Well-separated pair decomposition and t-spanners

Frédérik Paradis - University of Ottawa - fpara058@uottawa.ca

November 2, 2015

1 Introduction

A *well-separated pair decomposition* (WSDP) of a point set $S \subseteq \mathbb{R}^d$ is a set of pairs $\{\{A_1, B_1\}, \{A_2, B_2\}, \ldots, \{A_m, B_m\}\}$ such that for any distinct point p and q in S, there is a unique pair $\{A_i, B_i\}$, $1 \leq i \leq m$, so that $p \in A_i$ and $q \in B_i$ or $p \in B_i$ and $q \in A_i$, and for each pair $\{A_i, B_i\}$, $1 \leq i \leq m$, A_i and B_i are well-separated. A_i and B_i are well-separated if and only if there are two circles (d-balls) with the same radius ρ respectively enclosing A_i and B_i and the minimum distance between the circles is $s\rho$ where s is called the separation ratio which is greater than 0. The intuition is that the pair forms a cluster of two components. See Figure 1 for an illustration.

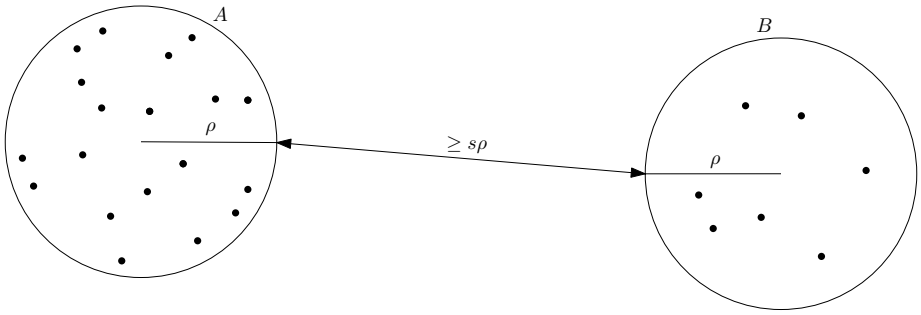

Figure 1: Visual representation of well-separated pair

In the purpose of finding well-separated pairs, it is necessary to define the concept of *bounding box*. A bounding box $R(S)$ of a point set S is the smallest axes-parallel d-dimensional hyperrectangle containing S. An axes-parallel d-dimensional hyperrectangle, or for short a hyperrectangle, is defined as the Cartesian product of closed interval for each dimension. For a hyperrectangle R, we have $R = [l_1, r_1] \times [l_2, r_2] \times \ldots \times [l_d, r_d]$ where $l_i \leq r_i$. The length of the side in the dimension i is given by $L_i(R) = r_i - l_i$. L_{max} and L_{min} define respectively the longest and the shortest length for all dimension. Finally, a d-cube is a hyperrectangle such that the length of the side on each dimension is the same.

2 The split tree

To construct a WSPD, we need to preprocess the points and obtain a data structure that will be used to generate the WSPD. This data structure is the split tree that Callahan and Kosaraju used in their original paper [3]. Note that it is possible to use other data structures like a quad tree as

used by Har-peled in [6]. We will show that it is possible to construct it in $O(n \log(n))$ time after having shown a simpler $O(n^2)$-time algorithm.

The general principle of the split tree is that each node u of the tree represents a set of points S_u and that the bounding box $R(S_u)$ of S_u is split along its longest side in two equal parts which form the two children of u and their point set. It is done recursively until there is only one point in the set. Here is the algorithm:

Algorithm 1 A basic algorithm for the split tree

function SplitTree(S)
  Let u be the node for S
  if $|S| = 1$ **then**
    $R(u) := R(S)$ ▷ $R(S)$ is a hyperrectangle which each side has a length of zero.
    Store in u the only point in S.
  else
    Compute $R(S)$
    Let the i-th dimension be the one where $L_{max}(R(s)) = L_i(R(S))$
    Split $R(S)$ along the i-th dimension in two same-size hyperrectangles and take the points contained in these hyperrectangles to form the two sets S_v and S_w.
    $v :=$ SplitTree(S_v)
    $w :=$ SplitTree(S_w)
    Store v and w as, respectively, the left and right children of u.
    $R(u) := R(S)$
  end if
  return u
end function

For each node, the algorithm stores the bounding box of his associated point set and his two child nodes or the only associated point if it is a leaf. A split tree has the n points as leaves and, thus, has $2n - 1$ nodes. Because there can exist point positions where there can be $O(n)$ splits which can cost $O(n)$ for each split, this give us a worst case time of $O(n^2)$ and, thus, the split tree can be very unbalanced.

