Alexandria University
Faculty of Engineering
Computer and Systems Engineering Department
Computational Theory Cousrse

Report About Un-Directed ST-connectivity in log Space

Presented To : Dr. Waild Gomaa
Presented By :
Moustafa Emara no.62
Mahmoud Basiony no . 57

Part I : Research Overview:

For more than two decades, undirected connectivity was one of the most appealing examples of the computa tional power of randomness. Whereas every graph (e.g., a planar graph representing a maze) can be efficiently traversed by a deterministic algorithm, the classical deterministic algorithms required an extensive use of (extra) memory (i.e., linear in the size of the graph). On the other hand, it was known that, with high probability, a random walk (of polynomial length) visits all vertices in the corresponding connected component. Thus, the randomized algorithm requires a minimal amount of auxiliary memory (i.e., logarithmic in the size of the graph). Even after more than a decade of focused attension at the issue, a significant gap remained between the space complexity of randomized and deterministic polynomial-time algorithms for this natural and ubiquitous problem. Undirected connectivity became the most famous example where randomized computations seemed more powerful than deterministic ones.
Omer Reingold presented his recent breakthrough result as- serting that any graph can be traversed by a deterministic polynomial-time algo- rithm that only uses a logarithmic amount of auxiliary memory. His algorithm is based on a novel approach that departs from previous attempts, where the latter tried to derandomize the random-walk algorithm. Instead, Reingold's algorithm traverses a virtual graph, which (being an "expander") is easy to traverse (in deterministic logarithmic-space), and maps the virtual traversal of the virtual graph to a real traversal of the actual input graph. The virtual graph is constructed in (logarithmically many) iterations, where in each iteration the graph becomes easier to traverse.
Here We will Present the USTCON problem and overview of the efforts done in solving it , and we will present Omer Reingold Deterministic Logarithmic Space Algorithm , Proving That

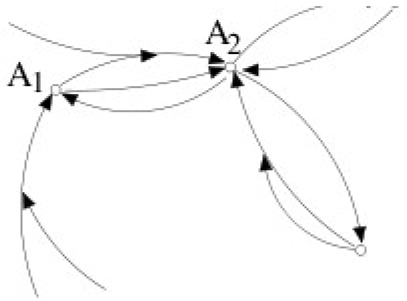
$$\mathbf{L = SL}$$

Part II : Defination of STCON , USTCON

The st-connectivity problem , also referred as the st-reachability problem is the following :
Given a directed graph G = (V , E) and two vertices **S** and **T** belongs to V , is there a path from **S** to **T** in G ?
If the edges of the graph are undirected , The problem is called undirected st-connectivity.
So The abbreviations STCON for the ST-connectivity and USTCON for the undirected ST-connectivity.
USTCONN = { G, s, t | G– undirected graph, s, t ∈ V (G); s and t are connected in G}

Part III :Symmetric Turing Machines and Symmetric Logarithmic Space Class

- Defination of Symmetric Turing Machines:
    - SL was first defined in 1982 by Lewis and Papadimitriou[1], who were looking for a class in which to place USTCON, which until this time could, at best, be placed only in NL, despite seeming not to require nondeterminism. They defined the symmetric Turing machine, used it to define SL. Symmetric Turing machines are kind of Turing machines with limited nondeterministic power. Informally Symmetric computations are reversible in an approximate manner. A symmetric computation is divided into locally deterministic sub computation (segments) between special configuration , where nondeterminism takes place , such that the only special configuration reached from a backward computation is the special configuration started the segment. Thus the segments are locally deterministic in its forward direction and have limited nondeterminism in its backward direction . Also if there is a way from special configuration A1 to special configuration A2 , there must be a way back from A2 to A1.



