistically with a trade off of $S \cdot T \in m^{1.5}n^{.5}\log^{O(1)} n$ [**?**,**?**]. (Even a deterministic JAG can use its non-uniformity to produce a universal traversal sequence [**?**].)

The current JAG lower bound [**?**] for USTCON allows only $O\left(\frac{\log n}{\log\log n}\right)$ pebbles and only gives $T \in n \times 2^{\Omega\frac{\log n}{\log\log n}}$. The journal version of the paper presents a fun open problem, a solution to which would give the desired lower bound of $n^{2-o(1)}$ for USTCON with this number of pebbles. However, the problem seems to be hard. It is unlikely that this technique will be able to prove a lower bound with the number of pebbles increased beyond $\log n$, because a pebble is required for each depth of recursion of the graph and a graph with n nodes cannot be constructed with more than $\log_2 n$ levels of recursion. As said, the current paper is the first to prove a bound with an arbitrary number of jumping pebbles.

Another direction to pursue is to try to prove a lower bound for USTCON on a graph that is similar to the comb graph but with undirected edges. There are a number of problems with trying to extend the proof techniques that are used in this paper to undirected graphs. The first issue is that on an undirected version of the comb graph, pebbles would be able to trivially travel from t backwards to s. This problem can be solved by have in two identical copies of the graph connected at the t nodes. The new question is whether s and s' are connected. The advantage of this is that all pebbles then are placed on one of the s nodes. Another problem is that the JAG can differentiate between the teeth by learning the degrees of their top nodes. However, a comb graph could be given for which these degrees are all the same. This would require a lower bound on the partition game when the input is guaranteed to be a partition of the m connecting edges into χ parts of equal sides. This is doable. A third problem arises from the labels on the edges from the tops of the teeth to the back nodes. In fact, the following is a 3 pebble, linear time deterministic algorithm for undirected comb graphs.

```
For each back node v_i
  move two pebbles P_1 and P_2 to v_i
  move P_1 into tooth connected to v_i
  move P_1 back up to the back nodes through the edge labeled 1.
  If P_1 finds P_2 then
    we know that the edged connected to v_i has label 1
    move P_1 back into the tooth,
    down to the bottom,
    look for node t
  end if
end for
```

Every tooth is connected to some back node v_i via the edge labeled 1. Therefore, every tooth will be traversed once and only once. Actually the same algorithm works for any layered graph.

The following kluge fixes the above problem and gives a $TS^{1/2} > n^{3/2}$ time space trade off for JAGs on undirected multi-graphs of high degree. Though the proof uses the formal definition of a JAG, it unfairly goes against the spirit of the definition. It depends heavily on having multiple edges between nodes and uses a funny quark of the model. Specifically suppose there are two edges from u to v and from both u and v they are labeled 1 and 2. Then there is no way of the model knowing whether one edge is labeled 1 on both ends or one is labeled 1 2 and the other 2 1. The way we get around the above algorithm is to not have the back of the comb. The graph has an s note and χ teeth. There are m/χ edges from s to the top of each of the tooth. The edges up from the top of the tooth are labeled $[1..m/\chi]$. If a pebble traverses up from the top of a tooth it learns nothing. Why? Because it always arrives at node s.

Hi Greg, I hope that you are doing fine.

You were conserned about the stack proof. You could well be right. But I dont yet see any faults in it. I thought of yet another model that is stronger than the stack model.

The JAG+ model: - JAG - arbitary number of states. - Each move it specifies a pebble and an arbitrary long sequence of edge labeles. The pebble is moved in one step to the node at the end of the sequence.

This is stronger than any STACK JAG. The nodes have no names. Hence the stack stores only a list of time steps. Popping the stack moves the pebble to the node that it was on at the time step popped.

The JAG+ model with it arbitary many states can remember for each pebble and for time step the path from s to the node the pebble was at on this time step. It can also remeber any stack of time steps. It can also move a pebble to such a time step by jumping it to s and moving it down the sequence.

I think the JAG+ model is simple enough that it can be defined at the beginning where the JAG is defined. Then the STACK JAG can simply be mensioned in reference to the EdmPoon paper.

