Spanners

Quality measures and efficient construction

Dmitriy Traytel

October 5, 2010

Abstract

This document is a written summary for a presentation on "spanners" given at the course 2 of the Ferienakademie in Sarntal 2010. It contains the definition of a (α, β)-spanner together with three construction algorithms for different approximation qualities.

Contents

1 Introduction

The running time of an algorithm that is solving a shortest paths or distances problem almost always depends on the number of edges of the given graph. Thus, if the graph is too dense, one can not expect an appropriate running time. Spanners are an approach to resolve this problem. The idea is to change the graph by removing edges such that the shortest paths are not stretched by more than some given bound.

Definition 1.1 (Spanner). *Let $G = (V, E)$ be an unweighted graph. A subgraph $H = (V, E')$ of G is called* (α, β)-spanner *if for all $u, v \in V$ it holds:*

$$\delta_H(u, v) \leq \alpha\delta_G(u, v) + \beta.$$

H is called additive *if $\alpha = 1$ and* purely additive *if $\alpha = 1$ and $\beta \in \mathcal{O}(1)$.*

In the following we denote by n the number of vertices, by m the number of edges. Further notation can be found in the appendix.

When constructing a spanner, there has to be a trade-off between the approximation quality, the size of the spanner and the construction time. The approximation quality is explicitly given by the values α and β. The best case for these to values is $\alpha = 1$, $\beta = 0$, i.e. the spanner conserves all shortest distances of the original graph. The number of edges is obviously a reasonable measure of the size of the spanner. The linear[1] construction time is the best that can be achieved. Table 1 shows a selection of known spanner constructions.

(α, β)	Number of edges	Construction time
$(2k-1, 0)$	$\mathcal{O}(n^{1+1/k})$	$\mathcal{O}(m)$
$(k-1, 2k-\mathcal{O}(1))$	$\mathcal{O}(n^{1+1/k})$	$\mathcal{O}(mn^{1-1/k})$
$(k, k-1)$	$\mathcal{O}(n^{1+1/k})$	$\mathcal{O}(m)$
$(1+\epsilon, 4)$	$\mathcal{O}(\epsilon^{-1}n^{4/3})$	$\mathcal{O}(mn^{2/3})$
$(1, 6)$	$\mathcal{O}(n^{4/3})$	$\mathcal{O}(mn)$
$(1, 2)$	$\mathcal{O}(n^{3/2})$	$\mathcal{O}(m\sqrt{n})$

Table 1: The spanners zoo

In the following three of these constructions will be presented.

2 Purely additive (1,6)-spanner

The following $(1, 6)$-spanner construction was introduced in [BKMP05].

2.1 Construction

The algorithm is divided in two phases. In the clustering phase the vertices of the graph are divided into $n^{3/2}$ clusters such that every cluster has a central node (i.e. a node that is adjacent to all other nodes in the cluster). The result of the clustering phase is the subgraph of the input graph that consists of the radius one spanning trees of these $n^{3/2}$ clusters together with all edges that are incident to unclustered vertices.

The second phase adds more edges to reach the needed approximation quality. Edges are only added if they are "worth" it. More precisely, an edge of a shortest path is included in the spanner if the value of the path is greater than a half of its cost[2].

[1] in the number of edges

[2] for the definition of *value* and *cost* see appendix

Algorithm 1: (1, 6)-spanner construction

```
input  : Graph $G = (V, E)$
output: $(1, 6)$-spanner $H$ of $G$
begin
    $S \leftarrow V$                                        // phase 1 - clustering
    for $i \leftarrow 1$ to $n^{2/3}$ do
        $v_i \leftarrow \arg\max_{x \in S} |\Gamma_{G[S]}(x)|$
        $C_i \leftarrow \{v_i\} \cup \Gamma_{G[S]}(v_i)$
        $S \leftarrow S \setminus C_i$
    $E' \leftarrow \{(v_i, x_{v_i}) \mid 1 \le i \le n^{2/3}, x_{v_i} \in C_i\} \cup \{(u, w) \mid u \in S, w \in \Gamma_G(u)\}$
    $H \leftarrow (V, E')$                                  // phase 2 - path buying
    foreach path $P \in \mathcal{P}_G$ do
        if $2 \cdot value_H(P) \ge cost_H(P)$ then  $H \leftarrow H \cup P$
    return $H$
end
```

2.2 Theoretical considerations

From the algorithm, it is unclear why the result must be a (1, 6)-spanner. The prove is subdivided in two steps. First we define a certain property that helps us to derive the (1, 6)-bound. In the second step we prove that this defined property holds for the graph resulting from the algorithm.

