6G6Z1109: Software Agents and

Term 2, Le

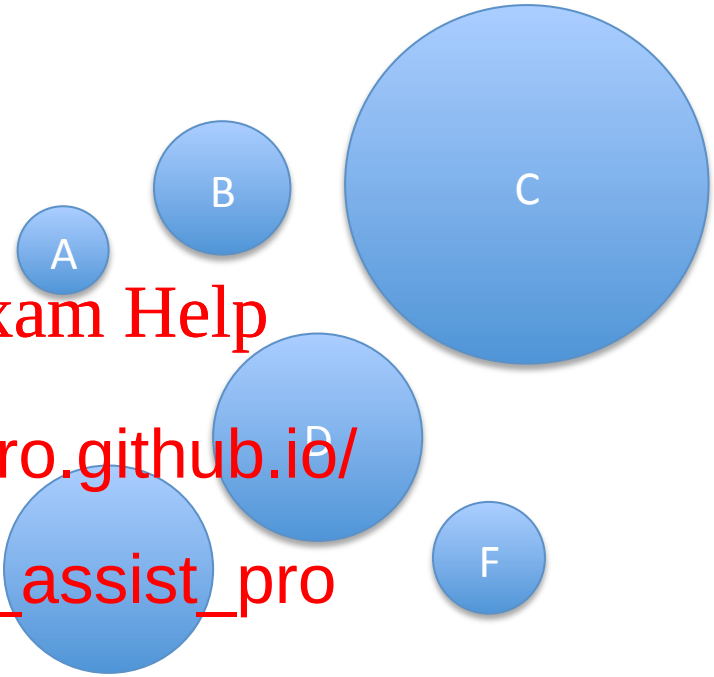
Local Search

Local search

- For many problems, we can improve the quality of genetic search by somehow *embedding* problem into the algorithm
- Benefits:
  - Reduces the probability of "illegal" solutions
  - Simplifies the representation scheme
  - Speeds up search/reduces size of solution space

An example

- Imagine we wish to pack a collection of *non-overlapping* circles with specified rad *smallest pos containing radius* (i.e., the assignment)
- How might we *represent* a solution to the problem in a GA?



One idea

- Represent a solution as a set of x,y *coordinate* for each circ point

A	B	C	D	E	F
x,y	x,y	x,y	x,y	x,y	x,y

One idea

- Represent a solution as a set of x,y *coordinate* for each circ point

A	B	C	D	E	F
x,y	x,y	x,y	x,y	x,y	x,y

Problems with this approach?

First problem

- For a (e.g.) 30x30 discrete space, there are 30x30=90 possible locations for each circle
- 90x90x90x90 possibilities of for n circles
- n=10, 90^n=34,867,844,010,000,000,000
- Problem 1: *size of search space* for even modest probem instances

First problem

- For a (e.g.) 30x30 discrete space, there are 30x30=90 possible locations for each circle
- 90x90x90x90 possibilities of for n circles
- n=10, 90^n=34,867,844,010,000,000,000
- Problem 1: *size of search space* for even modest probem instances

Next problem?

Second problem

- Nothing in this encoding prevents circles from *overlapping*
- *Illegal* soluti actually *muc* more likely than legal ones, as legal solutions require *all* circles to be disjoint
- Problem 2: *illegal* solutions are the norm

Second problem

- Nothing in this encoding prevents circles from overlapping
- *Illegal* soluti actually *muc* more likely than legal ones, as legal solutions require *all* circles to be disjoint
- Problem 2: *illegal* solutions are the norm

Which type of encoding would address both of these problems?

Order-based encoding

- Recall that an *order-based* encoding specifies a s
  “things”/mo
- We can use this type of encoding to specify the *order in which circles are placed*

Order-based encoding: example

- If we assume that the first circle is placed in the centre o then an ord CDAEFB is shown to the right

Order-based encoding: example

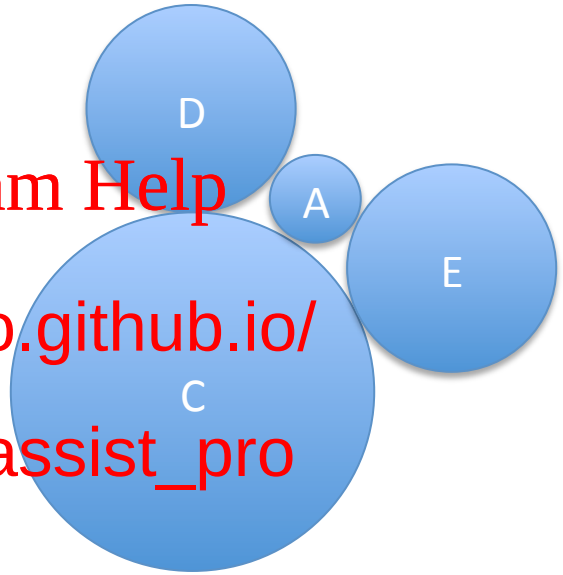
- If we assume that the first circle is placed in the centre o then an ord CDAEFB is shown to the right

