A Time-Space Trade-off for Undirected s-t Connectivity in $\tilde{O}(n^2)$

Schilliger Julian
scjulian@student.ethz.ch

November 11, 2018

1 Introduction

1.1 Overview

In this manuscript, we will first look at the relationship between space S and time T for algorithms solving the s-t connectivity problem of undirected graphs($USTCON$). Then we build a randomized Monte-Carlo algorithm that answers the question for any given space S with Space-Time Product(also know as Space-Time Complexity) $S \cdot T = \tilde{O}(n^2)$, where the $\tilde{O}$-notation disregards poly-logarithmic terms and prove its correctness.

1.2 Motivation and Problem Description

In the problem of $USTCON$, the question is whether or not two given nodes s and t from an undirected graph G are connected through a path, meaning if s and t are belonging to the same connected (sub-)component of graph G. Depending on the data-structure of G, whether it is immutable or not and your amount of available memory(denoted as Space S), you sometimes are limited in your choice of algorithms, that solve $USTCON$. We can only solve the problem in a deterministic manner if the available space S is larger or equal than $\log(n)$(Aleliunas, 1979). For smaller S, we need to rely on randomized algorithms. Therefore it is important to know algorithms that are fast(execution time T is as small as possible), even if the available space is restricted. Having access to efficient and fast algorithms solving $USTCON$ is key in a variety of different areas, such as route planning, network discovery or computer-aided verification.

1.3 Notation

The notation for all graph related variables will be taken over from the ETH HS18 "Algorithms, Probability and Computing" course. Anything new will be defined at the appropriate position in this manuscript.

2 Time-Space Complexity

In 1994 Broder et al. found a correlation between time complexity and space complexity in randomized algorithms solving USTCON (Broder, 1994). They observed that DFS, BFS and random walks had the same time-space trade-off of $T = \tilde{O}(\frac{mn}{S})$. They investigated if there exist algorithms with the same trade-off for every space S. What they found was the Landmark approach(explained in section 3.2) that had $T = \tilde{O}(\frac{mn}{S})$ for every possible space S between $\tilde{O}(1)$ and $\tilde{O}(n)$. Further research in this area gave us algorithms that have a runtime of $\tilde{O}(n + m)$ and are, in general more space efficient than BFS or DFS(both with time (m) and space Θ(n))(Kosowski, 2012).

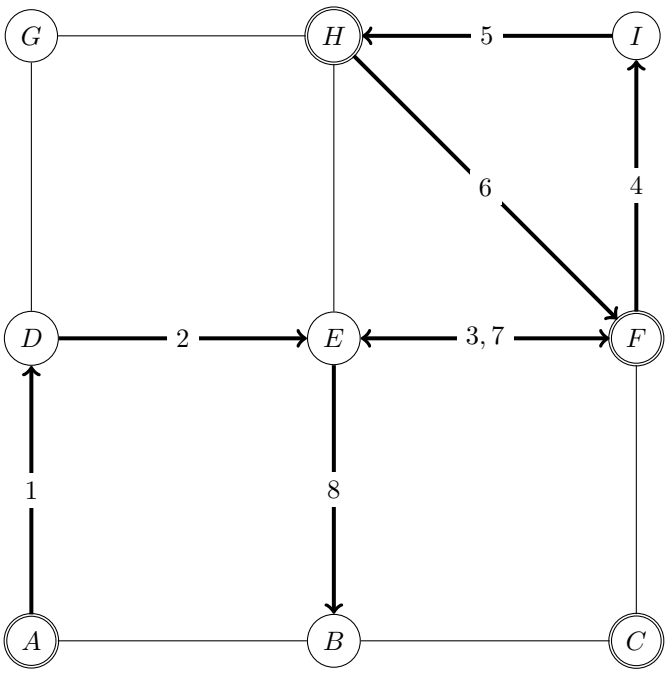
3 An Algorithm with $S \cdot T = \tilde{O}(n^2)$

3.1 Idea

The proposed algorithm from Kosowski makes use of two previously known key ideas: Landmarks and Metropolis Hasting Walks. By replacing the random walks in the Landmark approach with Metropolis Hasting walks, he nearly arrives at an algorithm with the desired trade-off of $S \cdot T = \tilde{O}(n^2)$.