Special Configurations where nondeterminism happens

- *Symmetric Log Space Complexity:*
    - *SSPACE(S(n) is the class of the languages accepted by a symmetric Turing machine running in space O(S(n))*
    - SL is The class of problems solvable by a nondeterministic Turing machine in logarithmic space, such that

1. If the answer is 'yes,' one or more computation paths accept.
2. If the answer is 'no,' all paths reject.
3. If the machine can make a nondeterministic transition from configuration A to configuration B, then it can also transition from B to A. (This is what 'symmetric' means.)
4. It was proved That SL = Co-SL
5. The Original Relation Between Classes

$$L \subseteq SL \subseteq NL$$

- *Symmetric Log Space Complete Problems*
    - The most important Complete Problem is the USTCON , Lewis and Papadimitriou by their defination showed that USTCON is complete for SL class . The idea that they constructed a nondeterministic machine for USTCON , and they made a leema for converting this machine into Symmetric Turing Machine. Then the theorem follows as any language can be accepted using a symmetric Turing machine is log-space reducible to USTCON as from the properties of the symmetric computation we can view the special configuration as the undirected edges of the graph
    - Some other complete problems
        - Simulation of symmetric Turing machines: does an STM accept a given input in a certain space, given in unary?
        - Vertex-disjoint paths: are there *k* paths between two vertices, sharing vertices only at the endpoints? (a generalization of USTCON, equivalent to asking whether a graph is *k*-edge-connected)
        - Is a given graph a bipartite graph, or equivalently, does it have a graph coloring using 2 colors?
        - Do two undirected graphs have the same number of connected components?
        - Given a graph, is there a cycle containing a given edge?

Part IV Efforts Done for Solving USTCON :

- Svatich (1970) :Showed That NL is subset L 2 implies a deterministic (log n) 2 space bound for SL directly. Remarkably, since then, all progress went via probabilistic algorithms for USTCON and their derandomization.

- In the late 70's, Cook suggested universal traversal sequences (UTS) as a basis for log-space algorithms for USTCON. A traversal sequence is a (deterministic) instruction sequence for a pebble moving on the vertices of a graph, in much the same way as the random coin provides such instructions in a random walk. Such a sequence is universal if it eventually leads the pebble to visit all nodes in every connected graph of a given size.

- Aleliunas et al. [1979] proved not only the existence of such a UTS of polynomial length, but did it via the probabilistic method, giving in particular a probabilistic log-space (RL) algorithm for USTCON.Thus, they established SL is a subset for RL . Unfortunately, it did not provide deterministic space-efficient algorithms to generate such short UTS

.

- In a seminal paper, Nisan [1992] proved that a UTS can be constructed in L 2 . This construction was based on a pseudorandom generator that fools RL machines. In particular, it can be used to derandomize the above probabilistic algorithm. This hierarchical generator requires log n universal hash functions of O(log n) bits each. While not directly improving the deterministic space bound for USTCON, Nisan's techniques were the basis of all subsequent progress, starting with his own paper [Nisan 1994] gives a (log n) 2 -space, polynomial time algorithm for USTCON).
- The first reduction in space for this problem was achieved by Nisan et al. [1992] who proved SL is subset from L 3/ 2 . The key idea is to scale down Nisan's UTS. They use short (O(log k) bits) UTS's from every vertex of the graph to visit large (size k) neighborhoods. Then a pairwise independent sample of the vertices, which is easily derandomized, is used to create a new graph which is much smaller (by a factor of k), but still captures the connectivity essence of the original. Iterating this process eventually leads to solving USTCON on a 2-node graph. The bottleneck for improving this bound was that log-space UTS can only guarantee neighborhoods of size exp(log n), implying a similar shrinking in size per iteration, which implies log n iterations

- ROY ARMONI showed That SL is subset from O(log(n)4/3) space

From what is seen from the efforts is finding ways for derandomization of the algorithms used.

Part V : Omer Reingold Solution for Undirected S-T connectivity in Log Space Complexity

Proof Idea and Approach : If you want to solve connectivity problem for your input graph first improve its connectivity , in other words transfer your input graph or each of its connected components to an expander. Insisting the final graph being constant degree. Once the connected component of S is an expander , it is trivial to decide whether t is connected or not , as expanders have logarithmic diameter. It is enough to enumerate all the paths starting form s to see whether T is connected to S or not. Since the degree of the graph is constant , the number of paths is polynomial and can be visited in Log-Space .