Order-based encoding: example

- If we assume that the first circle is placed in the centre o then an ord CDAEFB is shown to the right

Order-based encoding: example

- If we assume that the first circle is placed in the centre o then an ord CDAEFB is shown to the right



Order-based encoding: example

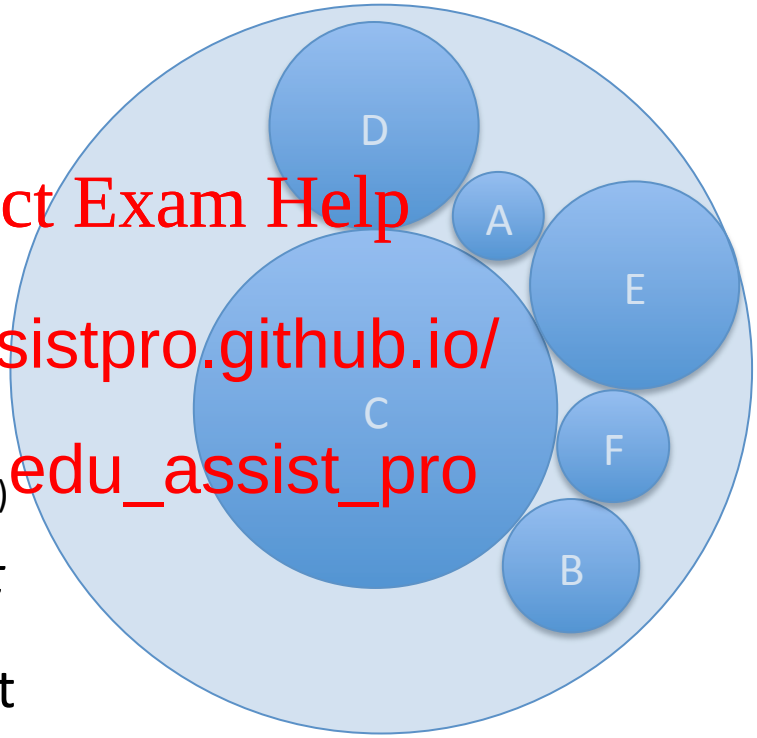
- If we assume that the first circle is placed in the centre o then an ord CDAEFB is shown to the right

Order-based encoding: example

- If we assume that the first circle is placed in the centre o then an ord CDAEFB is shown to the right

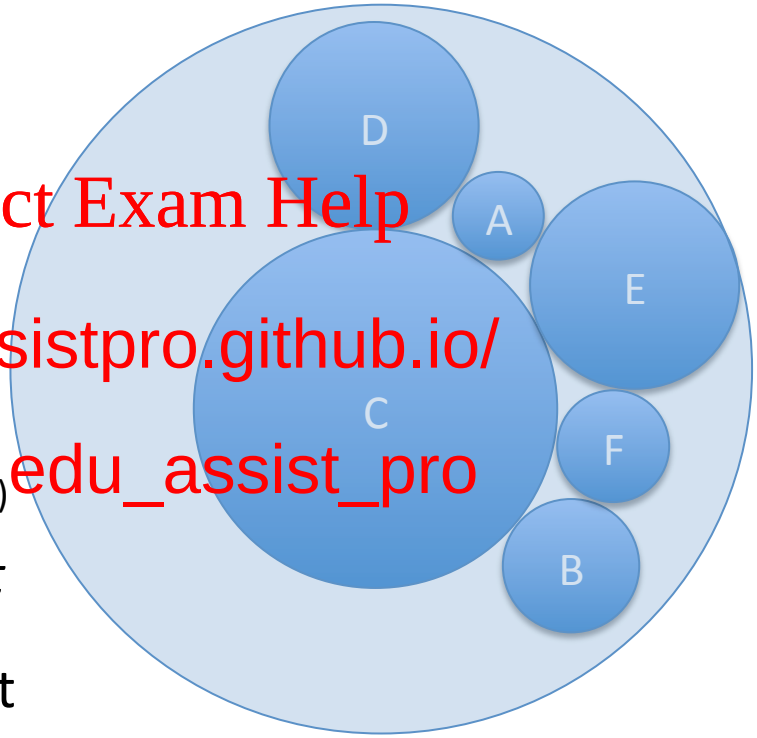
Order-based encoding: benefits

- If we get the placement algorithm right, then this encoding *automatically* prevents illegal solutions
- Number of possible solutions/ permutations of
- 10!=3,628,800
- *Significantly* less than 90^{10} (90^{10}=34,867,844,010,000,000,000)
- Placement ordering is *independent* of size of 2D space in which circles are placed (unlike direct placement encoding)



