Prune and Search Technique in Computational Geometry

Swami Sarvottamananda

Ramakrishna Mission Vivekananda University

IISCTIFR-IGGA, 2009

Outline I

Introduction to Prune & Search

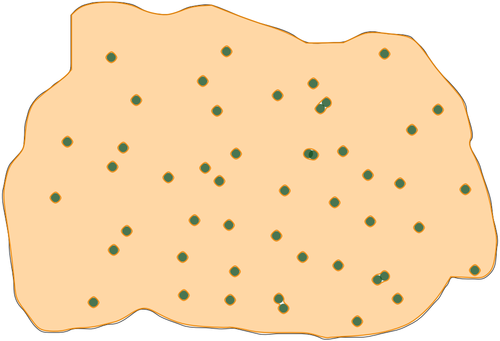
- What is *Prune and Search*?

Introduction to Prune & Search

- What is *Prune and Search*?
- Prune and Search is a technique of successively removing a subset of input without changing the solution (See [1, 9, 4]).

Prune & Search Technique—0

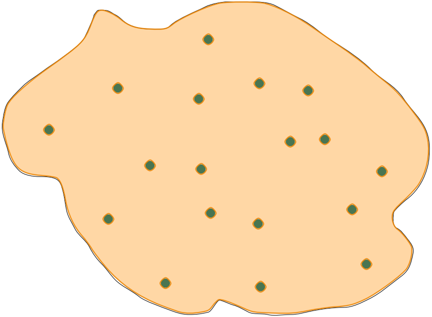
Example (Prune & Search)



We have a lot of data as input.

Prune & Search Technique—1

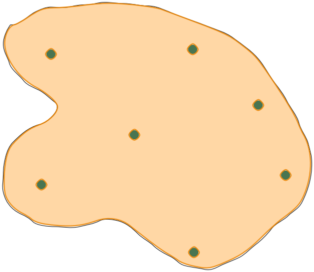
Example (Prune & Search)



We prune a subset at every step.

Prune & Search Technique—2

Example (Prune & Search)



Till we have only a small set.

Where and How

We need to know — where and how prune and search technique is applied?

Where and How

We need to know — where and how prune and search technique is applied?

- Where? It is applied wherever the solution space ultimately is determined by a small set of data. We shall call it *significant* data. Rest of data is extra, *spurious*.

Where and How

We need to know — where and how prune and search technique is applied?

- Where? It is applied wherever the solution space ultimately is determined by a small set of data. We shall call it *significant* data. Rest of data is extra, *spurious*.
- How? Easy! Instead of trying to locate the *significant* data directly, we go the other way round. We try to locate the *spurious* data and remove it.

Outline of Algorithm

Input : Set S of n objects
Output : $x^* = F(S)$

- Find prunable set $S' \subset S$ such that $F(S - S') = F(S)$

Outline of Algorithm

Input : Set S of n objects
Output : $x^* = F(S)$

- Find prunable set $S' \subset S$ such that $F(S - S') = F(S)$
- If $|S'| > 0$ then remove S' from S and recurse

Outline of Algorithm

Input : Set S of n objects
Output : $x^* = F(S)$

- Find prunable set $S' \subset S$ such that $F(S - S') = F(S)$
- If $|S'| > 0$ then remove S' from S and recurse
- Else find $F(S)$ directly

Outline of Algorithm

Input : Set S of n objects
Output : $x^* = F(S)$

- Find prunable set $S' \subset S$ such that $F(S - S') = F(S)$
- If $|S'| > 0$ then remove S' from S and recurse
- Else find $F(S)$ directly

There might be some modifications to the theme without affecting the overall computational complexity of the algorithm.

Outline of Algorithm

Input : Set S of n objects
Output : $x^* \in F(S)$

- Find prunable set $S', S'' \subset S$ such that $F((S - S') \cup S'') \subseteq F(S)$

Outline of Algorithm

Input : Set S of n objects
Output : $x^* \in F(S)$

- Find prunable set $S', S'' \subset S$ such that $F((S - S') \cup S'') \subseteq F(S)$
- If $|(S - S') \cup S''| < f|S|$ then remove S' from S, add S'' to S and recurse

Outline of Algorithm

Input : Set S of n objects
Output : $x^* \in F(S)$

- Find prunable set $S', S'' \subset S$ such that $F((S - S') \cup S'') \subseteq F(S)$
- If $|(S - S') \cup S''| < f|S|$ then remove S' from S, add S'' to S and recurse
- Else find $F(S)$ directly

Outline of Algorithm

Input : Set S of n objects
Output : $x^* \in F(S)$

- Find prunable set $S', S'' \subset S$ such that $F((S - S') \cup S'') \subseteq F(S)$
- If $|(S - S') \cup S''| < f|S|$ then remove S' from S, add S'' to S and recurse
- Else find $F(S)$ directly

Note: f is a fixed fraction less than 1.

