

```

1. ##### CaseStudyCplex.py #####
2. ## Program running the bi-objective model for
3. ## "Designing Spatially Cohesive Nature Reserves With Backup Coverage (IJGIS)
4. ## Authors: Delmelle, Desjardins, Deng
5. ## Inputs: Study area, distance between parcels/patches
6. ##     weareGrass_Area.txt: information on each patch, including ID, location and area

7. ##     weareGrass_weight.txt: information of biodiversity for each patch
8. ##     weareGrass_Dist.txt: distance between each patch, using vertices
9. ## Outputs: LP file
10. ## Contact: delmelle@gmail.com
11. #####
12.
13.
14. #import modules
15. import random, string, os, math, time
16.
17. #define distance function
18. def distance(x1, y1, x2, y2):
19.     dx = x2 - x1
20.     dy = y2 - y1
21.     d = math.sqrt(dx**2 + dy**2)
22.     return d
23.
24. ##### INPUTS #####
25.
26. #extent of study area; for predefined grid
27. left = 947994
28. right = 991752
29. bottom = 187782
30. top = 237660
31.
32. alpha = .95 #weighting scheme; equation 1
33. K=3 #number of clusters (K); equation 6
34. interval=20000 #discretization level
35. S = 10000 #separation distance; equation 7
36. M=50000000 #a very large number
37. A=.5 #percentage (ratio) of total area to set aside (A); equation 9
38. F=.16 #minimum percentage (ratio) of total area to set aside (F-
    note that F * K cannot exceed A); equation 8
39.
40.
41. ##### OUTPUT #####
42. outputFileName = "C:\\temp\\casestudy_weareAlpha" + str(alpha) + ".lp" #naming conventio
    n: concatenate alpha value.
43. outputFile = open(outputFileName,"w")
44.
45.
46. ##### INFORMATION ON PARCELS (or PATCHES) #####
47. ##creating empty vectors
48. areaList = []
49. parcelID = []
50. parcelAREA=[]
51. parcelX=[]
52. parcelY=[]
53.
54. ##Read and store information on area, coordinates and ID of parcels
55. areaFile = open("weareGrass_Area.txt",'r')
56. for line in areaFile:
57.     areaList.append(line.strip("\n").split("\t"))
58. for row in areaList:

```

```

59.     parcelID.append(int(row[0]))
60.     parcelAREA.append(float(row[1])+0.0001) #in case some parcels area are = 0
61.     parcelX.append(float(row[2]))
62.     parcelY.append(float(row[3]))
63. print "area read"
64.
65. ##Read and store information on biodiversity within each parcel
66. weightID=[]
67. weightVALUE=[]
68. weightFile = open ("weareGrass_weight.txt",'r')
69. for line in weightFile:
70.     a = line.strip("\n")
71.     b = a.split("\t")
72.     weightID.append(int(b[0]))
73.     weightVALUE.append(float(b[1]))
74. #Normalize biodiversity
75. NormBio=[]
76. for i in weightVALUE:
77.     bioVal = ((i-min(weightVALUE))/(max(weightVALUE)-min(weightVALUE))+0.00001)
78.     NormBio.append(bioVal)
79. print "biodiversity read and normalized"
80.
81. ## Reading distances between parcels (distance must be pre-calculated)
82. distanceFile = open("weareGrass_Dist.txt",'r')
83. oriID = []
84. desID = []
85.
86. ## Storing distances between parcels (average, minimum and maximum)
87. avgDist = [0] * len(parcelID)
88. minDist = [0] * len(parcelID)
89. maxDist = [0] * len(parcelID)
90. for i in range(0,408,1):
91.     avgDist[i] = [0] * len(parcelID)
92.     minDist[i] = [0] * len(parcelID)
93.     maxDist[i] = [0] * len(parcelID)
94.
95. for line in distanceFile:
96.     a = line.strip("\n")
97.     b = a.split("\t")
98.     oriID.append(int(b[0]))
99.     desID.append(int(b[1]))
100.    minDist[int(b[0])][int(b[1])]=float(b[2])
101.    maxDist[int(b[0])][int(b[1])]=float(b[3])
102.    avgDist[int(b[0])][int(b[1])]=float(b[4])
103. print "distances between patches read"
104.
105.
106.#### INFORMATION ON RESERVES (or CLUSTERS) ####
107.##creating empty vectors and store ID, location (generated on a grid)
108.clusterList = []
109.centerX = []
110.centerY = []
111.centerID = []
112.counter = 0
113.for i in range(left,right,interval):
114.    for j in range(bottom,top,interval):
115.        centerID.append(counter)
116.        centerX.append(int(i))
117.        centerY.append(int(j))
118.        counter+=1
119.