3.2 Landmarks

Landmarks L are a set of uniformly at random picked nodes from the given Graph G with $L \subseteq V \wedge s, t \in L$. Their cardinality depends on the amount of available space: $p = |L| = \tilde{O}(S)$. They are saved in a Union-Find data-structure with initially all belonging to another union. Those Landmarks serve as starting points for Metropolis Hasting Walks(initially random walks (Broder, 1994)). A Metropolis Hasting Walk is a sequence of visited nodes $(p_0, p_1, p_2, ..., p_k)$ with $(p_i, p_{i+1}) \in E \; \forall i \in [0, k), i \in \mathbb{N}$. The exact nature of the walk will be explained later. Whenever a walk originating from a landmark $a \in L$ hits another landmark $b \in L \setminus a$ the Union-Find data-structure gets updated by calling union(a, b) if a and b belong to a different union(which can be checked in $O(1)$). This results in at most p calls to the function union(node1, node2) with a known total amortized time complexity of $O(p \cdot \log(p)) \leq \tilde{O}(n)$. Here is an example of a walk in a graph to illustrate the process:



Double circled nodes represent the nodes from the set of Landmarks L. The random walk is drawn by arrows, the number denotes their order. The walks always start from a landmark node(in this case A). If the reached node is a landmark that does not belong to the same union as A, then A and this node gets unionized. This happens for the first time after step 3 at node F, then again with H. Since after step 6 F is a landmark but already in the same union as A, there is no call to the union function. In step 7, the same edge gets traversed as in step 3, but in the opposite direction. Nodes G and landmark C are not reached by this specific walk of length 8.

To complete the algorithm, we launch q walks from each of the $p = \Theta(S)$ Landmarks. The algorithm returns true if after all walks node s and t are in the same union and false else. If at the end of the algorithm s and t are in the same union, then they are connected and the given answer is right. There are two possibilities for the case that s is not in the same union as t:

- s and t are not connected, then the answer is correct
- s and t are connected but the connection was not found, then the answer is wrong

Note: This makes the proposed algorithm a Monte-Carlo type one with one sided error.

3.3 Metropolis Hasting Walks

The idea behind the Metropolis Hasting Walk is to be able to explore graphs faster than with a random walk, and still only use $S = \tilde{O}(1)$ space. We assign each edge $e = (u, v) \in E \, | u, v \in V \wedge u \neq v$ of the graph G a weight value such that we can easily pick an edge at random in dependence of their weight:

$$w_{v,u} = \min\left(\frac{1}{deg(v)}, \frac{1}{deg(u)}\right) \tag{1}$$

We then introduce self-loops with (positive) weight:

$$w_{v,v} = 1 - \sum_{u \in N(v) \setminus v} w_{v,u} \tag{2}$$

This way, the sum of all weights of edges incident to node v, including a self-loop, is equal to 1. We now can pick an incident node at random with the weight function distribution in time and space $\tilde{O}(1)$:

Algorithm 1 State transition function on V

```
procedure NEXTNODE(CURRENTNODE)
    u ← neighbor of v in G chosen uniformly at random          ▷ pick a new node
    if With probability min(deg(v)/deg(u), 1) then return u    ▷ decide whether to accept the node
    else return v                                              ▷ do not accept
```

One walk of length d has therefore runtime $\tilde{O}(d)$ with usage of $\tilde{O}(1)$ space.

3.3.1 Properties

Since the Landmark approach is proven by random walks and this algorithm uses Metropolis type walks; we need to show that these walks are more capable than random walks and how we have to use them to achieve our goal. Since the described walks are a reversible Markovian process, meaning starting at a node a and arriving at a node b has the same probability as reversing the walk. $\mathbb{E}_i Z_j(t)$ is the expected value of the random variable that describes the number of times node j was reached up to time t, by starting a walk from i. The lowered part of those formulas might also represent an edge and is defined accordingly. One call to Algorithm 1 represents the transition between t to $t+1$, t describes therefore the number of traversed edges(length of the taken walk) or the time-steps since the Metropolis Hasting Walk started. We observe following properties (Kosowski, 2012):