Now that we have an algorithm for the split tree, it is time to find a more efficient one. So, the goal is to loop over the list in only $O(n)$ operations per step of the recursion but only call the recursion on at most $n/2$ points each time. To do that, it is necessary to have sorted lists of the points for each dimension. Once the dimension of the longest side is identified, simply walk along the point list from the beginning and the end simultaneously and stop when one of the sides reaches the hyperplane that separates the hyperrectangle in two. Then, call the recursion on the side that reached the hyperplane. For the other side, repeat this until a call to the recursion can be made with a point set of size lesser than or equal to $n/2$. Because each call to the recursion involves less than or equal to $n/2$ points, this algorithm should run in $O(n \log(n))$.

Let S_i^j be the j-th coordinate of the i-th point in S such that S is sorted for each dimension and $p(S_i^j)$ be the point. Also, let $h(R(S)) = (l_i + r_i)/2$, where i is the dimension where $L_{max}(R(S)) = L_i(R(S))$, be the hyperplane that splits the longest side of $R(S)$ in two. Here is the algorithm in pseudo-code:

Algorithm 2 An $O(n \log(n))$ algorithm for the split tree

function SplitTree(S, u)
  if $|S| = 1$ **then**
    $R(u) := R(S)$ ▷ $R(S)$ is a hyperrectangle which each side has a length of zero.
    Store in u the only point in S.
  else
    $size := |S|$
    repeat
      Compute $R(S)$
      $R(u) := R(S)$
      $j := 1$
      $k := |S|$
      Let the i-th dimension be the one where $L_{max}(R(S)) = L_i(R(S))$
      $S_v := \emptyset$
      $S_w := \emptyset$
      while $S_i^{j+1} < h(R(S))$ **and** $S_i^{k-1} > h(R(S))$ **do**
        $size := size - 1$
        $S_v := S_v \cup \{p(S_i^j)\}$
        $S_w := S_w \cup \{p(S_i^k)\}$
        $j := j + 1$
        $k := k - 1$
      end while
      Let v and w be respectively, the left and right children of u.
      if $S_i^{j+1} > h(R(S))$ **then**
        $S_w := S \setminus S_v$
        $u := w$
        $S := S_w$
        SplitTree(S_v, v)
      else if $S_i^{k-1} < h(R(S))$ **then**
        $S_v := S \setminus S_w$
        $u := v$
        $S := S_v$
        SplitTree(S_w, w)
      end if
    until $size \leq \frac{n}{2}$
    SplitTree(S, u)
  end if
end function

To be able to maintain the sorted lists for each node, linked lists are used. Cross-pointers are kept for each list to the others to be able to retrieve a point in constant time. In the algorithm above, in each iteration of the loop, a call to the recursion is done. In reality, to be able to reconstruct the list without the overhead of resorting the points, it is necessary to rebuild the sorted lists once all points have been assigned to their nodes. To do the rebuilding, walk along each list for each dimension, add each point to the corresponding list of its nodes, and add cross-pointers in the

original list to be able to add the cross-pointers for the new lists. Finally, call the recursion on each node and his set.

3 The WSPD computation

Now that we have the split tree, we can compute the WSPD. The two following functions are used to compute the WSPD.

Algorithm 3 From two nodes u and v in the split tree that have disjoint point sets, this algorithm finds well-separated pairs of sets so that, for each pair $\{A, B\}$, $A \subseteq S_v$ and $B \subseteq S_w$.

```
function FindPairs(v, w)
    if S_v and S_w are well-separated with respect to s then
        return The set {{u, v}}
    else if L_max(R(v)) ≤ L_max(R(w)) then
        Let w_l and w_r be respectively the left and right children of w.
        WSPD := ∅
        WSPD := WSPD ∪ FindPairs(v, w_l)
        WSPD := WSPD ∪ FindPairs(v, w_r)
        return WSPD
    else
        Let v_l and v_r be respectively the left and right children of v.
        WSPD := ∅
        WSPD := WSPD ∪ FindPairs(v_l, w)
        WSPD := WSPD ∪ FindPairs(v_r, w)
        return WSPD
    end if
end function
```

This algorithm will always terminate because two points are always well-separated and the algorithm does the recursion branch where the bigger bounding box will be split. To compute the circles separating two point sets, the bounding boxes are used. See Figure 2 for an example.