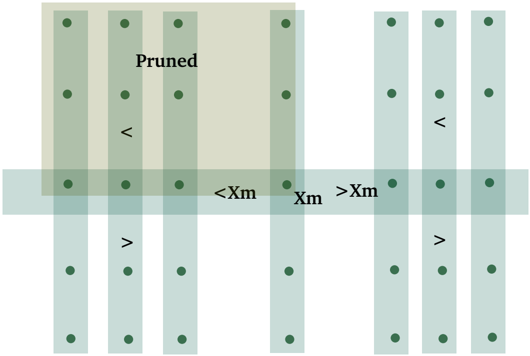
Example Problem

Linear-time Median (actually k-th largest) Finding Algorithm[2]

- Divide n numbers in $n/5$ 5-tuples
- Find medians of each of $n/5$ 5-tuples
- Find the median of $n/5$ medians of 5-tuples *recursively*
- Check if the median of medians is k-th largest
- If not then drop at least $n/4$ points (either smaller or larger) and recurse

Pruning in the case of Median

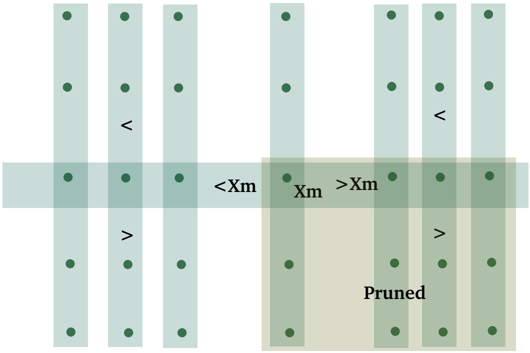
Example (Median)



We can prune the upper left quadrant if x_m is less than the k-th largest number.

Pruning in the case of Median — the Other Case

Example (Median)



We can prune the lower right quadrant if x_m is greater than the k-th largest number.

Time-complexity of Median Algorithm

- Why is the algorithm $O(n)$?

Time-complexity of Median Algorithm

- Why is the algorithm $O(n)$?
- $T(n) \leq T(3n/4) + T(n/5) + O(n)$

Time-complexity of Median Algorithm

- Why is the algorithm $O(n)$?
- $T(n) \leq T(3n/4) + T(n/5) + O(n)$
- Solution of above recurrence is $T(n) = O(n)$

When can Prune & Search be applied

Prune & Search is applied when

- (Obviously) If the problem is a *search* problem.
  Usually it means we want to search for a vector x^*.

When can Prune & Search be applied

Prune & Search is applied when

- (Obviously) If the problem is a *search* problem.
  Usually it means we want to search for a vector x^*.
- Final solution must be determined by only a few of the data.
  Most of the optimisation problems search for a value that in the final analysis is determined only by a small subset of constraints.

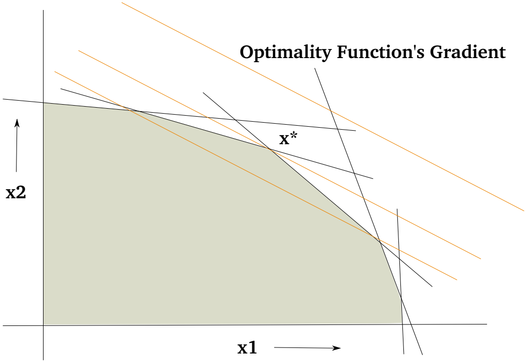
When can Prune & Search be applied

Prune & Search is applied when

- (Obviously) If the problem is a *search* problem.
  Usually it means we want to search for a vector x^*.
- Final solution must be determined by only a few of the data.
  Most of the optimisation problems search for a value that in the final analysis is determined only by a small subset of constraints.
- The solution space must not change after pruning. (Very important!)