- $\mathbb{E}_i Z_j(t) = \mathbb{E}_j Z_i(t)$
- $\mathbb{E}_\pi Z_i(t) = \frac{t}{n}$,π describes the uniform distribution of nodes, such that $\pi(v) = \frac{1}{n}$, for all $v \in V$
- with the above $\mathbb{E}_\pi Z_{e_{i,j}}(t) = \mathbb{E}_\pi Z_i(t) \cdot w_{i,j} = \frac{t}{n} \cdot w_{i,j}$
- $Com(i,j) = \mathbb{E}_j T_i + \mathbb{E}_i T_j$,$Com(i,j)$ is the time it takes to commute between i and j and $\mathbb{E}_j T_i$ is the time it takes to reach i from j

Lemma 1. *From Kosowski (2012): Suppose that G is connected and $\triangle$ is the maximal degree of a node in G. Let $A \subsetneq V$, and let $i \in A$. For a weighted random walk $RW(G_1)$ starting at node i:*

- *the expected time to reach a node from $V \setminus A$ is bounded by:*

$$\mathbb{E}T_{V \setminus A} < (|A| + 1)(6|A| + 2\triangle) \tag{3}$$

- *the expected number of visits to node i before any time $0 < t < 6n^2$, is bounded by:*

$$\mathbb{E}Z_i(t) < 5\sqrt{t} + 2\triangle \tag{4}$$

Proof. We construct a new graph G$^\circ$ as sub-graph of G induced by all nodes from set A, their neighbors in G and node $j \in V \setminus A$, where j is the nearest vertex from $i \in A$ that is not in A. The relation can be set up as follows, where the commute between i and j is $Com(i,j) = R(i,j) * \sum_{e \in E} w_e$ ($R(i,j)$ describes the resistance between i and j):

$$\mathbb{E}T_{V \setminus A} \leq \mathbb{E}T^\circ_{V \setminus A} = \mathbb{E}_i T^\circ_j < Com^\circ(i,j) = R^\circ(i,j) \cdot \sum_{e \in E(G^\circ)} w_e \tag{5}$$

Since the weights of edges are defined as $w_{v,u} = \min(\frac{1}{deg(v)}, \frac{1}{deg(u)})$, $R^\circ(i,j)$ can be upper bound by the resistances of the shortest path with length of $a \leq |A|$ and $i_0 = i$:

$$R^\circ(i,j) = \frac{1}{w_{i_0,i_1}} + \frac{1}{w_{i_1,i_2}} + ... + \frac{1}{w_{i_{a-1},j}} = \max(deg(i_0), deg(i_0)) + ... + \max(deg(i_{a-1}), deg(j)) < 2\sum_{c=0}^{a-1} deg(i_c) + 2\triangle \tag{6}$$

Because $\sum_{c=0}^{a-1} deg(i_c)$ describes the total deg's of a shortest path in A, $\sum_{c=0}^{a-1} deg(i_c) \leq 3|A|$ holds, therefore:

$$R^\circ(i,j) < 2\sum_{c=0}^{a-1} deg(i_c) + 2\triangle < 6|A| + 2\triangle \tag{7}$$

All edges from $e \in E(G^\circ)$ are incident to at most $|A|+1$ nodes, and every node has edges with added weights of 1, so $\sum_{e \in E(G^\circ)} w_e \leq |A| + 1$, resulting in the first statement of the lemma:

$$\mathbb{E}T_{V \setminus A} < R^\circ(i,j) * \sum_{e \in E(G^\circ)} w_e < (|A|+1)(6|A| + 2\triangle) \tag{8}$$

□

Proof. The second statement can be rewritten as follows, where $Pr[pick] = \frac{1}{n}$, pick := "picking node j" :

$$\mathbb{E}Z_i(T_{V \setminus A}) < Com(i,j) \cdot Pr[pick] < (6|A| + 2\triangle) \cdot \sum_{e \in E} w_e \cdot \frac{1}{n} < (6|A| + 2\triangle) \cdot n \cdot \frac{1}{n} = 6|A| + 2\triangle \tag{9}$$

With $s = \sqrt{6t}$ and $|A| < \frac{t}{s} \leq n$ (Kosowski, 2012):

$$\mathbb{E}Z_i(T_{V \setminus A}) < 6\frac{t}{s} + 2\triangle \tag{10}$$

And $\mathbb{E}Z_i(t) \leq \mathbb{E}Z_i(T_{V \setminus A}) + s < 5\sqrt{t} + 2\triangle$ □

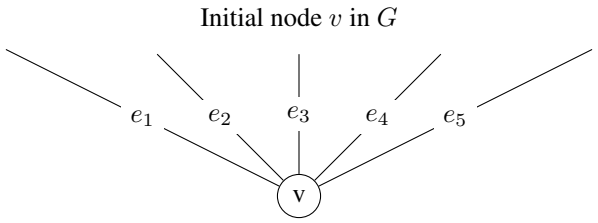
The above properties sadly only hold if the walks have a length $d = \Omega(\triangle^2)$. This issue can be overcome with adapting the graph by splitting nodes, such that the maximum degree $\triangle$ of a node can be chosen with an appropriate size(more on that later).