Definition 2.1 (Contented)**.** *We call a subgraph H of G that contains the resulting graph of the clustering phase* contented *if for any two clustered vertices u_0, u_q there exists a shortest path $P_G(u_0, u_q)$ and a cluster $C \in \mathcal{C}(P)$ such that for $i \in \{0, q\}$:*

$$\delta_H(\mathcal{C}(u_i), C) \le \delta_P(\mathcal{C}(u_i), C)$$

Lemma 2.2. *H is contented $\Rightarrow$ H is $(1, 6)$-spanner of G*

Proof. Assume H is contented. As all edges incident to unclustered vertices at the end of the clustering phase are added to the spanner, it is enough to consider only clustered vertices. Let $\langle u_0, u_1, \ldots, u_{q-1}, u_q\rangle$ be a shortest path between arbitrary two clustered nodes u_0 and u_q. Then there must exist a cluster C in the clustering of H that fulfils the inequality from the definition of contented. Figure 2.2 visualises the possible paths between clusters and allows us to bound the distance between u_0 and u_q by the following chain of inequalities.

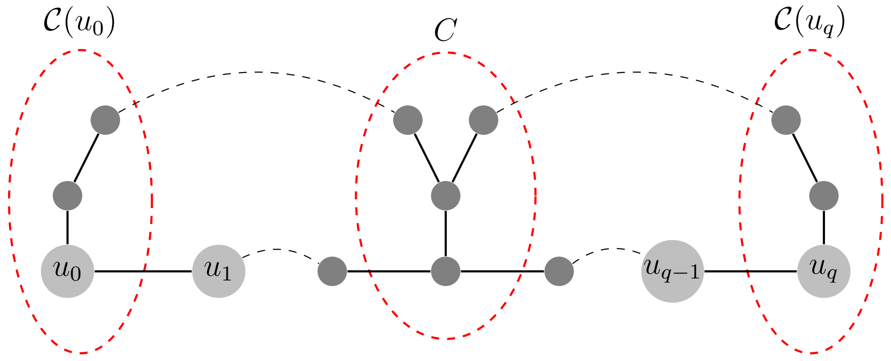

Figure 1: (u_0, u_q)-paths in the graph, $\langle u_0, u_1, \ldots, u_{q-1}, u_q\rangle$ a shortest path

$$\begin{aligned}\delta_H(u_0, u_q) &\leq \operatorname{diam}(\mathcal{C}(u_0)) + \delta_H(\mathcal{C}(u_0), C) + \operatorname{diam}(C) + \\ &\quad \delta_H(C, \mathcal{C}(u_q)) + \operatorname{diam}(\mathcal{C}(u_q)) \\ &\leq 2 + \delta_H(\mathcal{C}(u_0), C) + 2 + \delta_H(C, \mathcal{C}(u_q)) + 2 \\ &\overset{H \text{ contented}}{\leq} \delta_P(u_0, u_q) + 6\end{aligned}$$

□

Lemma 2.3 (without proof, see [BKMP05]). *For $P \in \mathcal{P}_G$ it holds: either $|\mathcal{C}(P)| = 1$ or $\exists$ subpath $P' \subseteq P$ such that $\mathcal{C}(P) = \mathcal{C}(P')$ and cost$_H(P') \leq 2|\mathcal{C}(P')| - 3$*

Lemma 2.4. *The given algorithm computes a contented subgraph of G.*

Proof. We consider a shortest path $P = P_G(u_0, u_q)$. The non-trivial case is when u_0 and u_q are clustered in different clusters of H. Lemma 2.3 provides a subpath P' of P. Again the interesting case is when $P' \notin H$. From the fact that P' was "refused" in the path buying phase and lemma 2.3 have:

$$2 \cdot \text{value}_H(P') < \text{cost}_H(P') \leq 2 \cdot |\mathcal{C}(P')| - 3.$$

Now, we consider the set:

$$A = \{\{C, C'\} \mid C \in \{\mathcal{C}(u_0), \mathcal{C}(u_q)\}, C' \in \mathcal{C}(P') \setminus \{C\}, \delta_{P'}(C, C') < \delta_H(C, C')\}$$

By its definition A has at most $2 \cdot |\mathcal{C}(P)| - 3$ elements. But the fact that A is obviously a subset of the set that is computed to gain the value of P' gives us a better upper bound for the cardinality of A. It holds:

$$|A| \leq \text{value}_H(P') \leq |\mathcal{C}(P)| - 2$$

Thus, there must exist al least $|\mathcal{C}(P')| - 1$ pairs C, C' such that $\delta_{P'}(C, C') \geq \delta_H(C, C')$. Finally, by the pigeonhole principle there must exist a $C'' \in \mathcal{C}(P') = \mathcal{C}(P)$ such that both:

$$\delta_P(C(u_0), C'') = \delta_{P'}(C(u_o), C'') \geq \delta_H(C(u_0), C'')$$

and

$$\delta_P(C(u_q), C'') = \delta_{P'}(C(u_q), C'') \geq \delta_H(C(u_q), C'')$$

hold. □

[BKMP05] proves additionally a $\mathcal{O}(n^{4/3})$ size bound and the $\mathcal{O}(nm)$ construction time, which can be achieved using a priority queue for the clustering phase and some technical tweaks for the expensive path buying phase.