LPP in 2-dimensions

Example (LPP)



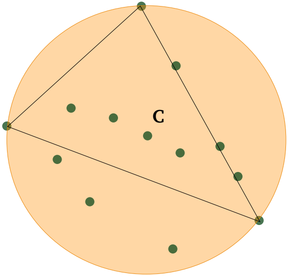
Final solution is determined only by two constraints.

Case of Minimum Enclosing Circle

If we pose the minimum enclosing circle in E^2 problem as an optimisation problem then the centre of minimum enclosing circle is determined by only three circumference points.

Minimum Enclosing Circle in 2-dimensions

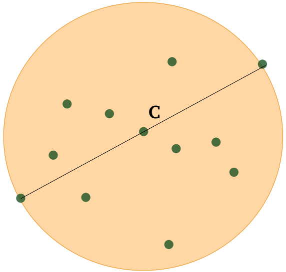
Example (Min Enc Circle)



Final solution is determined only by three circumference points.

Minimum Enclosing Circle in 2-dimensions - II

Example (Min Enc Circle)



or two circumference points!

Problem Definition

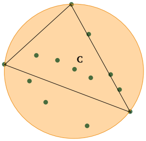
Problem Definition: Given a finite set of points in plane find the minimum radius circle such that all points lie inside the circle.

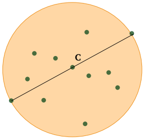
What is in the name?

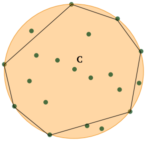
Other names of the problem are minimum stabbing disk problem, minimum enclosing ball or intersection radius problem (See [10, 7, 4, 3, 9, 6].

Example of Minimum Enclosing Circle

Example (Several Cases of Minimum Enclosing Circles)







In every case the centre of minimum enclosing circle lies inside the convex hull of the points on the circumference.

Example of Minimum Enclosing Circle

We are usually interested in stabbing circles of convex shapes only. Anything non-convex gives rise to bad cases and algorithms whose worst case performance is not linear.

Min Enc Circle as an Optimisation Problem

Non-linear Programming Problem of three variables

$$\begin{gathered}\texttt{Min } r \\ \text{Subject to } (x_i - x)^2 + (y_i - y)^2 \leq r^2 \\ x,\ y \text{ unconstrained}\end{gathered}$$

This is an optimisation problem in 3-variables. We can also write it in an equivalent optimisation problem of 2-variables.

Min Enc Circle as an Optimisation Problem - II

Non-linear Programming Problem of two variables

$$\texttt{Min Max}_i \sqrt{(x_i - x)^2 + (y_i - y)^2}$$
$$x,\ y \text{ unconstrained}$$

In effect, *minimise the maximum distance to any point from centre.*

Min-Max Optimisation Problem

The minimum enclosing circle computation problem belongs to a class of problems known as *Min-Max optimisation problems*.
Interestingly, computation problems of minimum stabbing circle for line segments or convex polygons are *min-max-min optimisation problems*.

Main Idea

- The main idea is that we need to localise the region where the most significant data lies.

Main Idea

- The main idea is that we need to localise the region where the most significant data lies.
- The significant data depends on the problem. Sometimes it may be somewhere near the centre and sometimes it might be near the periphery.

Main Idea

- The main idea is that we need to localise the region where the most significant data lies.
- The significant data depends on the problem. Sometimes it may be somewhere near the centre and sometimes it might be near the periphery.

  In computation of centre-point the significant data is near the centre. In the case of minimum enclosing circle it is the outer data which is significant (See [5]).

Follow up of Main Idea

- Need to localise the most significant data naturally implies a *need* to localise the final solution
  Note that exact localisation of the solution is not needed, only that much which allows us to prune.

Localisation of Optimal Point — I

- Most of the time we would be searching for an optimal point x^*

Localisation of Optimal Point — I

- Most of the time we would be searching for an optimal point x^*
- In our case it is the centre of minimum enclosing circle

Localisation of Optimal Point — I

- Most of the time we would be searching for an optimal point x^*
- In our case it is the centre of minimum enclosing circle
- Once it is located, other things such as the radius, points on the circumference etc. can be determined in $O(n)$ time

Localisation of Optimal Point — II

- The class of problems where we can easily apply prune and search is where the optimality function is convex.

Localisation of Optimal Point — II

- The class of problems where we can easily apply prune and search is where the optimality function is convex.
- Why? Seems that the requirement is a tall order.