What do you think?

Jeff

3 Open Problems

The obvious open problems are to decrease the gap between the upper and the lower bounds for STCON on the Stack' JAG and the Stack NNJAG models and to prove any bound on a stronger model.

When the space is $S = n^{1-\epsilon}$, the gap is given by $T = 2^{O(\log^2 n)}$ [**?**] vs $T = 2^{\Omega(\frac{\log^2 n}{\log\log n})}$ [**?**]. The $\log\log n$ factor could be gotten rid of. When the space is $S = O(n \log n)$, the bounds

are tight, i.e. $T = O(m)$ for depth first search and $S^{\frac{1}{2}}T = \Omega(m(n \log n)^{\frac{1}{2}})$ given here. When the space is just slightly sublinear there is again a gap, probably caused by the mn in the $T = 2^{O(\log^2(n/S))}{}_{\times mn}$ upper bound. This could be tightened as well.

Ultimately, one would like to prove lower bounds for STCON like those in this paper and like Edmonds and Poon's on a general model of computation. Any nontrivial bounds for general models would be a step in this direction. As argued in Section **??**, a more modest goal would be to add features to the JAG or NNJAG to make it more general (as Poon added node names to the JAG to devise the NNJAG [**?**]) and to prove the same bounds on these more general models. Section **??** mentioned two areas in which the JAG seems limited compared to a general model, and showed that some obvious solutions to these problems, such as strong jumps or the ability to move backwards along edges, make a JAG as strong as a Turing machine. Can one devise a weaker version of these operations and prove lower bounds on a JAG with such an operation? In particular, can one prove a bound similar to Edmonds and Poon's on the Stack JAG, and can one devise a weaker operation than strong jumping that allows one to implement the algorithms of Barnes and Ruzzo [**?**] and Nisan *et al.* [**?**] on a JAG?

Acknowledgments

We would like to thank Faith Fich for her extensive support, and Paul Beame, Al Borodin, Russell Impagliazzo and Hisao Tamaki for their helpful comments and suggestions.

An earlier version of some of this work appears in Edmonds' PhD thesis [**?**].

References

[AKLLR70] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. W. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *20th Annual Symposium on Foundations of Computer Science*, pages 218–223, San Juan, Puerto Rico, Oct. 1979. IEEE.

[BBRS92] G. Barnes, J. F. Buss, W. L. Ruzzo, and B. Schieber. A sublinear space, polynomial time algorithm for directed *s-t* connectivity. In *Proceedings, Structure in Complexity Theory, Seventh Annual Conference*, pages 27–33, Boston, MA, June 1992. IEEE.

[BBRS95] G. Barnes, J. F. Buss, W. L. Ruzzo, and B. Schieber. A sublinear space, polynomial time algorithm for directed *s-t* connectivity. To appear in *SIAM Journal on Computing*, 1995. Preliminary version [**?**].

[BE93] G. Barnes and J. A. Edmonds. Time-space lower bounds for directed *s-t* connectivity on JAG models. In *Proceedings 34th Annual Symposium on Foundations of Computer Science*, pages 228–237, Palo Alto, CA, Nov. 1993. IEEE.

[BF93] Greg Barnes, Uriel Feige. Short random walks on graphs. In *Proceedings of the Twenty Fifth Annual ACM Symposium on Theory of Computing*, pages 728–737, San Diego, CA, May 1993.

[BR91] G. Barnes and W. L. Ruzzo. Deterministic algorithms for undirected *s-t* connectivity using polynomial time and sublinear space. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, pages 43–53, New Orleans, LA, May 1991. Also Department of Computer Science and Engineering, University of Washington Technical Report 91-06-02.

[BR95] G. Barnes and W. L. Ruzzo. Undirected *s-t* connectivity in polynomial time and sublinear space. *Computational Complexity*, 1995. To appear. Preliminary version [**?**].