```

```

120.
121.### DISTANCE COMPUTATION ###
122.## Computing euclidean distances between parcels/patches and centers of cluster/reserve

123.Dist =[]
124.i=0
125.while i< len(parcelX):
126.    j=0
127.    while j<len(centerX):
128.        Dist.append(float(distance(parcelX[i], parcelY[i], centerX[j], centerY[j])))
129.        j=j+1
130.    i=i+1
131.
132.##Normalize Distances
133.print "minDist=" + str(min(Dist))
134.print "maxDist=" + str(max(Dist))
135.NormDist=[]
136.for i in range(0,len(Dist), 1):
137.    NormVal =(float(Dist[i])-float(min(Dist)))/(float(max(Dist))-
float(min(Dist))+0.001)
138.    NormDist.append(NormVal)
139.print "distances between clusters and patches read and normalized"
140.
141.
142.#### WRITING LP FORM ####
143.# 1. Objective function (equations 1, 2 and 3)
144.outputFile.write("Minimize\n")
145.
146.for i in range(0,len(parcelX), 1):
147.    for r in range(0,len(centerX), 1):
148.        b = math.pow(float(NormDist[(i*len(centerID))+r]), 1) #change power here
149.        b = float(b*alpha)
150.        if b<0.000001:
151.            b=0.000001
152.        outputFile.write(" + " + str(b)+" X")
153.        outputFile.write(str(int(parcelID[i]))+"_"+str(int(centerID[r])))
154.
155.
156.    b = (float(NormBio[i]*(1-alpha)))
157.    if b<0.000001:
158.        b=0.000001
159.    outputFile.write(" - " + str(b)+" B")
160.    outputFile.write(str(int(parcelID[i])))
161.print "objective function done"
162.
163.
164.# 2. Constraints
165.outputFile.write("\nSubject to")
166.outputFile.write("\n")
167.
168.#\ keep track of f1
169.for i in range(0,len(parcelX), 1):
170.    for r in range(0,len(centerX), 1):
171.        b = math.pow(float(NormDist[(i*len(centerID))+r]), 1) #change power here
172.        outputFile.write(" + " + str(b)+" X")
173.        outputFile.write(str(int(parcelID[i]))+"_"+str(int(centerID[r])))
174.outputFile.write(" -f1 = 0\n")
175.
176.#\ keep track of f2
177.for i in range(0,len(parcelX), 1):
178.    b = NormBio[i]

```

```

179.     if b<0.000001:
180.         b=0.0001
181.         outputFile.write(" + " + str(b)+" B")
182.         outputFile.write(str(int(parcelID[i])))
183. outputFile.write(" - f2 = 0\n")
184. print "tracking f1 and f2 done"
185.
186. outputFile.write("f1 > 0 \n")
187. outputFile.write("f2 > 0 \n")
188. print "f1 and f2 must be greater than zero"
189.
190.
191. # definition of state variables
192. outputFile.write("\n")
193. outputFile.write("\ define state variable Z\n")
194. r=0 #cluster r
195. while r < len(centerID):
196.     s = 0 #cluster s, s=r+1
197.     while s<len(centerID):
198.         i=0 #parcel i
199.         while i< len(parcelID):
200.             j=i #parcel j
201.             while j< len(parcelID):
202.                 if (r != s) and (j != i):
203.                     outputFile.write("Z" + str(int(parcelID[i]))+"_" +str(int(centerID[r
204. ])) + "_" + str(int(parcelID[j]))+"_" +str(int(centerID[s])))
204.                     outputFile.write(" + X" + str(int(parcelID[i]))+"_" +str(int(centerI
205. D[r])) + " + X" + str(int(parcelID[j]))+"_" +str(int(centerID[s])))
205.                     outputFile.write("=2\n")
206.                         j+=1
207.                 i+=1
208.                 s+=1
209.             r+=1
210.         #print i
211.
212.
213. # Equation 4: when a patch i is selected, it can belong to at most one reserve
214. i=0
215. while i< len(parcelID):
216.     r=0
217.     while r < len(centerID):
218.         outputFile.write("+")
219.         outputFile.write("X")
220.         outputFile.write(str(int(parcelID[i]))+"_" +str(r))
221.         r+=1
222.         outputFile.write("-B")
223.         outputFile.write(str(int(parcelID[i])))
224.         outputFile.write("=0\n")
225.         i+=1
226. outputFile.write("\n")
227.
228. # equation 5: Each parcel must be assigned to only one cluster
229. outputFile.write("\n")
230. outputFile.write("\ parcel be assigned to a cluster iff cluster exist \n")
231. i=0
232. while i< len(parcelID):
233.     r=0
234.     while r < len(centerID):
235.         outputFile.write("X")
236.         outputFile.write(str(int(parcelID[i]))+"_" +str(int(centerID[r])) + ' - ')
237.         outputFile.write("Y"+str(int(centerID[r])) + "<=0\n")