- Tools for Understanding the algorithm
  - **Basics :**
    - D-Regular graphs : The sum of edges from each vertix is equal to D
    - If the graph is D-regular, the sum of each row is equal
    - Normalized Adjaceny Matrix : Is the matrix where we divide each value by D (the degree of the graph)
    - For normalized matrix , the largest eigen value is 1 wit the vector (1 , 1 ....)
    - A (N , D , λ) is a D-regular graph over N-vertices where λ(G) <= λ
    - For a d-regular graph G with adjacency matrix A whose eigenvalues are (in descending order) $\lambda 0 \geq \lambda 1 \geq . . . \geq \lambda n-1$ the following are true:
      - $\lambda 0 = d$.
      - $\lambda n-1 \geq -d$
      - G is connected iff $\lambda 0 > \lambda 1$.
      - G is bipartite iff $\lambda n-1 = -\lambda 0$.

  - **Expanders** :
    - Defination : An **expander graph** is a sparse graph which has high connectivity properties, quantified using vertex or edge expansion as described below. Expander constructions have spawned research in pure and applied mathematics, with several applications to computer science, and in particular to theoretical computer science, design of robust computer networks and the theory of error-correcting codes.
    - For a graph G = (V, E) and two sets of vertices A, B ⊆ V we denote by

      $E(A, B) = \{ \{u, v\} \in E \mid u \in A, v \in B\}$

      - the set edges between A and B. For a set of vertices S ⊆ V we denote the complement set by S = V \ S.

- We defined the expansion of G to be

$$h(G) = \min_{\substack{S \subseteq V \\ |S| \leq \frac{|V|}{2}}} \frac{|E(S, \overline{S})|}{|S|}.$$

- G = (N,D, λ) is an expander iff the spectral gap 1- λ > 0
- G = (N,D, λ) is an expander if there exists ε >0 such that for any set S of at most half the vertices in G, at least (1+ ε)|S| vertices of G are connected to some vertex in S
- In Expanders : The path between S and T is of Length O(log n)
    - Proof :From any vertex s there are $D^{l=O(\log N)}$ different paths. Simply enumerate them all and see if any of them reach t.
    - At each vertex we have a choice of D vertices, log(D) to represent 1…D. Each path is log(N) long.
    - we need log(D) at each stage of a log(N) path, altogether O(logD*logN)

- **Rotation Maps :**
    - Defination : Let G be a D regular undirected graph. The **rotation map**

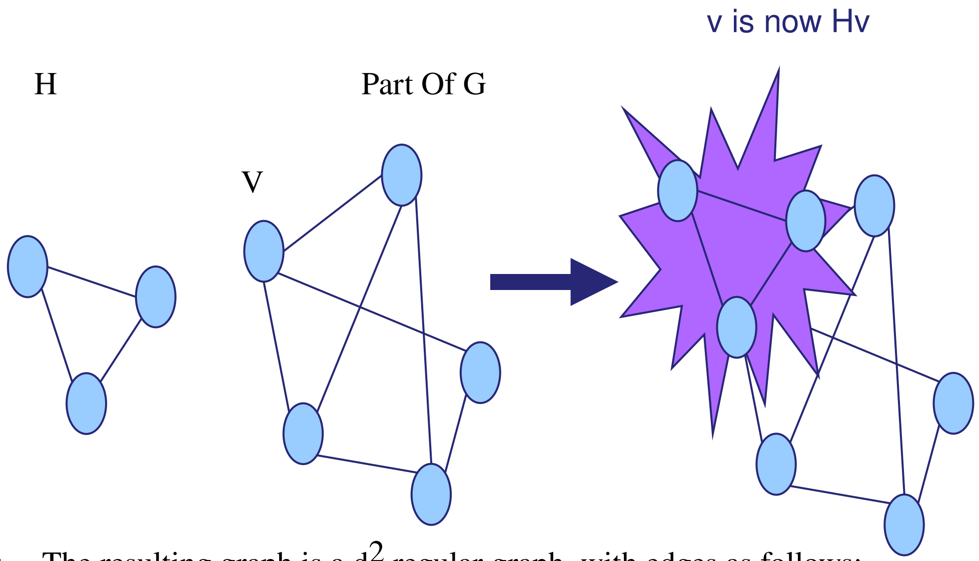
$$\mathrm{Rot}_G : [N]\ [D] \quad [N]\ [D]$$
$$\mathbf{Rot}_G\,(\mathbf{v}, \mathbf{i}) = (\mathbf{w}, \mathbf{j})$$