[BBRRT90] P. W. Beame, A. Borodin, P. Raghavan, W. L. Ruzzo, and M. Tompa. Time-space tradeoffs for undirected graph connectivity. In *Proceedings 31st Annual Symposium on Foundations of Computer Science*, pages 429–438, St. Louis, MO, Oct. 1990. IEEE. Full version: University of Washington Department of Computer Science and Engineering Technical Report 93–02–01, 44 pages; submitted for publication.

[BS83] P. Berman and J. Simon. Lower bounds on graph threading by probabilistic machines. In *24th Annual Symposium on Foundations of Computer Science*, pages 304–311, Tucson, AZ, Nov. 1983. IEEE.

[Bor82] A. Borodin. Structured *vs.* general models in computational complexity. *L'Enseignement Mathématique*, XXVIII(3-4):171–190, July-Dec. 1982. Also in [**?**, pages 47–65].

[BRT92] A. Borodin, W. L. Ruzzo, and M. Tompa. Lower bounds on the length of universal traversal sequences. *Journal of Computer and System Sciences*, 45(2):180–203, Oct. 1992.

[BKRU89] A. Z. Broder, A. R. Karlin, P. Raghavan, and E. Upfal. Trading space for time in undirected *s-t* connectivity. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 543–549, Seattle, WA, May 1989.

[CR80] S. A. Cook and C. W. Rackoff. Space lower bounds for maze threadability on restricted machines. *SIAM Journal on Computing*, 9(3):636–652, Aug. 1980.

[Edm93a] J. A. Edmonds. *Time-Space Lower Bounds for Undirected and Directed ST-Connectivity on JAG Models.* PhD thesis, University of Toronto, Aug. 1993.

[Edm93b] J. A. Edmonds. Time-space trade-offs for undirected ST-connectivity on a JAG. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 718–727, San Diego, CA, May 1993. Submitted to the *SIAM Journal on Computing.*

[EP95] J. A. Edmonds and C. K. Poon. Super polynomial time-space lower bounds for directed st-connectivity on the NNJAG model. To appear in *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing*, Las Vegas, NV, May 1995. Full version available in Home Page: http://www.icsi.berkeley.edu/~edmonds. To be submitted to the *SIAM Journal on Computing.*

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17(5):935–938, Oct. 1988.

[LP82] H. R. Lewis and C. H. Papadimitriou. Symmetric space-bounded computation. *Theoretical Computer Science*, 19(2):161–187, Aug. 1982.

[Spec82] *Logic and Algorithmic,* An International Symposium Held in Honor of Ernst Specker, Zürich, Feb. 5–11, 1980. Monographie No. 30 de L'Enseignement Mathématique, Université de Genève, 1982.

[NSW92] N. Nisan, E. Szemerédi, and A. Wigderson. Undirected connectivity in $O(\log^{1.5} n)$ space. In *Proceedings 33rd Annual Symposium on Foundations of Computer Science*, pages 24–29, Pittsburgh, PA, Oct. 1992. IEEE.

[Poo93a] C. K. Poon. Space bounds for graph connectivity problems on node-named JAGs and node-oriented JAGs. In *Proceedings 34th Annual Symposium on Foundations of Computer Science*, Palo Alto, CA, Nov. 1993. IEEE.

[Poo93b] C. K. Poon. A sublinear space, polynomial time algorithm for directed st-connectivity on the JAG model. Manuscript, University of Toronto, 1993.

[Sav70] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2):177–192, 1970.

[Sav73] W. J. Savitch. Maze recognizing automata and nondeterministic tape complexity. *Journal of Computer and System Sciences*, 7(4):389–403, 1973.

[Sze88] R. Szelepcsényi. The method of forcing for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.

[Tom82] M. Tompa. Two familiar transitive closure algorithms which admit no polynomial time, sublinear space implementations. *SIAM Journal on Computing*, 11(1):130–137, Feb. 1982.

[Wig92] A. Wigderson. The complexity of graph connectivity. In I. M. Havel and V. Koubek, editors, *Mathematical Foundations of Computer Science 1992: Proceedings, 17th Symposium*, volume 629 of *Lecture Notes in Computer Science*, pages 112–132, Prague, Czechoslovakia, Aug. 1992. Springer-Verlag.