```

```

238.         r=r+1
239.         i=i+1
240.print "constraint that each parcel must be assigned to only one cluster/reserve... done
"
241.
242.
243.## equation 6: Number of clusters (reserves) should equal to K (p in the formulation)
244.outputFile.write("\n")
245.outputFile.write("\ total number of clusters \n")
246.r=0
247.while r < len(centerID):
248.    outputFile.write("+")
249.    outputFile.write("Y"+str(int(centerID[r])))
250.    r=r+1
251.outputFile.write("=" + str(K))
252.print "constraint stipulating the number of clusters/reserves... done"
253.
254.
255.# equation 7: If a parcel i belongs to cluster r and parcel i' belongs to cluster r', t
hen
256.# i and i' must be separated by a minimum distance S.  $d(i,i',r,r') \geq S - (2 - X_i - X_{i'}) * M$ 
257.outputFile.write("\n")
258.outputFile.write("\ minimum distance between parcels\n")
259.
260.r=0 #cluster r
261.while r < len(centerID):
262.    s = 0 #cluster s, s=r+1
263.    while s<len(centerID):
264.        i=0 #parcel i
265.        while i< len(parcelID):
266.            j=i #parcel j
267.            while j< len(parcelID):
268.                if (r != s) and (j != i):
269.                    b = round(float(minDist[i][j]), 2)
270.                    outputFile.write(str(M) + " Z" + str(int(parcelID[i]))+"_" +str(int(
centerID[r])) + "_" + str(int(parcelID[j]))+"_" +str(int(centerID[s])))
271.                    outputFile.write(">=" + str(S-b) + "\n")
272.                    j+=1
273.                    i+=1
274.                    s+=1
275.                r+=1
276.
277.
278.# equation 8: cluster (reserve) should contain at least a certain percentage (F) of the
total area
279.outputFile.write("\n \ a cluster/reserve should contain at least a percentage (F) of th
e total area")
280.outputFile.write("\n")
281.r=0
282.while r < len(centerID):
283.    i=0
284.    while i< len(parcelID):
285.        outputFile.write("+ " + str(parcelAREA[i]))
286.        outputFile.write(" X")
287.        outputFile.write(str(int(parcelID[i]))+"_" +str(int(centerID[r])))
288.        i+=1
289.        outputFile.write("-
" + str(float(F) * sum(parcelAREA)) + " Y" + str(int(centerID[r])) + ">=0\n")
290.        r+=1
291.outputFile.write("\n")

```

```

292.print "cluster should contain at least F parcels"
293.
294.
295.
296.# equation 9: total area of parcels which are kept should be equal a percentage (A) of
    the total area
297.outputFile.write("\n \ total number of area set aside should be should be equal  a perc
    entage (A) of the total area")
298.outputFile.write("\n")
299.i=0
300.while i < len(parcelID):
301.    outputFile.write("+" + str(parcelAREA[i]))
302.    outputFile.write(" B")
303.    outputFile.write(str(int(parcelID[i])))
304.    i+=1
305.outputFile.write(">=" + str(float(A) * sum(parcelAREA)) + "\n")
306.print "total area greater than A"
307.
308.
309.
310.# equation 10: General constraints
311.outputFile.write("\n \n")
312.outputFile.write("GENERAL\n")
313.
314.r=0 #cluster/reserve r
315.while r < len(centerID):
316.    s = 0 #cluster/reserve s, s=r+1
317.    while s<len(centerID):
318.        i=0 #parcel/patch i
319.        while i < len(parcelID):
320.            j=i #parcel/patch j
321.            while j < len(parcelID):
322.                if (r != s) and (j != i):
323.                    outputFile.write("Z" + str(int(parcelID[i]))+"_" +str(int(centerID[r
    ])) + "_" + str(int(parcelID[j]))+"_" +str(int(centerID[s])))
324.                    outputFile.write("\n")
325.                    j+=1
326.                    i+=1
327.                    s+=1
328.                r+=1
329.print "constraint: integer constraint...done"
330.outputFile.write("\n \n")
331.
332.
333.
334.# equation 10: Binary constraints
335.outputFile.write("BINARY\n")
336.i=0
337.while i < len(parcelID):
338.    r=0
339.    while r < len(centerID):
340.        outputFile.write("X")
341.        outputFile.write(str(int(parcelID[i]))+"_" +str(int(centerID[r])))
342.        outputFile.write("\n")
343.        r +=1
344.        i=i+1
345.print "constraint: binary constraint...done"
346.
347.
348.i=0
349.while i < len(parcelID):

```

```
350.     outputFile.write("B"+str(int(parcelID[i]))+"\n")
351.     outputFile.write("\n")
352.     i=i+1
353. print "constraint: binary constraint...done"
354.
355.
356. r=0
357. while r < len(centerID):
358.     outputFile.write("Y")
359.     outputFile.write(str(int(centerID[r])))
360.     outputFile.write("\n")
361.     r +=1
362.
363. outputFile.write("END\n")
364. outputFile.close()
```