Localisation of Optimal Point — II

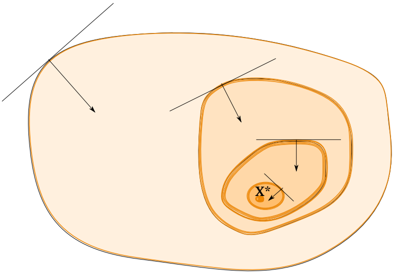
- The class of problems where we can easily apply prune and search is where the optimality function is convex.
- Why? Seems that the requirement is a tall order.
- For convex functions, the gradient allows us to localise the optimal point.

Localisation of Optimal Point — II

- The class of problems where we can easily apply prune and search is where the optimality function is convex.
- Why? Seems that the requirement is a tall order.
- For convex functions, the gradient allows us to localise the optimal point.
  It doesn't look like but there is a corresponding function even for median finding problem which is convex.

Localisation

Example (Localisation)



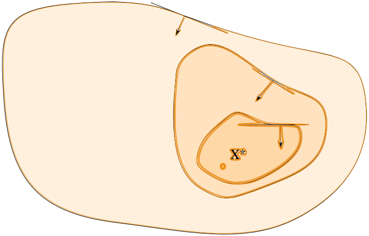
The direction of gradient allows us to locate the region where x^* lies.

Localisation — Explanation

- In minimisation problem we take the opposite direction than the gradient. We do not directly deal with gradients because our functions are not continues and differentiable. They are only peace-wise continuous and differentiable.

Localisation for non-convex optimality criteria

Example (Concave optimality function)



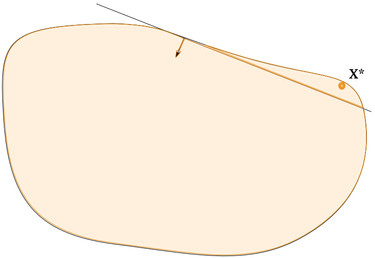
The case of non-convex optimality criteria

Localisation for non-convex optimality criteria — explanation

- The optimality function need not be convex.
- Important thing is to determine the location of x^* with respect to our query object (line, hyper-plane or whatever) reliably.

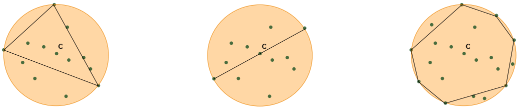
Failure of Localisation

Example (Failure)



We fail to properly localise the optimal point x^*.

Simplest Algorithm for Minimum Enclosing Circle



Check every ${}^nC_3$ 3-combination of points, if the resulting circle contains all the points.

Leads to ${}^nC_3$ times $n = O(n^4)$ algorithm.

Can we do better?

Observations for Not-so-simple Algorithm

- Minimum enclosing circle encloses convex hull

Observations for Not-so-simple Algorithm

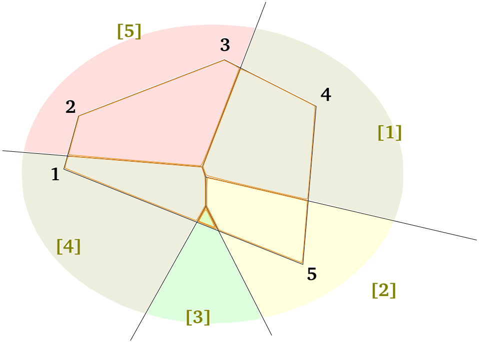
- Minimum enclosing circle encloses convex hull
- The centre of minimum enclosing circle can be found using farthest-point Voronoi diagram. It is the point where the farthest distance is minimum

Observations for Not-so-simple Algorithm

- Minimum enclosing circle encloses convex hull
- The centre of minimum enclosing circle can be found using farthest-point Voronoi diagram. It is the point where the farthest distance is minimum
- Compute only the farthest-point Voronoi diagram of the points on convex hull
  How does it look like?

Farthest-point Voronoi Diagram

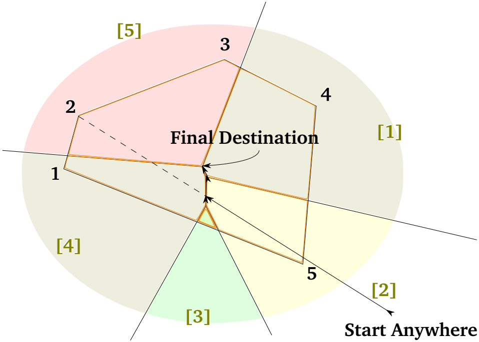
Example (Voronoi Diagram)



How do we use it?

