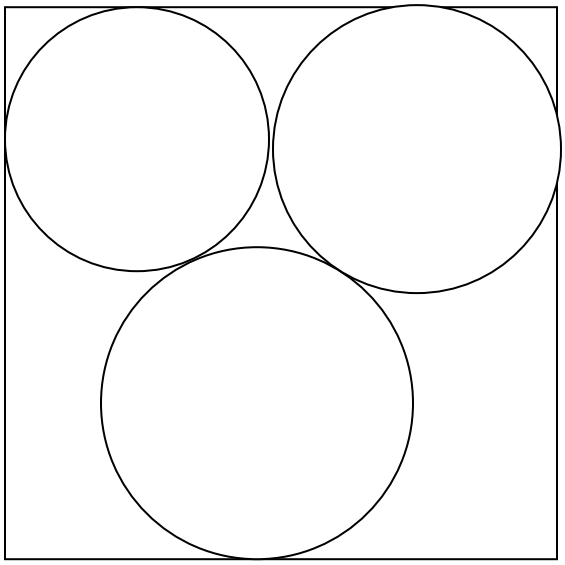
Geometric Constraints – Introduction.

OverOverView on Geometric Constraints

Orientation:

Let's consider a square box of side 'a' (2d). Now imagine that we want to draw 3 circles in this square without the circles overlapping each other or the boundaries of the square. A desirable configuration would be something like this.



Basically what we want is to make these 3 circles as big as possible while still making them fit into the box. That would translate to saying maximize $\sum r_i$ where r_i denotes the radii of the circles.

Generally:
Pack(without overlapping) a disk D in 2-d with given radius d with disks D1,D2,.. Dn of radius x1,x2, ..,xn s.t. $\sum xi$ is maximized.

Task 1: Can this be expressed as a LP problem in the xi? (You are allowed to add more variables.)

Task 1': If no (answer to Task 1), express as simply as possibly. (Please note "simply" here. We'll discuss it in detail later.)

Task 1: Ideally we would like to capture all the constraints that have been implicitly expressed in the above diagram and description in the form of equations. Write out one such formulation. (Hint: Linear equations may not always work out). You are free to add as many variables as necessary.

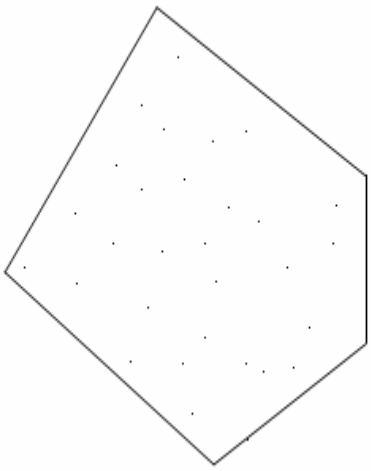
The problem that was just described is known as a packing problem. And it is an example of a geometric constraints problem. We have a lot of constraints and we would like to solve for a possible configuration within those constraints. This particular problem under what are known as “Packing” problems.

Task 2: Give a large class of GCP’s which can be expressed as LP’s.

How does Computational Geometry differ from what we are going here? Computational Geometry deals with all the problems that have combinatorial means of solution. There are problems for which we do not have purely combinatorial solutions. Such failure is due to the fact that we need to consider some algebra (which CG usually ignores). We are concerned with such problems here.

Book Tip: H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Heidelberg, Germany, 1987: This book deals with computational geometry in a nice way, in that it gives you a single characterization of all problems that arise in computational geometry. It then explores the subject in terms of that characterization.

The Convex Hull Problem:



What is the convex hull of a set of points? This can be answered two ways:

- *Formally:* It is the smallest convex set containing the points.
- *Informally:* It is a rubber band wrapped around the "outside" points.

Combinatorially it is solved using algorithms like Graham's scan, Jarvis’ march (gift-wrapping), and Quick hull (Recursive formulation). The complexity of these algorithms is n.log n

Task 3:

Find LP/IP/QP formulations of

1. Convex Hull Problem;
2. Voronoi diagram.

Task 3: Fix some variables and formulate the convex hull problem. Formulate the problem of voronoi diagrams.

A Covering problem is when you want to cover a space with certain objects. Consider Delaney triangulations. We cover a given area with a certain object (triangles).

Task 4: Relate the convex hull and voronoi problems to packing and covering problems.

Examples:

V (variables) could be coordinates in R^d of the geometric primitives & the solution v^* which is subset of x^* gives an embedding in R^d

V could be some parameters such as "radius" associated with a circle (geometric primitive), "distance" associated with line segment (geometric primitive)

V could be "boolean variables" picking geometric primitives.

GCS(in this class) = GCP\Computational Geometry Problems

Computational Geometry Problems = Problems that can be solved combinatorially using discrete events w/o ever solving a poly-system whose size depends on the size of the input.

Homework: Given a GCP of size n, give the "best" algorithm's tradeoff between combinatorial part and the equation system solving. (Hint: we can assume by some combinatorial work, the largest size of the polynomial systems we need to solve is m and the highest degree is d. Refer to Renegar's paper, assume the complexity to solve is d^(m^2) .) (Please verify this assumption.)

A **Geometric Constraint problem** typically consists of an objective function that needs to be maximised or minimised, given a set of equations and inequalities that have to be satisfied. These equations and inequalities are made up of "Primitives".

Primitives Suppose your system involves circles, then we might include the radius of the circle as a primitive and the co-ordinate of the center as another primitive. Suppose you are interested in finding the convex hull of a set of points, then we can have a boolean variable that indicates whether a point is included or excluded from the final collection (of the points that are in the hull).

Note :- Earlier we made an observation that Computational Geometry solves problems that do not involve Algebra. That statement needs to be further qualified like this : Computational Geometry Problems can and do involve some algebra, but in all these cases the the equation solving part **does not depend** on the siez of the input.

More formally A Geometric Constraint satisfication Problem(GCP) is specified as a system of polynomial equalities & inequalities s.t. solution to them corresponds to instantiation of variables associated with a set of geometric primitives.

One of our primary GOALs in Geometric Constraint Solving would be to reduce the siez of the systems that we need to solve.

Task 5 : What is the complexity for solving quadratic constraints ?

Task 6 : We know that in GC problems, we are supposed to constantly decide between pushing work to and from the combinatorial part and the algebra part. Establish some method to weigh the amount of work that it takes in either parts.