Order-based encoding: benefits

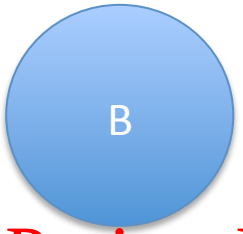
- If we get the placement algorithm right, then this encoding *automatically* prevents illegal solutions
- Number of possible solutions/ permutations of
- 10!=3,628,800
- *Significantly* less than 90^{10} (90^{10}=34,867,844,010,000,000,000)
- Placement ordering is *independent* of size of 2D space in which circles are placed (unlike direct placement encoding)

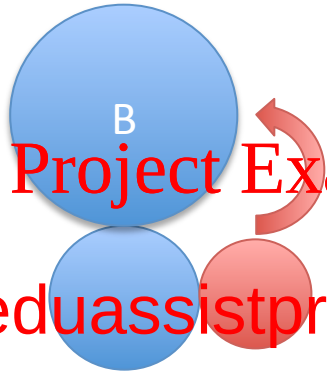


This is our "local search". For each circle in turn, we "search locally" for the best place in which to put it…

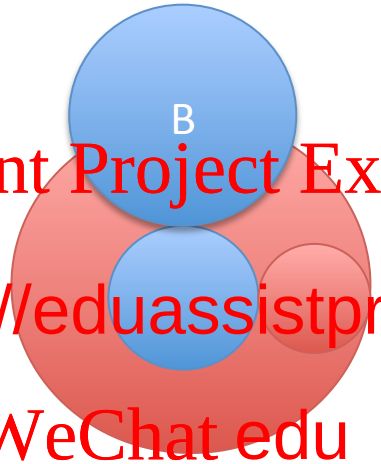
Circle placement

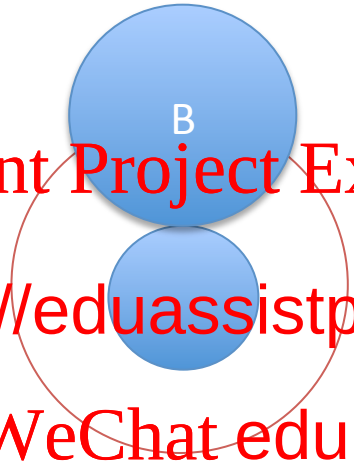
- Assume a set of circles, with specified radii
- We place the first circle in the centre of the space
- Where do w



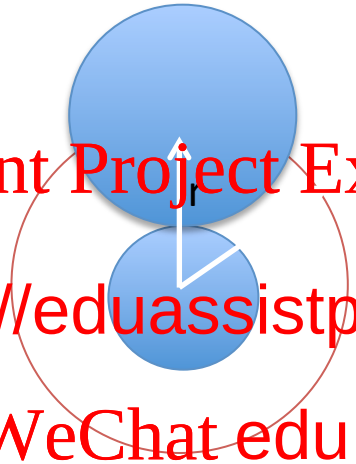


The summed radii of the circles defines a "shell" of valid placement points for B….





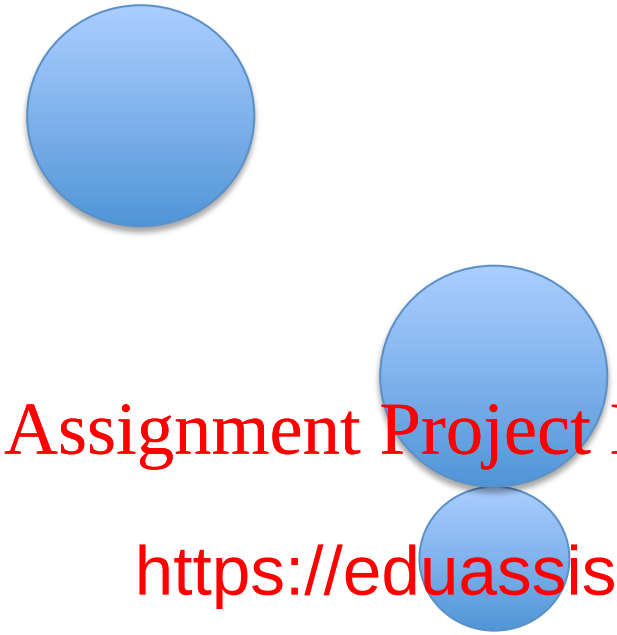
Given a point x,y, a radius r, and an angle, g, we can find the location of the point on the circle using:



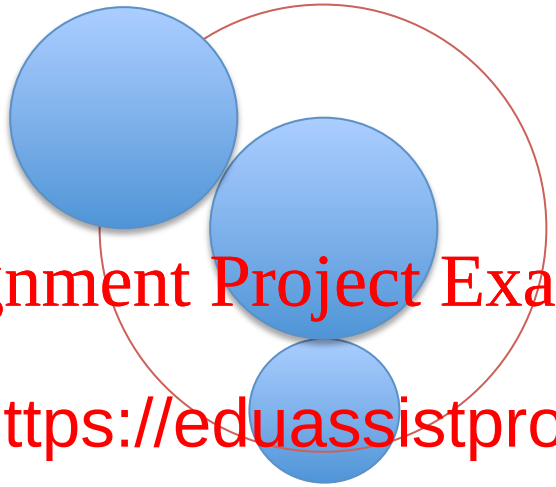
px=x + cos(a) * r
py=y + sin(a) * r

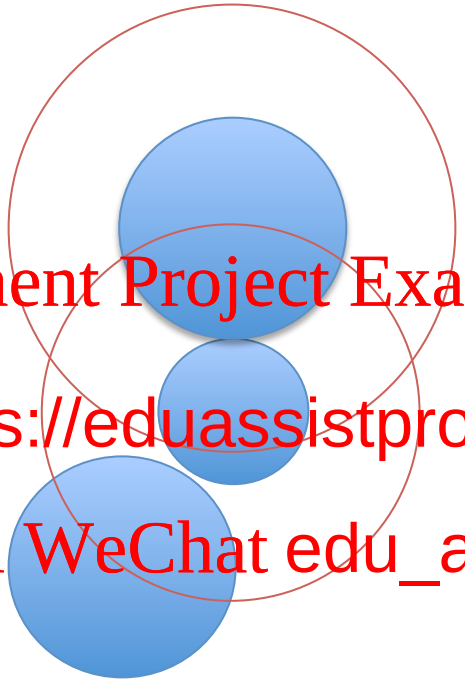
NB: all angles must be in *radians*

NB: r must be equal to the radius of a *plus* the radius of b

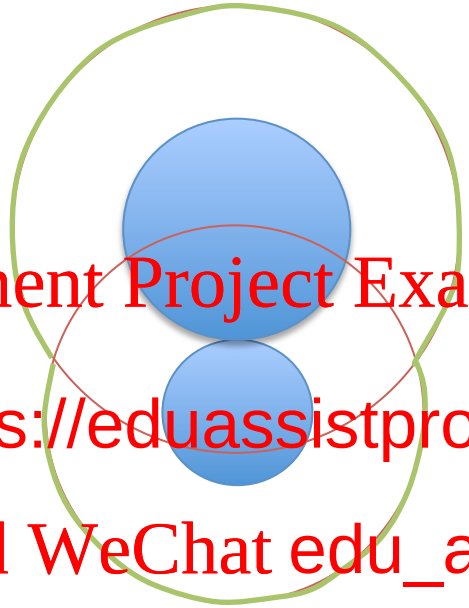


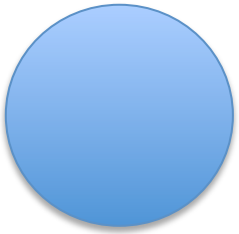
When adding a *new* circle, we start by calculating the set of “shells” that must exist for each *existing* circle…



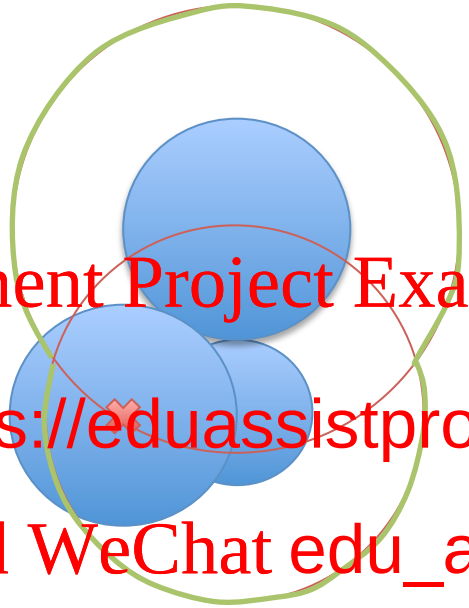


Green lines represent the set of *open points*; that is, given a circle to be added, open points represent the sum of all of the "shells" minus the points on the shells that would place th new circle overlapping an circle