Farthest-point Voronoi Diagram Usage

Example (Using Voronoi Diagram)



Start from a point at ∞ and move towards solution

Complexity of Voronoi Diagram Method

- Construction of both farthest-point voronoi diagram and convex hull takes $O(n \log n)$ time (See [4], one among many references).
- Shamos and Hore erroneously conjectured that this $O(n \log n)$ algorithm is optimal.

Prune & Search applied to Min Enc Circle Problem

We solve the problem in three stages (See [9, 8])

Pr 1 Given the centre how can we determine the MEC centred on it (x, and y given)

We also need to know if the resulting enclosing circle is optimal and if not, where can we localise the optimal point.

Prune & Search applied to Min Enc Circle Problem

We solve the problem in three stages (See [9, 8])

Pr 1 Given the centre how can we determine the MEC centred on it (x, and y given)

Pr 2 (Constrained version) Given a line on which centre of enclosing circle lies how can we determine the constrained MEC ($ax + by + c = 0$).

We also need to know if the resulting enclosing circle is optimal and if not, where can we localise the optimal point.

Prune & Search applied to Min Enc Circle Problem

We solve the problem in three stages (See [9, 8])

Pr 1 Given the centre how can we determine the MEC centred on it (x, and y given)

Pr 2 (Constrained version) Given a line on which centre of enclosing circle lies how can we determine the constrained MEC ($ax + by + c = 0$).

Pr 3 (Unconstrained version) How can we determine the MEC whose centre is unconstrained (x and y unconstrained).

We also need to know if the resulting enclosing circle is optimal and if not, where can we localise the optimal point.

Assumptions

Without loss of generalisation (wlog) we assume

- (For Problem 1) The constrained centre is at origin

Assumptions

Without loss of generalisation (wlog) we assume

- (For Problem 1) The constrained centre is at origin
- (For Problem 2) The constrainted line of centres is the x-axis

Assumptions

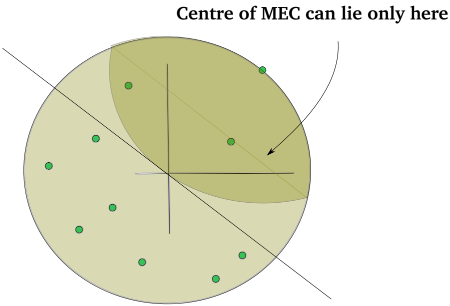
Without loss of generalisation (wlog) we assume

- (For Problem 1) The constrained centre is at origin
- (For Problem 2) The constrainted line of centres is the x-axis

  We can suitable translate the coordinate system to achieve our objective.

Problem 1

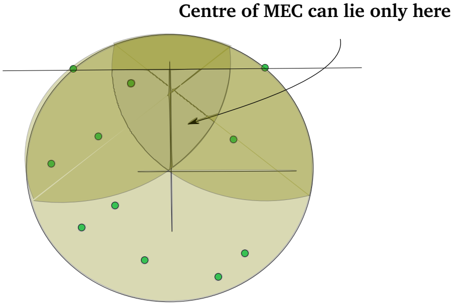
Example (Problem 1)



Easy! Find the maximum distance to points. If more than 1 are there, then determine if they lie around the origin or in some half space.

Problem 1

Example (Problem 1)



In case two points are in circumference, then centre of MEC lies in the shaded region as above.

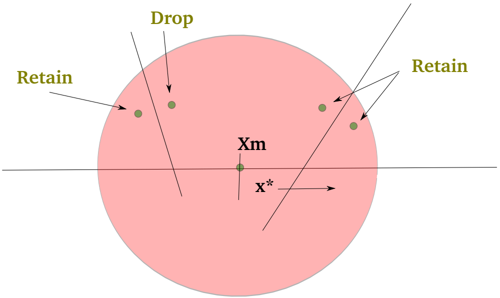
How do we use the solution of Problem 1

Exactly, how do we use the solution we presented in our constrained MEC problem

- We use the solution of Problem 1 to locate x^* on the half line

Problem 2

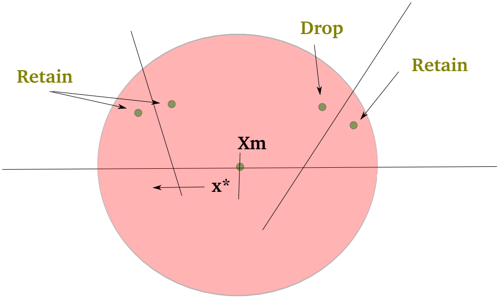
Example (Problem 2)