3 Multiplicative spanners

Multiplicative spanners may be worse in terms of approximation, but they have their advantages in faster construction time and smaller size. We consider first the $(2k-1, 0)$ linear time construction by [ADD$^+$93] which serves as basis for the $(k, k-1)$-spanner introduced by [BKMP05].

Algorithm 2: Randomized $(2k-1,0)$-spanner construction

```
input  : Graph $G=(V,E)$ and integer $k$
output: $(2k-1,0)$-spanner $H$ of $G$
begin
    $S \leftarrow \emptyset$
    $\mathcal{C}_0 \leftarrow \{\{v\} \mid v \in V\}$
    for $i \leftarrow 1$ to $k$ do
        if $i = k$ then $\mathcal{C}_i \leftarrow \emptyset$
        else $\mathcal{C}_i \leftarrow \{C \in \mathcal{C}_{i-1} \mid \text{randomBool}(n^{-1/k})\}$
        concurrently foreach $v \in V \setminus \mathcal{C}_i$ do
            if $\exists C \in \mathcal{C}_i : \mathcal{E}(v,C) \neq \emptyset$ then                // (R1)
                $C \leftarrow C \cup \{v\}$
                $S \leftarrow S \cup \text{random}(\mathcal{E}(v,C))$
            else                                                                         // (R2)
                foreach $C \in \mathcal{C}_{i-1}$ do $S \leftarrow S \cup \text{random}(\mathcal{E}(v,C))$
    return $(V,S)$
end
```

3.1 $(2k-1,0)$-spanner

This algorithms builds something like a hierarchy of clusterings inside of the for-loop. For each level $i = 1 \dots k$ there exists a clustering $\mathcal{C}_i$ that consists of radius k spanning trees as clusters(`(R1)`). The "central nodes" are determined by sampling with probability $n^{-1/k}$. Vertices that are unclustered on level $i-1$ get connected to clusters on the level $i-1$ whenever it is possible(`(R2)`).

Again the spanner property is not obvious at all. The best way to demonstrate the intuition behind this algorithm is to resolve its magic by proving its correctness.

Lemma 3.1. *The given algorithm computes a* $(2k-1,0)$*-spanner* H *of* G.

Proof. We show that every single edge is stretched by at most $2k-1$.

Let $\{u,v\}$ be an arbitrary edge. Let l be minimal, such that either u or v is unclustered in $\mathcal{C}_l$ (note that on the k-th level all vertices are unclustered). Without loss of generality let u be the vertex that is unclustered in $\mathcal{C}_l$. By `(R2)` there exists a vertex $w \in \mathcal{C}_{l-1}(v) : \{u,w\} \in H$. By `(R1)` it holds $\delta_H(w,v) \leq 2(l-1) = \text{diam}(\mathcal{C}_{l-1}(v))$. Putting this together gives us:

$$\delta_H(u,v) \leq 1 + 2(l-1) \leq 2k-1 = (2k-1)\delta_G(u,v).$$

□

3.2 $(k,k-1)$-spanner

[BKMP05] could show that by adding few new edges to the $(2k-1,0)$-spanner from the previous subsection, one can obtain a better spanner in terms of approximation quality. There are two kinds of edges that need to be added. First, edges between clusters from pairs of clusterings whose levels add up to $k-1$ are described by `(R3)`. `(R4)` describes edges between clusterings on consecutive levels, but only in the bottom half of the clustering hierarchy. The rather technical correctness proof of this algorithm can be found in [BKMP05].