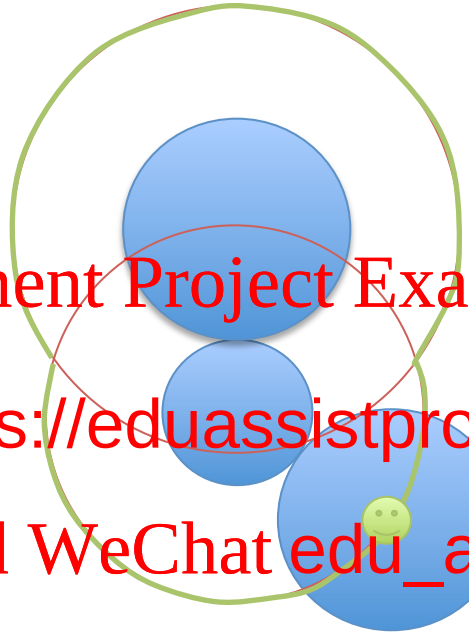


Green lines represent the set of *open points*; that is, given a circle to be added, open points represent the sum of all of the "shells" minus the points on the shells that would place th new circle overlapping an circle

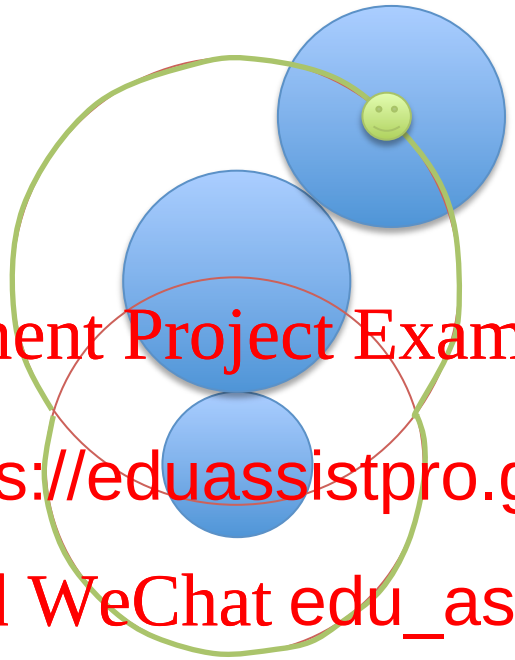


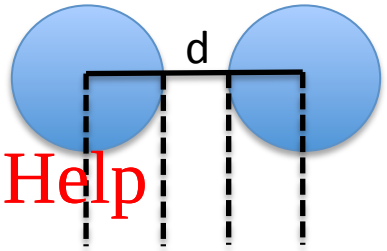
Green lines represent the set of *open points*; that is, given a circle to be added, open points represent the sum of all of the "shells" minus the points on the shells that would place th new circle overlapping an circle

Green lines represent the set of *open points*; that is, given a circle to be added, open points represent the sum of all of the “shells” minus the points on the shells that would place th new circle overlapping an circle



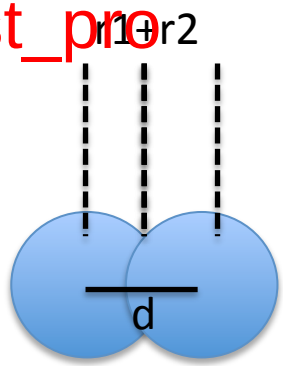
Green lines represent the set of *open points*; that is, given a circle to be added, open points represent the sum of all of the "shells" minus the points on the shells that would place th new circle overlapping an circle





How do we check to s

If the distance, d, between their centre poi
between the sum and the difference of the
then they overlap

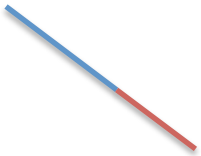


To place the circle, we simply find the open point closest to the *centre* of the space…

Create the “shell” of open points around all other *existing* circles, based on the radius of the circle, c, currently being placed

Accepts an array of the circles, returns the list of open points (only in order to draw them)

Find the open point closest to
the centre of the space (cx, cy)



Next lecture

- Next week: Comparative analysis of algorithms, more help with the assignment
- This week's assignment