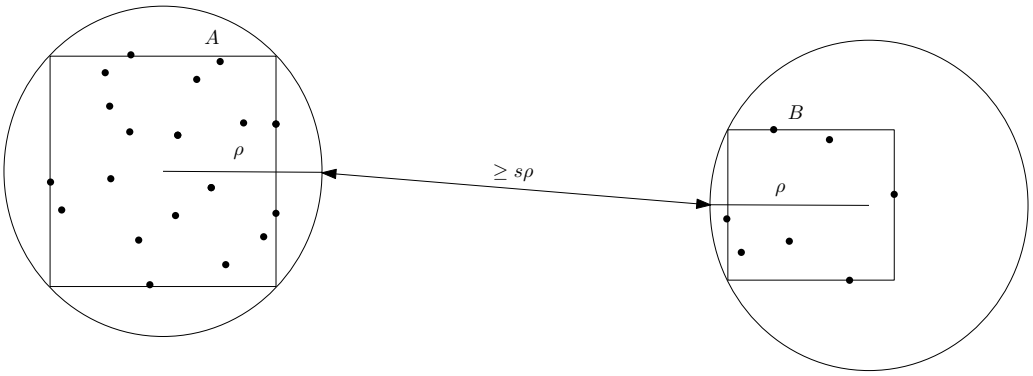

Figure 2: Visual representation of a well-separated pair computed with the bounding boxes

Algorithm 4 From the split tree, a WSPD with respect to s is found.

```
function FindWSPD(T, s)
    WSPD := ∅
    for each node u that is not a leaf in the split tree T do
        Let v and w be respectively the left and right children of u.
        WSPD := WSPD ∪ FindPairs(v, w)
    end for
    return WSPD
end function
```

Proof of correctness. It is clear that the pairs returned by the algorithm are well-separated because of the return condition of the function FindPairs.

Now, we have to prove that for any distinct points p and q in S, there is a unique pair $\{A, B\}$ so that (i) $p \in A$ and $q \in B$ or (ii) $p \in B$ and $q \in A$. Assume without loss of generality that (i) holds.

Let u be the lowest common ancestor of p and q in the split tree and let v and w be the children of u. Because of the last assumption, p is in the subtree of v and q in the subtree of w. A call to FindPairs(v, w) is necessarily done in FindWSPD. Because, each time there is a recursion, the recursion tree creates two branches that contain all the points of the current recursion call, there will be a sequence of call to FindPairs leading to having p in A and q in B.

Because u is the lowest common ancestor of p and q, calling FindPairs on the children of a higher node would result of p and q not being in a pair and calling FindPairs on the children in one of the nodes of one of the subtrees of u would result by p or q not being in any pair. Thus, the pair $\{A, B\}$ is the unique one separating p and q. □

Running time analysis. Each time the recursion tree split in two, there is one more pair added to the decomposition. So, the algorithm run-time is in the number of pairs in the final decomposition. □

Now that we have an algorithm to compute a WSPD, it is time to verify the size of the computed WSPD. Because there can be $\binom{n}{2}$ pairs in all, it is important to have a bound asymptotically lesser. So, we have the following theorem:

Theorem 1. *The algorithm* FindWSPD *find a Well-separated pair decomposition (WSPD) of size* $O(s^d n)$.

Callahan and Kosaraju proved this in their original paper [3]. The proof is very technical and long, so we will not do it here. However, the idea of the proof is that a set A cannot be in too many pair $\{A, B\}$, and thus, the total number of pairs is bounded linearly to what the theorem 1 says.

4 t-spanners

4.1 Construction and proof

One of the applications of the WSPD is the t-spanners. A t-spanner of the point set S is a graph such that S is the vertex set of the graph and that the euclidean length of a path between two points p and q in S is less than or equal to $t|pq|$ where $|pq|$ is the euclidean distance between p and

q. Of course, the complete graph satisfies this definition but is not very interesting because of his great number of edges.

Consider the following construction. Construct the graph with the point set S as vertex set and for each pair $\{A, B\}$ in a WSPD, add an edge from an arbitrary point $a \in A$ to an arbitrary point $b \in B$. Note that the resulting graph has a linear number of edges. What would be interesting with this graph is if it could be proven that it is a t-spanner for some constant t in function of s. That is what we are going to show. But, before that, we need two simple lemmas.