3.4 Runtime

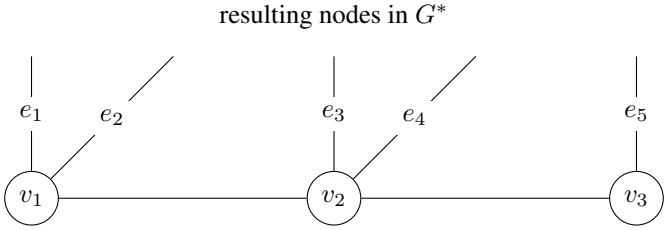
The properties of the last subsection can now be used to be plucked into the proof of using landmarks with random walks and replace the random walks by Metropolis Hasting walks. We arrive at the conclusion that by using $q = 72\log(n)$ walks of length $d = n_p^2$, $n_p = \max\left(60\frac{n}{p}\log(n), \triangle\right)$ from every of the p landmark nodes, the algorithm gives the right answer with probability of $1 - \frac{1}{n}$. Therefore the algorithm gives the right answer with high probability(but only for $d = \Omega(\triangle^2)$). Its running time is $\tilde{O}(q \cdot d \cdot p + n) = \tilde{O}\left(\frac{n^2}{S}\right)$ which achieves the desired time-space trade-off of $S \cdot T = \tilde{O}(n^2)$.

4 Tune the Graph G to remove the dependence on $d = \Omega(\triangle^2)$

To find an algorithm for every S, we need to be able to make the length of the walks $d = \Omega(\triangle^2)$ independent from $\triangle$. To achieve this we therefore need to be able to lower the maximum degree ($\triangle$) of nodes in G to some desired value $D + 2$. We create a new Graph G^* from G by splitting all nodes that have degree $> D + 2$ into multiple nodes, where those new nodes are connected with D nodes from the old graph G and have two connections(or one if it is the "rightmost" or "leftmost" node created from v) to other newly created nodes which originated from the same node $v \in V$. Those connections are shaped like a path such that if you only look at newly created nodes out of a node v, then those nodes build a connected component. To better understand it, here is a visualization with $D = 2$:



Right now node v has $deg(v) = 5 > 4 = D + 2$. We now split node v into v_1, v_2 and v_3:



Each node v_1, v_2 and v_3 has degree $\leq D + 2 = 4$ now. Building this new graph G^* costs at most $\tilde{O}(m)$ time, since there are only m edges in G. If one runs the described algorithm on this newly created graph, he gets runtime $\tilde{O}\left(m + n_p^{*2} \cdot p\right)$, $n_p^* = \max\left(60\frac{n^*}{p}\log(n^*), \triangle^*\right) \leq \max\left(60\frac{n+2m/D}{p}2\log(n), D+2\right)$. Therefore:

$$T = \tilde{O}\left(m + \frac{n^2}{p} + \frac{m^2}{D^2 p} + D^2 p\right) \tag{11}$$

Proof. Choosing $D = \lceil\sqrt{m/p}\rceil$ to replace D:

$$T = \tilde{O}\left(m + \frac{n^2}{p} + \frac{m^2}{m} + m\right) = \tilde{O}\left(\frac{n^2}{S} + m\right) \tag{12}$$

Which proves, that there is an algorithm with $T = \tilde{O}(\frac{n^2}{S} + m)$ for every S between $\tilde{O}(1)$ and $\tilde{O}(n)$. □

References

R. Aleliunas. Random walks, universal traversal sequences, and the complexity of maze problems. *Proc. 20th Annual Symposium on Foundations of Computer Science (FOCS)*, page 218–223, 1979.

Broder. Trading space for time in undirected s-t connectvity. *SIAM journal on Computing*, 23:324–334, 1994.

Adrian Kosowski. Faster walks in graphs: O(n^2) time-space trade-off for undirected s-t connectvity. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 2012.