Algorithm 3: Randomized $(k, k-1)$-spanner construction

```
input  : Graph $G = (V, E)$ and integer $k$
output: $(k, k-1)$-spanner $H$ of $G$
begin
    construct the $(2k-1, 0)$-spanner (V,S)
    for $i \leftarrow 1$ to $k-1$ do                                        // (R3)
        $S \leftarrow S \cup \{\text{random}(\mathcal{E}(C, C')) \mid C \in \mathcal{C}_i, C' \in \mathcal{C}_{k-1-i}, \mathcal{E}(C, C') \neq \emptyset\}$
    for $i \leftarrow \lceil k/2 \rceil$ to $k-1$ do                          // (R4)
        $S \leftarrow S \cup \{\text{random}(\mathcal{E}(C, C')) \mid C \in \mathcal{C}_i, C' \in \mathcal{C}_{i-1}, \mathcal{E}(C, C') \neq \emptyset\}$
end
```

3.3 Algorithmic properties

Both algorithms, the (2k-1,0)- and the (k,k-1)-spanner construction provide the expected spanner size bound $\mathcal{O}(kn^{1+1/k})$. The reason is that the edge-inclusion rules `(R3)` and `(R4)` add each $n^{1+1/k}$ edges as expected value to the initial $(2k-1, 0)$-spanner. Also, both algorithms need $\mathcal{O}(km)$ deterministic construction time. Especially, `(R3)` and `(R4)` only need linear time.

An other very handy property of this algorithms is the fact that the edge-inclusion rules `(R1)`-`(R4)` can be reformulated as local rules for a node in a synchronized distributed network, such that after $\mathcal{O}(k)$ (*constant!*) rounds of execution of these rules a spanner of the network is build globally.

4 Conclusion

Spanners are necessary if the original graph is too dense to make computations on it. There exist several known constructions, that can be used according to the needs for good approximation quality, small spanner size and short construction time.

The most important applications of spanners are the computation of approximate shortest paths in distributed networks, construction of approximate distance oracles and simulation of synchronized protocols in unsynchronized networks. Spanners can also be used as basis of space-efficient routing tables.

A variation of a spanner allows to deal with directed graphs. For this purpose, the roundtrip distance must be used instead of the shortest path.

The zoo of spanners is not completely explored yet. At least there is no proof for the lowest possible size bound of a spanner. This corresponds to one of the most famous open questions in graph theory: the girth conjecture. The conjecture was stated by Erdös in 1963. It says that there exist graphs with $\Omega(n^{1+1/k})$ edges and girth[3] $2k+2$. If the conjecture holds, the proof of the lower bound of the spanner size is easy: assume $G = (V, E)$ fulfils the girth conjecture and let $(u, v) = e \in E$, $H = (V, E \setminus \{e\})$. Then $\delta_H(u, v) \geq 2k+1 \geq (2k+1)\delta_G(u, v)$ holds and thus, any (α, β)-spanner with $\alpha + \beta \leq 2k$ must have at least $\Omega(n^{1+1/k})$ edges.

Currently, only for $k = 1, 2, 3, 5$ there exists a proof of the girth conjecture.

[3]length of the shortest cycle

References

[ADD$^+$93] Ingo Althöfer, Gautam Das, David P. Dobkin, Deborah Joseph, and José Soares. On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, 9:81–100, 1993.

[BKMP] S. Baswana, T. Kavitha, K. Mehlhorn, and S. Pettie. Additive spanners and (α, β)-spanners. *ACM Transactions on Algorithms.* to appear.

[BKMP05] Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. New constructions of (α, β)-spanners and purely additive spanners. In *Proc. 16th Ann. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 672–681, 2005.

A Notation

$\delta_G(u, v)$		length of a shortest path between u and v in G
$\delta_G(X, Y)$	$=$	$\min_{x \in X, y \in Y} \delta_G(x, y)$
$\mathrm{diam}_G(D)$	$=$	$\max_{x,y \in D} \delta_G(x, y)$
$\Gamma_G(v)$		neighbourhood of the vertex v in G
$G[S]$		subgraph of G induced by the vertex set S
$P_G(u, v) = \langle u, \ldots, v \rangle$		a shortest path between u and v in G
$\mathcal{P}_G$		set of $\binom{n}{2}$ shortest paths between all vertex pairs in G
cluster C		set of vertices
clustering $\mathcal{C}$		set of disjoint clusters
$\mathcal{C}(v)$		cluster of clustering $\mathcal{C}$ that contains v
$\mathcal{C}(D)$	$=$	$\{\mathcal{C}(v) \mid v \in D\}$
$\mathcal{E}_{(V,E)}(X, Y)$	$=$	$(X \times Y) \cap E$
$\mathcal{E}_{(V,E)}(x, Y)$	$=$	$(\{x\} \times Y) \cap E$
$\mathrm{value}_H(D)$	$=$	$\lvert\{\{C, C'\} \subseteq \mathcal{C}(D) \mid \delta_D(C, C') \leq \delta_H(C, C')\}\rvert$
$\mathrm{cost}_H(D)$	$=$	$\lvert D \setminus H \rvert$