Lemma 1. *Let $\{A, B\}$ be a well-separated pair with respect to s. Let $p, p' \in A$ and $q \in B$. Then, $|pp'| \leq (2/s)|pq|$.*

Proof. Because p and p' are in the same set, we have that $|pp'| \leq 2\rho$ where ρ is the radius of the enclosing circle of A and B. Because p and q are in two well-separated sets, we have that $|pq| \geq s\rho$. We obtain that:

$$\begin{gathered}\frac{|pp'|}{2} \leq \rho \leq \frac{|pq|}{s} \\ \Leftrightarrow \\ \frac{|pp'|}{2} \leq \frac{|pq|}{s} \\ \Leftrightarrow \\ |pp'| \leq \frac{2}{s}|pq|\end{gathered}$$

□

Lemma 2. *Let $\{A, B\}$ be a well-separated pair with respect to s. Let $p, p' \in A$ and $q, q' \in B$. Then, $|p'q'| \leq (1 + 4/s)|pq|$.*

Proof. By the triangle inequality, we have:

$$|p'q'| \leq |p'p| + |pq| + |qq'|$$

From Lemma 1, we obtain:

$$\begin{aligned}|p'q'| &\leq (2/s)|pq| + |pq| + (2/s)|pq| \\ &= (1 + 4/s)|pq|\end{aligned}$$

□

Now, it is time to prove that adding an arbitrary edge for each pair in the WSPD produces a t-spanner for a point set S.

Proof. Let $\{A, B\}$ be a well-separated pair with respect to s in a WSPD. Let $[a, b]$ be the edge connecting A to B. Let any point $p \in A$ and $q \in B$. Because of the definition of the WSPD, it is only necessary to prove that there is a t-spanner path, or t-path for short, between p and q noted P_{pq}. Let the length of the path P_{pq} be noted $|P_{pq}|$.

Suppose there is a t-path between any pair of points which distance is less than or equal to $|pq|$ and that $s > 2$. From the triangle inequality, the assumptions and the fact that $|pa| \leq (2/s)|pq| \Rightarrow |pa| < |pq|$ and $|bq| \leq (2/s)|pq| \Rightarrow |bq| < |pq|$ according to Lemma 1, we have:

$$|P_{pq}| \leq t|pa| + |ab| + t|bq|$$

From Lemma 1 and 2, we obtain:

$$\begin{aligned}
|P_{pq}| &\leq t(2/s)|pq| + (1 + 4/s)|pq| + t(2/s)|pq| \\
&= t(4/s)|pq| + (1 + 4/s)|pq| \\
&= \left(\frac{4t + 4}{s}\right)|pq| + |pq| \\
&= \left(\frac{4(t + 1)}{s} + 1\right)|pq|
\end{aligned}$$

So what we want is:

$$\begin{aligned}
t &= \frac{4(t + 1)}{s} + 1 \\
t - 1 &= \frac{4(t + 1)}{s} \\
s(t - 1) &= 4(t + 1) \\
s(t - 1) - 4(t + 1) &= 0 \\
st - s - 4t - 4 &= 0 \\
t(s - 4) - s - 4 &= 0 \\
t(s - 4) &= s + 4 \\
t &= \frac{s + 4}{s - 4}
\end{aligned}$$

So, if $t = (s + 4)/(s - 4)$ and $s > 4$, then we have a t-spanner for the point set S. □

Finally, it is possible to choose s from t by isolating s from $t = (s + 4)/(s - 4)$ which gives $s = 4(t + 1)/(t - 1)$.

4.2 t-spanners with bounded diameter

The diameter of a graph is the minimum number of edges for which it is possible to have a path from any point to any other point in the graph. This quality is often desirable in many context. In this section, we will show that it is possible to construct a t-spanner that have a bounded diameter.

When constructing a t-spanner with a WSPD, a choice of edge for each pair is done arbitrarily. The two endpoints of the chosen edge are called representatives of their respective point set. However, if this choice is done by an informed decision, it is possible to construct a t-spanner such that the diameter of the graph is bounded.

The construction of the t-spanner with a bounded diameter uses the split tree. For each node of the split tree, the edge leading to a child with a bigger subtree is called heavy edge and the other one is called light edge. Then, the representative of each point set associated to a node is the point associated to the leaf reachable with a chain containing only heavy edges.