If the edge (v,w) exists and is the ith edge coming out of v and the jth edge coming out of w

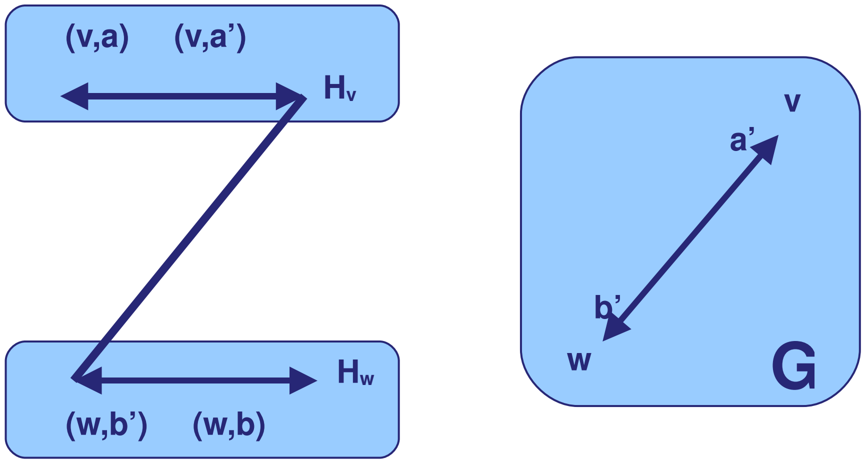
- **Powering :**
    - Defination : If we had no limitations on the degree of the graph, we could make it into an expander graph by **powering:**
    - The k-th power of the D-regular graph G is the graph G^k where there is an edge (u,v) iff there is a path of length ≤ k between u and v in G
    - In rotation map notation, this means that

$$\mathrm{Rot}_{G^k}\left(v_0, (a_1 \ldots a_k)\right) = \left(v_k, (b_k \ldots b_1)\right) \text{ where } (v_i, b_i) = \mathrm{Rot}_G\left(v_{i-1}, a_i\right)$$

- **Zig-Zag Products :**
  - Defination : If G is a D-regular graph with N vertices and H is a d-regular graph with D vertices and rotation map Rot_H, we replace each vertex v in G with a copy of H, H_v. Therefore, our new graph has [N]x[D] vertices.



  - The resulting graph is a d^2 regular graph, with edges as follows:



- **Rot** **((v,a), (i,j)) =** $((w,b),(j',i'))$ where
  $(a',i') = \mathrm{Rot}_H(a,i)$
  $(w,b') = \mathrm{Rot}_G(v,a')$
  $(b,j') = \mathrm{Rot}_H(b',j)$

- If G is an (N,D,λ) graph and H is a (D,d, ’) graph, then zig-zag product provides an expander with ample spectral gap!

Proof :

- Show a transformation that turns every connected component of a graph into an expander But Not any graph, but a D^{16} regular graph.
  Such a graph H can be found either

- by exhaustive search or by using one of the expander constructions. we will assume that the input graph G for which we need to check (s, t) connectivity is a d16 -regular non-bipartite graph. We will later remove these restrictions on G.

- Furthermore, we will assume G is a connected graph. Actually, this is a stupid assumption since if G were indeed connected, then there is nothing to prove. What we actually mean is the following: Reingold's algorithm works independently for each connected component of the graph and checks if t exists in the connected component that contains s. Since every component of G is d16 - regular, connected and non-bipartite, we have that $1 - \lambda(C) \geq 1/d16\ n2$, for all components C of G.

- **Reignold Algorithm**
  - Input: G – d16 -regular graph and two vertices s, t $\in$ V (G).

  - 1. Set l to be the smallest integer such that
    1 – (1/ d^16 * n^2))^2^l < 1/2
    Comment: l is O(log n)

  - 2. Set G0 $\leftarrow$ G.

  - 3. For i = 1, . . . , l do, set Gi $\leftarrow$ (Gi–1 **Z** H)8 .