This is the case when $x^* > x^m$, x_m is the median of intersection of bisectors with x-axis

Problem 2

Example (Problem 2 — Second Case)



And this is when $x^* < x^m$

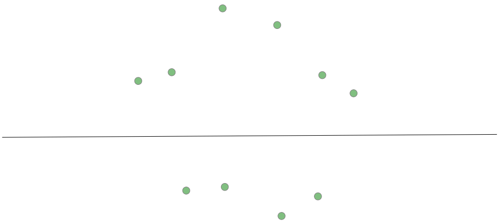
Problem 2 — Algorithm

Algorithm for Constrained MEC

- Pair all the point arbitrarily in $\lfloor n/2 \rfloor$ pairs
- Find the intersections of perpendicular bisectors with x-axis
- Find median x_m of the intersections
- If x_m is not the solution determine the direction of x^* and prune the set
- Recurse

Example for Constrained Problem

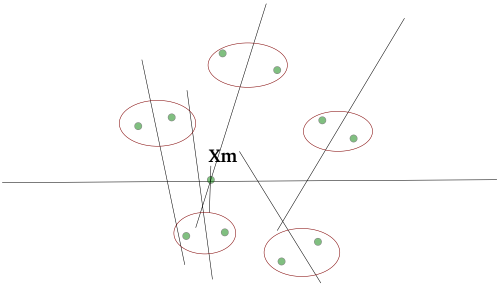
Example (MEC — Step 1)



Input Set

Example for Constrained Problem

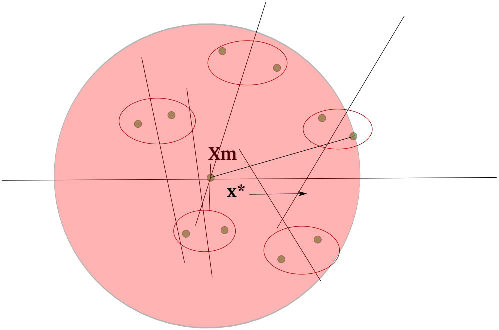
Example (MEC — Step 1)



Arbitrary Pairing

Example for Constrained Problem

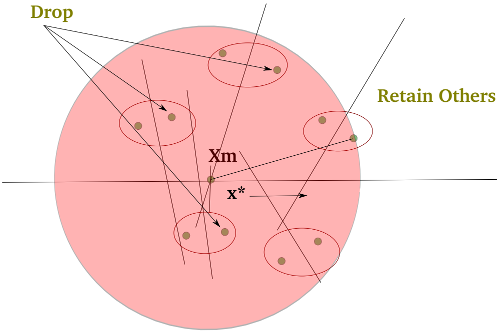
Example (MEC — Step 1)



Enclosing circle centred on median

Example for Constrained Problem

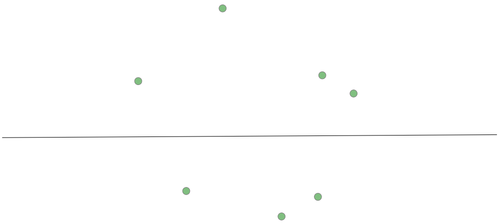
Example (MEC — Step 1)



Decide about pruning

Example for Constrained Problem

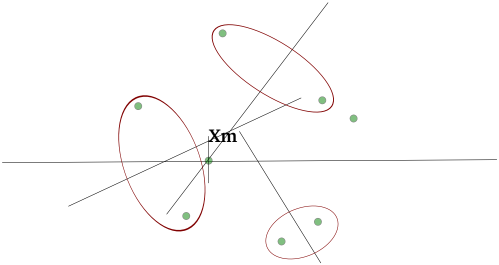
Example (MEC — Step 2)