Let G be a t-spanner formed from a WSPD with this construction. Let any well-separated pair $\{A, B\}$. Let us prove this claim:

Claim 1. *Suppose that from any point in A to the representative of A, there is a path containing at most $\lg(n)$ edges in G. Suppose the same for B. So, the diameter of a path P_{pq} between $p \in A$ and $q \in B$ is at most $2\lg(n) - 1$.*

Proof. Because the logarithmic function is an increasing function and that $\lg(n/2) + \lg(n/2) \geq \lg(n-1)$ (the right side is the case where all but one point are on one side), the worse case is when $|A| = |B| = n/2$. Then, the diameter of the path P_{pq} noted $||P_{pq}||$ is:

$$\begin{aligned} ||P_{pq}|| &\leq \lg(n/2) + \lg(n/2) + 1 \\ &= \lg(n) - \lg(2) + \lg(n) - \lg(2) + 1 \\ &= 2\lg(n) - 1 \end{aligned}$$

□

Now, it is time to prove the prerequisite of this claim to prove that there exists a path between any two points having at most $2\lg(n) - 1$ edges given the construction.

Theorem 2. *From any point $p \in A$ to the representative of $a \in A$, there is a path containing at most $\lg(n)$ edges in G.*

Proof. Let the pair $\{C, D\}$ be the one where $p \in C$ and $a \in D$ without loss of generality. This proof is by induction on the size of A.

Because a is the representative of A, this means that there is a chain of heavy edges from the node of a in the split tree to the node of A. Because a is also in D, this means that the node of D is an ancestor of the node of a. Then, because D is subset of A or else p would also be in D, the chain of heavy edges in which D is leads to the node of a and, thus, a is also the representative of D.

Because C and D are disjoint, C is not in the same heavy chain of A and, thus, contains at most $|A|/2$ points. By induction, the number of edges on the path from p to the representative of C is at most $\lg(|A|/2) \leq \lg(n) - 1$. By adding the edge from C to D, a path from p to a having at most $\lg(n)$ edges is obtained. □

4.3 t-spanners with other specific properties

When constructing a t-spanner, arbitrary points are chosen to form an edge for each pair $\{A, B\}$. It is possible to choose these representatives so that the resulting t-spanner has specific properties.

In [2], Callahan and Kosaraju gives many constructions that give t-spanners with specific properties. One of the constructions is that it is possible to have a t-spanner such that each vertex degree is bounded by constant that only depends on separation ratio s and the number of dimension d. Another construction is that it is possible to have a t-spanner such that the diameter of the graph is equal to an arbitrary constant. Such t-spanner can have high degree. It is also possible to construct a t-spanner such that its degree is bounded and its diameter is bounded by $O(\log n)$.

5 Bibliographic Notes

The definition of the well-separated pair decomposition (WSPD) in section 1 Introduction was taken from one text of Smid [7] and one from Narasimhan and Smid [5] but the WSPD was originally defined by Callahan and Kosaraju in [3]. Sections 2 The split tree and 3 The WSPD computation are from Callahan and Kosaraju in [3]. The proof of the t-spanner from a WSPD in section 4 t-spanners is from Callahan and Kosaraju in [4] but was taken in [7]. Section 4.2 t-spanners with bounded diameter is from Arya, Mount and Smid in [1].

6 Exercises

1. Give an example of point positions that would lead to a computation time of n^2 for the split tree with the naive algorithm (Algorithm 1) described in this text. What does the split tree look like?

2. Give the algorithm to compute if two point sets are well-separated by using the bounding boxes.

3. Give an intuition of why the choice for the condition in the FindPairs algorithm.

4. Simple algebraic calculation: Isolate s in $t = (s + 4)/(s - 4)$.

5. Give the base cases for each induction proof in this text.

References

[1] S. Arya, D.M. Mount, and M. Smid. Randomized and deterministic algorithms for geometric spanners of small diameter. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 703–712, Nov 1994.

[2] Sunil Arya, Gautam Dast, David M. Mount, Jeffrey S. Salowe, and Michiel H. M. Smid. Euclidean spanners: short, thin, and lanky. In *ACM Symposium on Theory of Computing*, pages 489–498, 1995.

[3] Paul B. Callahan and S. Rao Kosaraju. A decomposition of multi-dimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42:546–556, 1992.

[4] Paul B. Callahan and S. Rao Kosaraju. Faster algorithms for some geometric graph problems in higher dimensions. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 291–300, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[5] Michiel Smid Giri Narasimhan. *Geometric Spanner Networks*. Cambridge University Press, 2007.

[6] Sariel Har-peled. *Geometric Approximation Algorithms*. Mathematical Surveys and Monographs. American Mathematical Society, 2011.

[7] Michiel Smid. The well-separated pair decomposition and its applications. In Teofilo Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, pages 53–1 – 53–12. Chapman & Hall/CRC, Boca Raton, 2007.