    Comment: (1) Each Gi is a d16 -regular graph.
    (2) Each connected component of Gl is an expander with spectral expansion at most 1/2 (proved Below)
  - Check if s and t are connected in Gl by enumerating over all O(log n) paths

originating at s.

- **Proofing the algorithm:**

  - We first prove the comment in Step 3, which will suffice to prove the correctness of Reingold's algorithm. For this we need the following proposition.

    - For i = 1, . . . , l, λ(Gi) ≤ min{λ2 (Gi−1), 1/2}.

    - Proof. Since Gi = (**G Z** H)8 , we have from Lemma 64 that λ(Gi) = λ8 (Gi−1 **Z** H) ≤ [1 − (1 − λ(Gi−1))/3]8 . Now, consider the following two cases.

      - Case (i): λ(Gi−1) ≤ 1/2. Then,

$$\lambda(G_i) = (\lambda(G_{i-1} \circledZ H))^8 \leq \left(1 - \frac{1}{3}\cdot\frac{1}{2}\right)^8 = \left(\frac{5}{6}\right)^8 < \frac{1}{2}.$$

      - Case (ii): λ(Gi−1) > 1/2. In this case, we can by expansion check that

$$\left(1 - \frac{1}{3}(1-x)\right)^4 \leq x, \qquad \text{for all } \frac{1}{2} \leq x \leq 1$$

Hence

$$\lambda(G_i) = (\lambda(G_{i-1} \circledZ H))^8 \leq \left(1 - \frac{1}{3}(1-\lambda(G_{i-1}))\right)^8 \leq \lambda^2(G_{i-1}).$$

  - By our choice of l, we have the following theorem on the expansion of each connected
component of Gl .

  - *Theorem : The spectral expansion of each connected component of Gl is at most 1/2.*

- **<u>Space Complexity of Reingold's Algorithm</u>**
    - Each squaring operation requires an additional space of O(log deg G) Similarly, it can be shown that each zigzag product with H (of constant size also requires additional space at most O(log deg G).

    - Since there are at most O(log n) squaring and zig-zag products (since l = O(log n)) and
    the degree of all the graphs is at most d16 , a constant, the total space complexity of the
    algorithm is at most O(log n).

- **<u>Handling non-regular bipartite graph</u>** :
    - To start with, we will convert the graph into a
    3-regular graph by replacing each vertex of degree d greater than 3 by a cycle of size d and
    connecting each of the d neighbors of the vertex to the d distinct points on the circle. To
    convert the graph into a d16 -regular graph, we then add d16 − 3 self loops to each vertex.
    Note, the addition of self loops also makes the graph non-bipartite. Both these conversions
    can be effected in log space.

<u>Part VI : Results of Proving L = SL</u>

The collapse of **L** and **SL** has a number of significant consequences. Most obviously, all **SL**-complete problems are now in **L**, and can be gainfully employed in the design of deterministic log-space and polylogarithmic-space algorithms. In particular, we have a new set of tools to use in log-space reductions. It is also now known that a problem is in **L** if and only if it is log-space reducible to USTCON; this may be useful for showing that problems are *not* in **L**, by showing that such a reduction doesn't exist.

Part VII : References:

- **O. Reingold, Undirected ST-Connectivity in Log-Space, STOC 2005.**
- **Lecture Notes :CS369E: Expanders in Computer Science**
  By Cynthia Dwork & Prahladh Harsha
- **http://www.cs.huji.ac.il/~advtheory/ Lecture Notes**
- **Randomization and Derandomization in Space-Bounded**
  Computation Michael Saks†
- **A Fast Randomized Log Space Algorithm for un directed graph Connectivity**
  Uriel Fiege
- **Deterministic Space Bounded Graph connectivity Algorithms**
  Jesper Janson
- **Sharon Bruckner Lecture Notes**
- **Mathematisches Forschungsinstitut Oberwolfach Report no 26/2005**
- **An O(log(n)4/3) Space Algorithm for (s, t) Connectivity**
  in Undirected Graphs ROY ARMONI
- **Symmetric Complementation**
  JOHN H. REIF
- **Symmetric LogSpace is closed undercomplement Noam Nissa**