New pruned set

Example for Constrained Problem

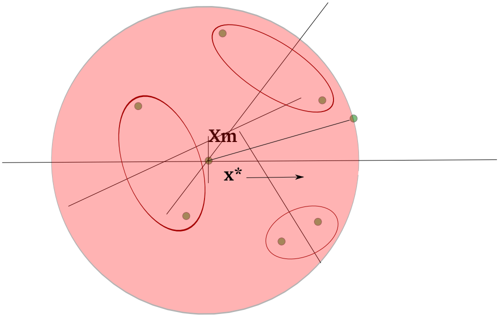
Example (MEC — Step 2)



Arbitrary pairing

Example for Constrained Problem

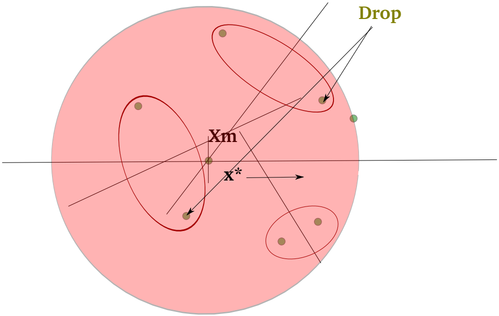
Example (MEC — Step 2)



Enclosing circle centred on median

Example for Constrained Problem

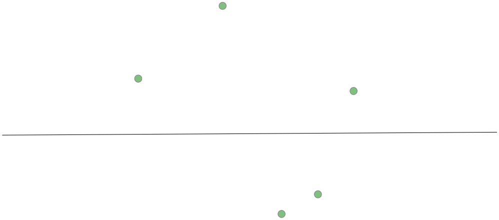
Example (MEC — Step 2)



Decide about pruning

Example for Constrained Problem

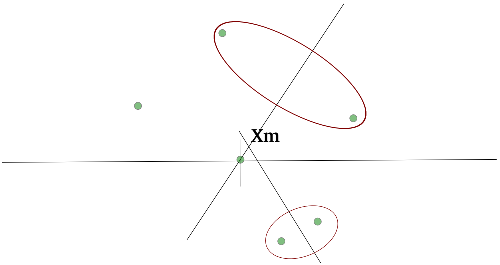
Example (MEC — Step 3)



New pruned set

Example for Constrained Problem

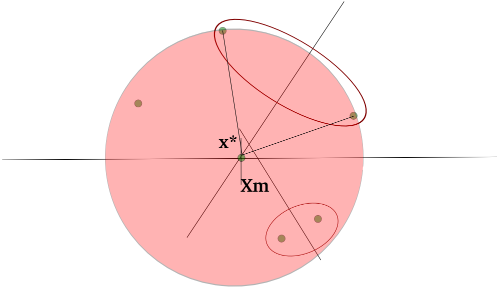
Example (MEC — Step 3)



Arbitrary pairing

Example for Constrained Problem

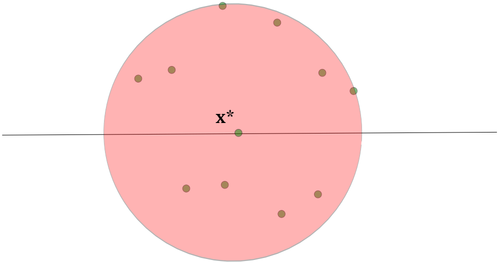
Example (MEC — Step 3)



Enclosing circle which is otimal

Example for Constrained Problem

Example (MEC — Step 3)



Minimum Enclosing Circle

Minimum Enclosing Circle with original set

Complexity of Algorithm for Problem 2

$O(n)$ time because we are able to drop at least $n/4$ points in any case and pruning step is $O(n)$ time

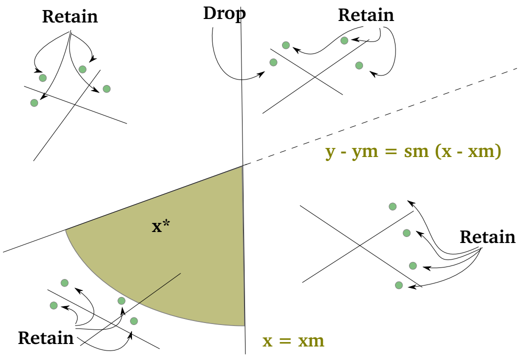
How do we use the solution of Problem 2

Exactly, how do we use the solution we presented in our unconstrained MEC problem

- We use the solution of Problem 2 to locate x^* on some quadrant!!

Problem 3

Example (Problem 3)



x^* is in lower-left quadrant

Problem 3

Algorithm for Unconstrained MEC

- Pair all the point arbitrarily in $\lfloor n/2 \rfloor$ pairs
- Find median slopes s_m of perpendicular bisectors
- Pair bisectors one with $< s_m$ slope and another with $> s_m$ slope, compute intersections
- Find median x_m of intersections and solve constrained MEC for line $x = x_m$
- Wlog let x^* lie on the left
- Drop projections parallel to slope s_m of intersections on the right on the line $x = x_m$, computer median y_m
- Solve constrained MEC for line $y - y_m = s_m(x - x_m)$
- Wlog, let x^* lie on lower left quadrant, prune points on upper right quadrant
- Recurse

Complexity of Algotithm for Problem 3

$O(n)$ time because we are able to drop at least $n/16$ points in any case and pruning step is $O(n)$ time

Extensions of Minimum Enclosing Circle

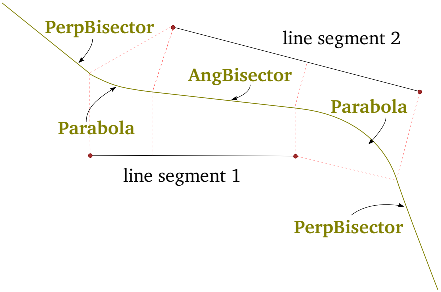
- Minimum enclosing circle problem is well studied in plane. It can be solved optimally for almost every possible kind of convexity satisfying object (See [4, 6]).

Extensions of Minimum Enclosing Circle

- Minimum enclosing circle problem is well studied in plane. It can be solved optimally for almost every possible kind of convexity satisfying object (See [4, 6]).
- Nevertheless, intersection radius is not trivial everywhere.

Intersection Radius for Line Segments

Example (Bisector of two Line segments)



Bisector of line segments is a complex object, not at all convex.

Some Open Problems

- Finding intersection radius of lines in E^3 optimally (linear?)
- Finding intersection radius of convex polyhedra optimally (linear?)
- Finding minimum volume ellipsoids enclosing or stabbing different classes of objects. The case of points and hyper-planes is solved. (linear?)
- Finding 2, 3 or k enclosing/stabbing circles such that maximum radius of these circles is minimum (or their sum is minimum). ($o(n^k)$-time?)

Summary

- We explored the notion of prune and search
- Next, we computer minimum enclosing circle using prune and search technique
- Lastly, we sketched how prune and search can be extended

References I

Pankaj K. Agarwal and Micha Sharir.
Efficient algorithms for geometric optimization.
ACM Comput. Surv., 30(4):412–458, 1998.

Manuel Blum, Robert W. Floyd, Vaughan R. Pratt, Ronald L. Rivest, and Robert E Tarjan.
Two papers on the selection problem: Time bounds for selection [by Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan] and expected time bounds for selection [by Robert W. Floyd and Ronald L. Rivest].
Technical report, Stanford, CA, USA, 1973.

Herbert Edelsbrunner.
Algorithms in Combinatorial Geometry.
Springer-Verlag, New York, 1987.

References II

Jacob E. Goodman and Joseph O'Rourke, editors.
Handbook of Discrete and Computational Geometry.
Chapman & Hall/CRC, second edition, 2004.

S. Jadhav and A. Mukhopadhyay.
Computing a centerpoint of a finite planar set of points in linear time.
Discrete Comput. Geom., 12:291–312, 1994.

Shreesh Jadhav, Asish Mukhopadhyay, and Binay K. Bhattacharya.
An optimal algorithm for the intersection radius of a set of convex polygons.
J. Algorithms, 20(2):244–267, 1996.

J. Matousek.
Lectures in Discrete Geometry.
Springer-Verlag, New York, 2002.

References III

Nimrod Megiddo.
Linear-time algorithms for linear programming in R^3 and related problems.
In *SFCS '82: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 329–338, Washington, DC, USA, 1982. IEEE Computer Society.

Nimrod Megiddo.
Linear programming in linear time when the dimension is fixed.
J. ACM, 31(1):114–127, 1984.

J. Pach and P. K. Agrawal.
Computational Geometry.
John Wiley & Sons, New York, 1995.

At Last . . .

Thank You

shreesh@rkmvu.ac.in
sarvottamananda@gmail.com